CICS TS Performance Tutorial – I/O Tuning

Eugene S. Hudders
C\TREK Corporation
eshudders@aol.com

407-469-3600

August 4, 2010

SHARE in Boston

DISCLAIMERS/TRADEMARKS

- **YMMV**
- **Remember the Political Factor**
- **CICS/VS, CICS/MVS, CICS/ESA, CICS TS, COBOL LE, COBOL 2, VSAM, DB2, OS/390, MVS, z/OS and z/VSE Are Trademarks of the International Business Machines Armonk, NY**

Agenda

- Introduction
- NSR
  - Introduction
  - NSR File Definitions
  - NSR Buffer Definitions
  - NSR Performance
  - Recommendations
- LSR
  - Robin Hood Theory
  - Introduction to LSR
  - LSR Tuning Areas
  - Overlooked LSR Tuning Areas
  - Recommendations
- Closing

Introduction

- CICS uses two techniques to handle VSAM files within CICS TS:
  - Non-Shared Resources (NSR)
  - Local Shared Resources (LSR)
- In recent years, new VSAM features announced for CICS have been LSR oriented
- The major difference between the two techniques lies in the "ownership" of the resources
  - NSR → resources are used exclusively by the file
  - LSR → resources are shared between participating files
- Note: There is an error in the CICS Performance Guide regarding CA splits and their effect on how it can tie up the main task TCB for NSR files – this information is in error
  - There is no TCB lockout as stated in the manual
  - Applies to z/OS as well as z/VSE

Introduction

- **I/O generates CPU usage**
  - **CICS to**
  - **VSAM to**
  - **SVC Handler to**
  - **IOS**
  - **Start the I/O and eventually back to**
  - **CICS to have task wait**
  - **Process I/O Interrupt**
  - **Create SRB**
  - **Dispatch the SRB to Post Completion**
  - **To the CICS Dispatcher that dispatches the task when its turn occurs**
- **To improve response time and reduce CPU overhead, you need to eliminate I/O**
  - **Find the data/index in a buffer called a Look-Aside Hit**
  - **CPU requirements for a Look-Aside Hit is much lower**

Non-Shared Resources

NSR

Introduction to NSR

- **NSR advantages include:**
  - **Resources are reserved so one file can be specifically tuned**
  - **Allows for chained read operations that can give better sequential performance**
    - **BROWSE**
    - **CA Splits**
    - **Mass inserts**
- **Does not support Transaction Isolation**
- **Does not support VSAM Threadsafe**
- **NSR = BATCH Processing**

NSR File Definition

- A file is defined as NSR by specifying LSRPOOLID (NONE)
- String number defines the number of concurrent file accesses allowed
- One BUFND and one BUFNI is required per string
- Minimum buffer allocations:
  - BUFND is string number plus one
    - Extra buffer is used for split processing
  - BUFNI is string number

NSR File Definition

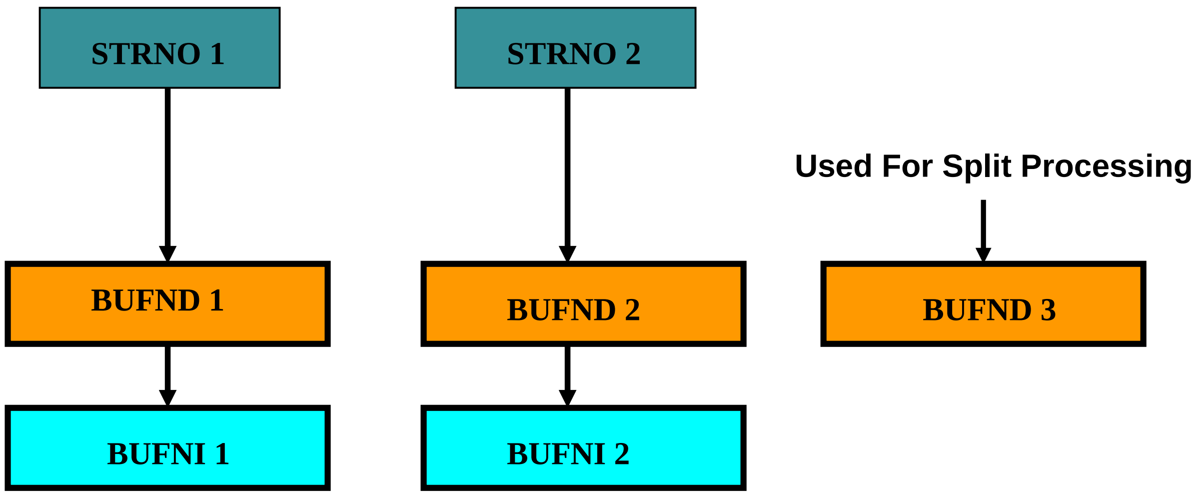
- String definition for an NSR file can be a challenging task
  - Many NSR files are over allocated in strings when considering the I/O activity against the file
  - The major reason is that NSR allows duplicate CIs to exist between strings
  - NSR allows STARTBR/READNEXT/READ for UPDATE sequence without an intervening ENDBR
    - This results in two strings being allocated to the task
    - The requested CI appears 2X in VS
    - As a result, many files would appear to be deadlocked due to lack of strings
    - This type of request will not work in LSR
  - Remember that a string needs a BUFND/BUFNI
    - Eliminate strings in favor of more buffers

