Virtual CICS user group: Newsletter 53

Welcome to the Virtual CICS user group newsletter. The Virtual CICS user group at www.fundi.com/virtualcics is an independently-operated vendor-neutral site run by and for the CICS user community.

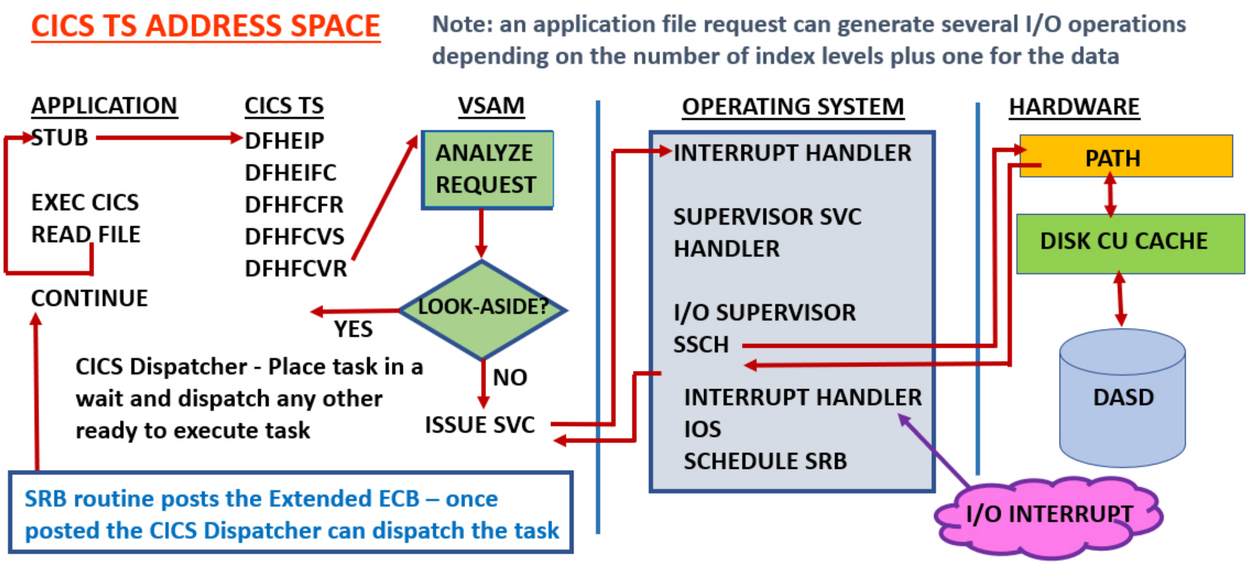


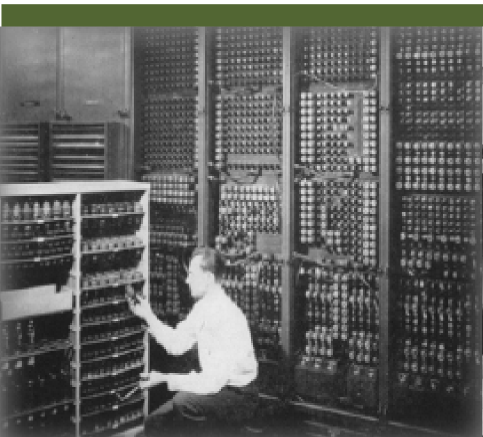
Figure 1: The very big I/O picture

Virtual CICS user group presentation

The latest webinar from the Virtual CICS user group was entitled, “LSR Tuning Today”. It was presented by Eugene S Hudders, President of C\TREK Corporation.

Gene is president of C\TREK Corporation, a company that developed C\TREK a performance and problem determination tool for CICS. He has worked on IBM mainframe computers for over 50 years. He has made presentations related to CICS and VSAM at technical conferences such as SHARE, CMG, and WAVV. In addition, Gene has written several books on CICS

Contents:



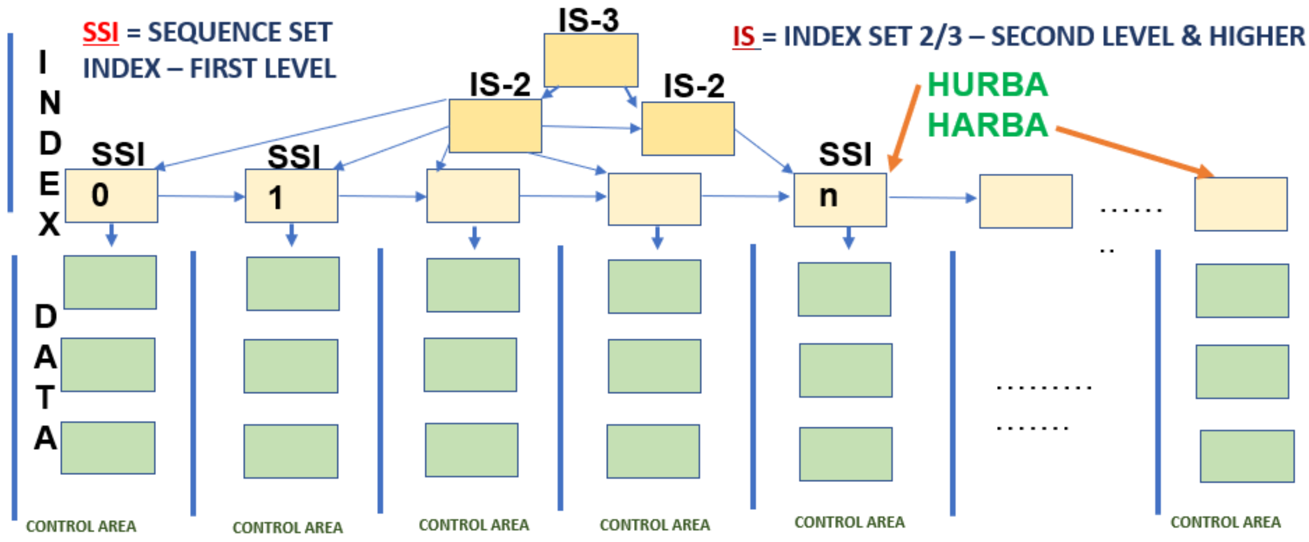


Figure 2: KSDS structure and terms

and VSAM, in the areas of problem determination and tuning, as well as many articles for technical magazines on the same topics. In his current position, Gene continues to develop new performance functions for C\TREK software and provides consulting and educational services for clients in the USA and Latin America.

Eugene S Hudders started the session by saying that CICS uses three techniques to handle VSAM files within CICS TS: Non-Shared Resources (NSR); Local Shared Resources (LSR); and Record Level Sharing (RLS). In recent years, new VSAM features announced for CICS have been LSR/RLS-oriented. The major difference between the three techniques lies in the 'ownership' of the resources. NSR resources are used exclusively by the file. LSR resources are shared between participating files. And RLS resources are in a different address space (SMSVSAM) and require a Coupling Facility (CF). CA splits tie up the main task TCB for NSR files. Gene suggested the user group might consider the use of the CO TCB (multi-processor) or consider moving the file to LSR.

The advantages of using LSR are:

- More efficient use of virtual storage versus NSR as resources are shared.
- Better look-aside hit ratio because the Sequence Set Indexes (SSI) are maintained in the buffer pool.
- Tends to be more self-tuning because buffers are allocated on a Least Recently Used (LRU) algorithm keeping the information for the more active files in buffers at the expense of lower activity files.
- Only one copy of a CI is allowed (better read integrity).
- Can allocate up to 255 pools to segregate file access. This provides more strings and buffers (VS is the limit). Best used by VSAM threadsafe (parallel access versus single thread).
- Supports Transaction Isolation.