NSR File Definition

- Additional buffers can be allocated
  - Extra BUFND – will be used in sequential operations
    - All available buffers will be allocated to the 1st sequential request
  - Extra BUFNI – will be used to store Index Set (IS) indices (high level indices)
  - Sequence Set Indices (SSI) are never read into the extra BUFNIs
    - SSI CIs are read into the string index buffer
    - No look aside to other string buffers are done

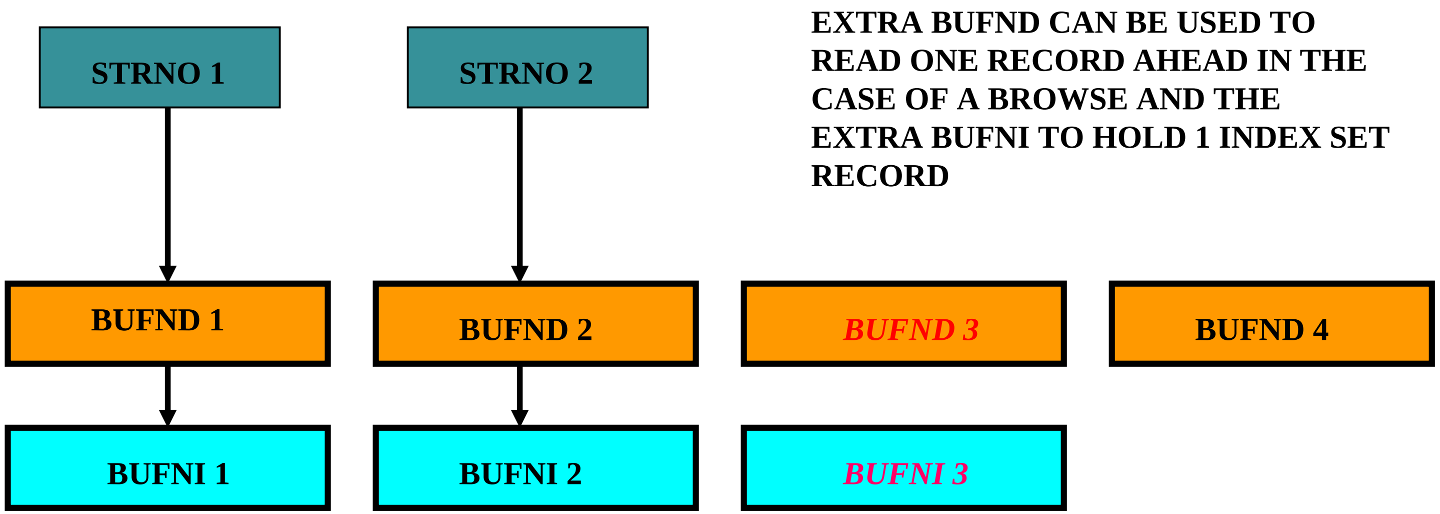
NSR Buffer Definition

- **Example # 1:**
  - STRNO = 2 BUFND = 3 BUFNI = 2



NSR Buffer Definition

- **Example # 2:**
  - STRNO = 2 BUFND = 4 BUFNI = 3



NSR Performance

- **Why would you want extra BUFNDs?**
  - **In the case of a BROWSE request, to read ahead a number of CIs to improve performance of the task**
  - **In the case of Mass Insert, to write behind a series of CIs to improve task performance**
  - **In the case of CA Splits, to be able to move more than one CI at a time to the new CA**
  - **Overall, extra data buffers can speed up the process and reduce I/O requests to the file**

NSR Performance

- **What is the hidden agenda?**
  - **Browse**
    - **The number of BUFNDs defined should contain the approximately the same number of records read (READNEXT) by the program**
      - ***For example, if a CI can contain 5 records and the average # of READNEXT operations issued is 20, then a BUFND specifying 4 additional buffers (5 records/CI*4 read ahead buffers) would be fine***
      - ***However, what programmer knows on the average how many READNEXT operations are issued to a file?***
      - ***Also, only the 1st BROWSE request would benefit***
      - ***What happens if the BROWSE is ended (ENDBR) before the 20 READNEXT operations are done?***
    - **Adding additional buffers for sequential BROWSE processing will increase the task response time plus unneeded I/O operations may result**
    - **In addition, having the data in storage is good for this task but may affect the response of other tasks in the system**

NSR Performance

- **Mass Inserts**
  - **The number of buffers should be around the same number of writes (WRITE) issued to the file at one time**
    - ***Same logic as the BROWSE***
  - **However, if the number of writes ends before all the buffers are full, then there is no I/O penalty as in the case of a BROWSE**
- **CA Splits**
  - **The number of buffers should be large enough to copy ½ of a CA at time**
    - ***However, if the file does Mass Inserts or BROWSE operations, there is no way to segregate the buffers for one particular use***

NSR Performance

- **What is the best approach for files that are heavily or mainly browsed?**
  - **If too many buffers are read, performance of other tasks may be affected**
  - **The key is to try and get a CISZ that generally accommodates the # of READNEXT commands issued**
    - **If too many, try to get a large multiple**
  - **This approach can be used for LSR pool files too**

NSR Performance

- **Why would you want extra BUFNIs?**
  - **Two types of index look asides occur for an NSR file**
    - **The 1st look aside is for the Index Set records that are in extra BUFNI buffers**
    - **The 2nd look aside is within the string buffers to see if the Sequence Set Index and/or the data CI are present**
      - ***No look aside possible to other string buffers***

NSR Performance

- **Additional index buffers allows VSAM to load the Index Set records into virtual storage**
  - **User should allocate sufficient BUFNIs as there are Index Set CIs in the file**
  - **Consideration should be given to adding additional index buffers if the file reflects CA splits**
    - ***Data CA splits can cause index CA splits creating new index set records***

NSR Performance

- **Determining the number of BUFNIs required entails computing how many Sequence Set Index (SSI) records exist in the file**
  - **There is one Sequence Set Index record per data CA**
  - **This is a one to one relationship**

NSR Performance

- **Compute:**

1) # CAs = (Data HURBA / (# CI/CA*Data CISZ) this represent the # of Sequence Set Index records in the file
2) From LISTCAT get the total number of Index records in the file and determine the number of Index Set records in the file: (Total Number of Index Records – # of CAs)
3) Determine the # of BUFNIs = (Total # Of Index Set records + # of strings + CA split adjustment)
4) CA Split adjustment is any figure from zero to “n”, where “n” is the # of additional Index set records created as a result of CA splits

NSR Performance

LISTCAT Extract

Data Information

```
STATISTICS  (* - VALUE MAY BE INCORRECT)
  REC-TOTAL-----------5296*     SPLITS-CI--------------2*
  REC-DELETED------------4*     SPLITS-CA--------------1*
  REC-INSERTED----------66*     FREESPACE-%CI----------0
  REC-UPDATED-----------77*     FREESPACE-%CA----------2
  REC-RETRIEVED-----444481*     FREESPC----------2875392*
ALLOCATION
  SPACE-TYPE------CYLINDER      HI-A-RBA---------4147200
  SPACE-PRI--------------5      HI-U-RBA---------2488320
  SPACE-SEC--------------1
```

Are there any splits?

Need these two values

```
CISIZE------------18432
CI/CA-----------------45
```

Need these two values

Need the number of index records

Index Information

```
STATISTICS  (* - VALUE MAY BE INCORRECT)
  REC-TOTAL--------------4*     SPLITS-CI--------------1*
  REC-DELETED------------0*     SPLITS-CA--------------0*
  REC-INSERTED-----------0*     FREESPACE-%CI----------0
  REC-UPDATED------------4*     FREESPACE-%CA----------0
  REC-RETRIEVED----------0*     FREESPC------------29696*
ALLOCATION
  SPACE-TYPE---------TRACK      HI-A-RBA-----------33792
  SPACE-PRI--------------1      HI-U-RBA------------4096
  SPACE-SEC--------------1
```

```
INDEX:
LEVELS-----------------2
```

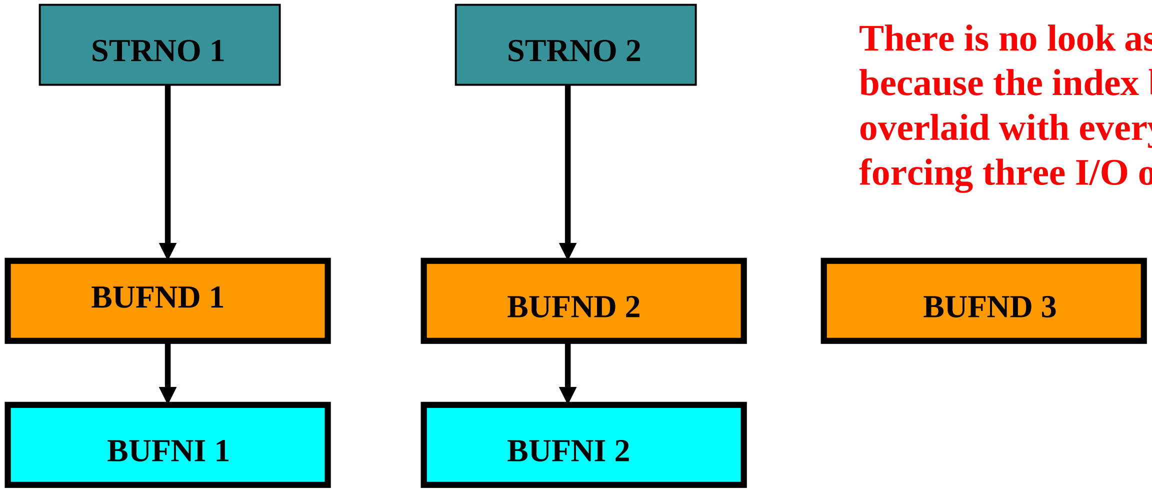
Determine number of IX Levels

NSR Performance

- Example using previous LISTCAT information
  - Data CISZ 18K (18,432)
  - CI/CA 45
  - Bytes/CA 829,440 (18432*45)
  - CA splits Yes
  - # of IX records 4
  - HURBA 2,488,320
  - # of IX levels 2
  - (2488320/829440)=3 CAs or Sequence Set Records
  - (4-3)=1 Index Set Record
  - If STRNO=5, then (5+1+2)=8 BUFNI request for the file. The +2 is a buffer for future CA splits at the index level. The CA adjustment is optional and the value can vary

NSR Buffer Definition

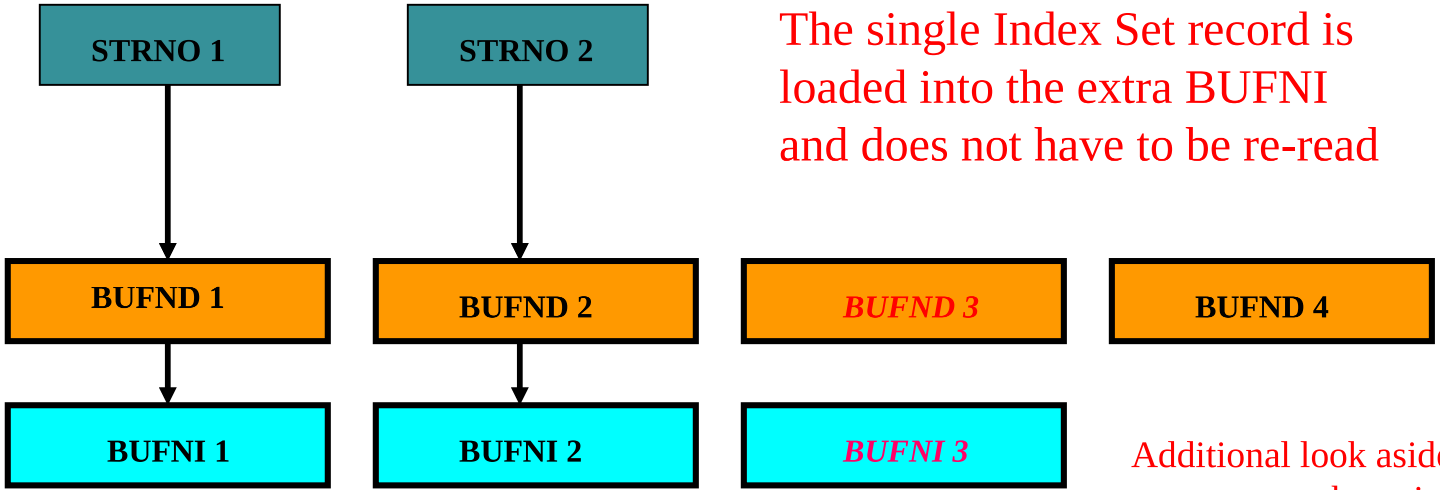
- **Example # 1 – VSAM 2 Index Levels:**
  - STRNO = 2   BUFND = 3   BUFNI = 2
  - Requires three I/Os (2 index and 1 data)
  - No opportunity for look aside



There is no look aside possible because the index buffer gets overlaid with every request forcing three I/O operations

NSR Buffer Definition

- **Example # 2 – VSAM 2 Index Levels:**
  - **STRNO = 2   BUFND = 4   BUFNI = 3**
  - **After 1st read, each request would require a maximum of two reads or a 33% I/O operations savings**



The single Index Set record is loaded into the extra BUFNI and does not have to be re-read

Additional look aside can occur at the string buffers, potentially saving additional I/O

NSR Recommendations

- NSR files should be reviewed to see why they are not in LSR for better performance
  - For example, Share Options 4 file
  - Command Level Browse restrictions
- If the file is to be in NSR
  - Ensure valid CISZ for files that are browsed
  - Ensure sufficient BUFNIs allocated to hold the entire Index Set indices in buffers
  - Ensure that excess strings are eliminated and the storage used to allocate correct file buffering
  - Do not over allocate BUFND unless the file is prone only to CA splits
- If NSR must be used and files takes CA splits, consider activating the CO TCB (SUBTSKS=1 in SIT) (multiple CPUs)
- NSR and Transaction Isolation are incompatible
- NSR is not supported under VSAM Threadsafe

Local Shared Resources

LSR

Robin Hood Theory

- Tuning LSR files is simply applying the Robin Hood theory in reverse
- In Sherwood Forrest Robin stole from the rich to give to the poor
- In LSR you steal from the poor to give to the rich!!!!!
  - Poor = Low to Medium Activity Files
  - Rich = Most Active Files
- In other words the major contribution that low activity files provide to LSR are their resources so that higher activity files can use them (Cruel Reality)

Introduction to LSR

- **LSR advantages include:**
  - **More efficient VS use because resources are shared**
  - **Better look-aside because buffers can maintain the Sequence Set Index records**
  - **Tends to be more self-tuning because buffers are allocated on an LRU basis keeping information of the more active files in the buffers at the expense of less active files**
  - **Only one copy of a CI allowed (better read integrity)**
  - **Can allocate up to 8 pools to segregate files**
  - **Supports Transaction Isolation (TI)**
  - **Supports VSAM Threadsafe (Local VSAM)**

LSR Tuning Areas

- Pool Definition – Dynamic vs. Static
  - Separate Index and Data Buffers
  - Number of Strings
  - Maximum Key Length
  - Number, Sizes and Types of Buffers
- Pool Measurement – Hit Ratios
  - Data
  - Index
  - Combined

LSR Tuning Areas

- Tuning Hit Ratio
- Overlooked LSR Tuning Areas
  - Buffer Fragmentation
  - LSR Buffer vs. File CISZ Reconciliation
  - Page Boundary Allocation
  - Buffer Pool Monopolization
  - Maximum Key Size
  - Number Strings Required
  - Number of LSR Pools
- LSR Pool Candidates
  - Share Options 4 Files
  - File Activity

Pool Definition

- Dynamic Pool Definition – No CEDA Definition
  - Advantages
    - Allows for Quick Implementation and Installation
    - Reduces System Programmer Intervention
      - *No Need to Compute CISZ vs. Buffers Required*
      - *No Need to Determine Maximum Key Length*
      - *No Need to Compute Number of Strings Required*
  - Disadvantages
    - CISZ Contention Between Data and Index – Combined Pool
    - Cannot Allocate Hipercache Buffers (If Available)
    - Allocation of Buffers Is Based on a Percentage Not Activity
    - String Allocation Based on % – Usually Over-Allocated
    - Slow CICS Initialization (First File Opened)
    - Combined Data/Index Pools Can Hide Bad Data/Index Performers

Pool Definition

- Static Definition
  - Disadvantages
    - Requires System Programmer Intervention to Determine
      - *Buffers Sizes and Quantity Required*
      - *Maximum Key Length*
      - *Number of Strings Needed*
    - Exposes System Programmer to Errors
      - *Incorrect Buffer Size Selection – Buffer Fragmentation*
      - *Incorrect String Allocation*
      - *Incorrect Maximum Key Size Specification*
    - Requires Planning – Not Everyone Likes to Do This!
      - *Must Specify Required Buffers, Maximum Key Length and Number of Strings Required – Otherwise Pool Is Dynamically Created*

Pool Definition

- **Static Definition**
  - **Advantages**
    - **Separate Pools for Data and Index Can Be Defined**
      - ***No CISZ Contention Between Data and Index***
    - **Can Optimize Buffers that Have Higher Activity**
    - **Can Optimize String and Maximum Key Size Required**
    - **Can Allocate Hiperspace Buffers**
      - ***If applicable, need more than 32K buffers of a particular buffer size***
    - **Faster CICS Initialization**

Pool Definition

- Recommendation
  - Define LSR Pools Explicitly
  - Determine Individual File Requirements
    - Data and Index (If Applicable) CISZ required
    - Maximum Length Key
    - Strings
  - Get “Big Picture” of Requirements
    - CICS Performance Tool/Monitor
    - CICS Statistics (EOD)
    - Dynamic Definition – One Time

LSR Pool Measurement

- LSR Pool Effectiveness Is Based on Look-Aside Hit Ratios
  - Generally Accepted Hit Ratios Are:
    - Data – 80%+
    - Index – 95%+
    - Combined – 93%+
- Buffer Tuning Should Concentrate on Improving the Index Hit Ratio First
  - Generally, Index I/O Requests Are Higher Than the Data
  - Real Storage Investment to Improve Index Hit Ratio Is Less Due to Smaller CISZ Associated with the Index Component

LSR Pool Measurement

- **Important Note:**
  - **LSR Buffer Attainments Can Be Misleading**
  - **If the 4 KB Buffer Reflects a Hit Ratio of 85%, This Does Not Mean That Every File Is Getting an 85% Look-Aside Hit Ratio**
  - **The 85% is an Average of All the Files Using This Buffer Size**
    - **Some Get a Higher Attainment**
    - **Others Get a Lower Attainment**

LSR Pool Measurement

- **Data Buffer Tuning Is Highly Dependent on Access Patterns**
  - **Good Look-Aside Hit Ratios Usually Requires a Substantial Storage Investment (80%+)**
  - **The Major Cause Is That the Data Component Is Usually Very Large (vs. Index Component)**
  - **Good Hit Ratios Usually Result in Files with:**
    - **Sequential Activity**
    - **Read for Update/Rewrite/Delete**
    - **Concentrated Read Activity**

LSR Pool Measurement

- **Data Buffer Tuning Is Highly Dependent on Access Patterns**
  - **Bad Hit Ratios Usually Result in Files with:**
    - **Disperse Read Activity (Very Large Files)**
    - **Share Options 4**
- **Recommendation**
  - **Buffer Tuning Is Usually a "trial and error" process in determining the number of buffers to add to each buffer size**
  - **Reiterative process**
    - **You Add Buffers**
    - **You Measure**
    - **If Objective Met, Temporary End, Else Go Back to Add Buffers**
    - **Temporary End Because Things Change and Require Periodic Observation**
  - **Tune Buffer Pools and CI Sizes Individually**
    - **Set Realistic Objectives, for Example:**
      - ***Data – 80%***
      - ***Index – 95%***
      - ***Combined – 93%***
  - **Define a Minimum of Three 32K Catch-All Buffers**

Overlooked LSR Tuning Areas

- Buffer Fragmentation
  - Only Eleven Valid CISZ for LSR Buffers (K)
    - 0.5 1.0 2.0 4.0 8.0 12.0
    - 16.0 20.0 24.0 28.0 32.0
  - Therefore, a 2.5K Byte CISZ Would Use a 4K LSR Buffer
  - If a 4K Buffer Was Not Available, Then the Next Largest Available Buffer Is Used
  - Some Fragmentation May Be Desired for Certain CISZ (e.g., non VSAM/E – 18.0K)

Overlooked LSR Tuning Areas

- Buffer Fragmentation
  - Avoid Unnecessary Fragmentation (e.g., a 6K CISZ Using a 12K Buffer)
  - Certain Default Index CISZ Should Be Forced to an LSR CISZ (e.g., 1536 to 2048 or 2560 to 4096)
  - Virtual Fragmentation Results in Real Storage Fragmentation

Overlooked LSR Tuning Areas

- **LSR Buffer vs. File CISZ Reconciliation**
  - **Best Alternative to Reducing Fragmentation**
  - **Determine File CI Sizes Required and Assign LSR Pool Buffers to Match**
    - **Number and Size of Buffers**
    - **Number of Strings (Overall)**
  - **Set CISZ Standards (If possible) for LSR Pool Files**
  - **Complex Task, If Done Manually**

Overlooked LSR Tuning Areas

- LSR Buffer vs. File CISZ Reconciliation
  - Some Installations Simply Define a Certain Number of Buffers for Every Possible Buffer Size (11 Buffer Sizes)
  - Alternate Example:
    - Suppose You Don't Have Any 16K Buffer Users (CISZ Range Is 14K and 16K files)
    - You Determine That You Want to Have Twenty 16K Buffers Defined (320 K) Just in Case One Day You Get a 14K or 16K File
    - This allocated Storage Will Not Be Used – Wasted Storage Every Day of the Year
    - Instead, Why Don't You Simply Define Sixteen 20K Buffers (320K) (or Next Useable Size) That Will Be Used Every Day

Overlooked LSR Tuning Areas

- **Page Boundary Buffer Allocation (Minor)**
  - **VSAM Requests Buffers on a Page Boundary and in Page (4K) Increments**
  - **Fragmentation That Occurs from Buffer Allocation Should Be Avoided – Loss of Virtual Storage**
  - **Allocate the Following Buffers in the Following Multiples:**
    - **0.5K Multiple of 8 (0.5K Times 8 = 4K)**
    - **1.0K Multiple of 4 (1.0K Times 4 = 4K)**
    - **2.0K Multiple of 2 (2.0K Times 2 = 4K)**

Overlooked LSR Tuning Areas

- Buffer Monopolization
  - Theory Behind LSR Is to Share Resources When Needed
    - So What Can Be Bad If the Principal Files (Most Active) Control a High Percentage of the Buffers?
    - Even at the Expense of Low Activity Files
  - How Do You Determine If a File Is Monopolizing a Particular Buffer Size?
    - I/O Activity
    - Buffer Hit Ratio
    - Number of Buffers Held (By CISZ)

Overlooked LSR Tuning Areas

- Buffer Pool Monopolization
  - Need a CICS Tuning/Monitor to Determine the Number of Buffers Being Held by a File
  - Important If Principal Files Are Not Providing a Good Response Time
- Remember the Reverse "Robin Hood" Theory
  - "Rob from the Poor to Give to the Rich"
  - The "Rich" Are Your More Important Active Files
- Point of Diminishing Return
  - Keep Adding Buffers Until Higher Activity Files Do Not Require More

Overlooked LSR Tuning Areas

- Maximum Key Size (Minor)
  - Maximum Key Size Is Important as All VSAM Control Blocks Are Shared and Must Accommodate the Largest File Key of the Shared Pool
  - If the Maximum Key Size Allocated to the Pool Is too Small, Files with Larger Keys Will Not Open
  - Many Installations Force the LSR Pool Key Size to 255 Bytes
  - Although Using This Maximum Can Waste Storage, the Actual Amount Depends on the Number of Strings Allocated Times the Excess Key Size
  - Decision is Installation Dependent

Overlooked LSR Tuning Areas

- Number of Strings Allocated
  - Probably Only Tuned When Wait on Strings Conditions Occur
    - String Waits Can Occur If
      - *Maximum Number of Strings in the Pool Is Reached*
      - *Maximum Number of Strings Assigned to the File Is Reached*
  - Many LSR Pools Strings are Over-Allocated
  - The Objective Should Be to Have Sufficient Strings to Handle Peak Periods Without Waiting for Strings
  - Try to Allocate So That the High Used String Number Is Around 50 to 60% of the Total Strings Allocated to the Pool

Overlooked LSR Tuning Areas

- Number Of Defined LSR Pools
  - Two Schools of Thought
    - School 1 – Use as Many Pools as Possible So That Files Can Be Segregated to Reduce Contention and/or Interference
    - School 2 – Use as Few as Possible Pools So That Resources Can Be Used More Efficiently
  - Considerations
    - Are the Pools Allocated with a “Fudge Factor”?
    - Which Files Are More Important So That Resources Should Be Allocated to Them?

Overlooked LSR Tuning Areas

- There Are 8 (MVS) or 15 (VSE) LSR Pools Available Since LSR Was Made Available to CICS
  - Made Sense in the Beginning Because Buffer Search Algorithm Was Sequential
  - Larger Pools Increased CPU Time to Search
  - Search Algorithm Changed – Hashing Technique
- Theory Behind LSR Is to Share Resources When Needed (Repeat)
  - So What Can Be Bad If the Principal Files (Most Active) Control a High Percentage of the Buffers?
  - Even at the Expense of Low Activity Files

Overlooked LSR Tuning Areas

- **Multiple pool considerations**
  - **Data Tables –**
    - ***Output operations go against the VSAM file***
    - ***LSR pool used for look-aside for records before going to disk***
    - ***ROT = 90%+ Read Operations***
      - ***Low activity reduce look-aside capacity***
  - **LSR VSAM Threadsafe files**
    - ***Lock mechanism may require more distribution of requests***
    - ***Multiple pools for DB2/MQ CICS regions***
    - ***In case of FOR, single pool is probably better as no VSAM Threadsafe is available (FCQROLY=YES)***

LSR Pool Candidates

- LSR Provides the Best Look-Aside Algorithm Within CICS
- Generally, Files (High, Intermediate and Low Activity) Should Be Assigned to LSR Except:
  - Share Options 4 Files
  - Files That Do Not Follow Command Level Guidelines
    - Start Browse, Read Next …..Read for Update (Non-RLS)
  - High CA Split Activity Files (Tune Independently)
- LSR Is the Gate to New File Features Within CICS

LSR Recommendations

- LSR Is Preferred Over NSR Buffering
  - Superior Look-Aside Hit Ratio
- Tuning LSR Involves:
  - Ensuring Proper Number of Buffers Defined
    - Achieve Installation Look-Aside Hit Ratio Goals
  - Eliminating Fragmentation
  - Static Definition of the Pool(s)
- Continuous Review – Especially When Major Application Changes Occur
  - VSAM Tuning

Closing

- Use LSR over NSR
- Tune to eliminate I/O – Look-Aside Hits
- Monitor File Statistics periodically to ensure that Look-Aside Hit Ratio objectives are being met
- When tuning LSR remember Robin Hood!