Gene went on to say that physical I/O costs CPU cycles, and the best I/O is

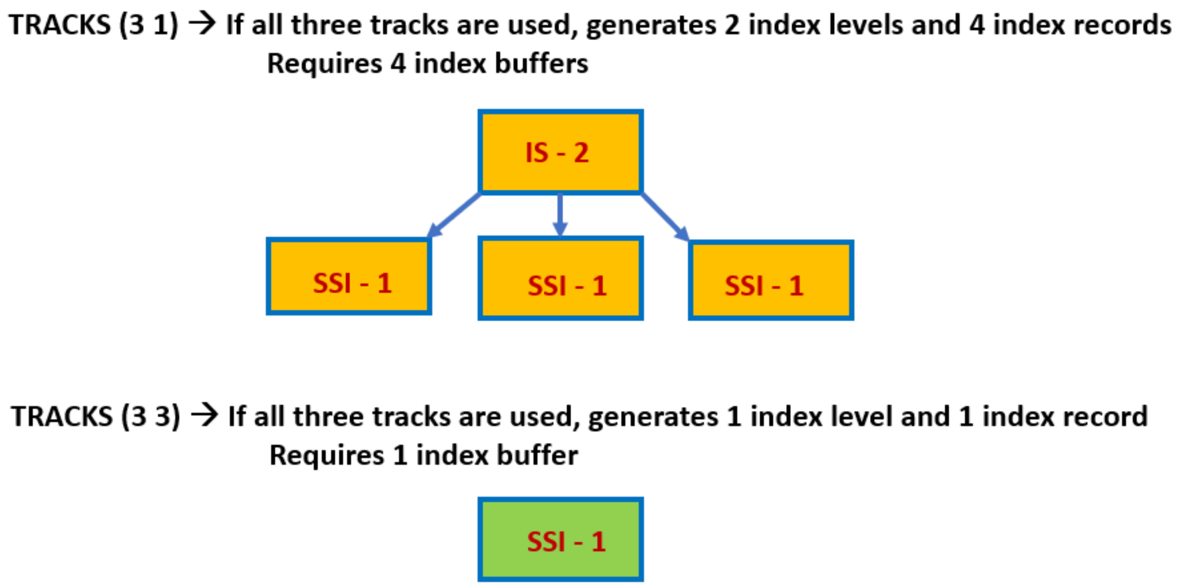


Figure 3: VSAM CA size

the one that is not done. He suggested that the key is to reduce physical I/O operations. One misconception that many people have is that a disk cache hit doesn't generate physical I/O overhead. All the I/O involved is shown in Figure 1.

Figure 2 shows the structure and terms associated with a KSDS file.

Tuning LSR files is simply the opposite of what Robin Hood did. Robin Hood stole from the rich to give to the poor. In LSR you will steal from the poor to give to the rich! In LSR terms, the poor are low to medium activity files, and the rich are most active files.

So, the major contribution made by low/medium activity files in LSR is to provide their resources so that higher activity files can use them.

Gene recommended that users define LSR pools explicitly. They should, initially, bring the system up dynamically to get an idea of the buffers and strings required and the maximum key length (test environment). Then, using the buffers defined, use the definition to define the data component. Initially, use the same definition for the index component. Run transactions and determine the actual buffers used. Using this information, adjust the buffers required for the data and index components. Always define a safety valve buffer of 32K, if none is defined. Then use a performance monitor or CICS statistics (STAT or EOD). Once in production, monitor and adjust as required.

Data buffer tuning is highly dependent on a file's access patterns. Good look-aside hit ratios for the data component usually require a substantial amount of storage to obtain an 80%+ hit ratio. The major cause is that the data component for all the files is usually very large (versus the index component). Good look-aside hit ratios usually result in files with: sequential activity; Read for Update/Rewrite/Delete activity; and concentrated read activity. The LSR buffer look-aside percentage can be misleading. The percent specified does not mean that every file is getting that percentage (remember Robin Hood). The look-aside percentage is the average of all the files using that buffer.

When it comes to string allocation, each pool can have up to 255 strings. They are usually tuned when you get a wait on strings condition. There are 2 types of wait on strings for LSR: wait on string related to the number of strings allocated to the file; and wait on string related to the number of strings allocated to the pool. String allocations are controlled by CICS. The objective should be to have the LSR string assignment somewhere between 50% to 60% of the peak string usage.

The maximum key size is 255 bytes. Because LSR control blocks are shared, the maximum key length must be defined (PLH control block). If the maximum key size specified for the LSR pool is too small, the file will not open. To avoid this situation, many installations define the maximum key size as 255. The actual virtual storage cost depends on the number of strings.

An area that must be monitored is the possibility of a file monopolizing a buffer in a pool. The problem is that CICS does not provide information regarding the number of LSR buffers being used by a file. The statistics provided indicate the activity, but this does not translate into number of buffers. A file could perform 100K accesses to the file, but this does not translate into number of buffers because it could be that the access is to one or a few buffers. There are several options to resolve this situation. You could move the file to a separate LSR pool; or increase the number of buffers to reach the file's point of Diminishing Return. Once you reach this point, other files will have access to buffers.

Hiperspace buffers were designed to use Expanded Storage (ES). ES worked like a very fast synchronous paging device. ES was less expensive than real storage and was 4K addressable (not byte addressable like real storage). z/Architecture does not support ES. In order to maintain this functionality, ES is simulated using real storage by z/OS. CICS supports Hiperspace buffers in LSR. However, you are using real storage to simulate ES. There's a moving real to real overhead. It's better to allocate the equivalent Hiperspace buffers into the regular LSR buffers. You may want to use Hiperspace buffers under the following conditions if enough real storage exists: you need more than 32K buffers of a specific size; or you are running low on the region virtual storage availability.

Fragmentation represents the lost space due to the difference in the CISZ and the LSR buffer assigned to handle the CI. The major cause of fragmentation is that VSAM has 28 different CISZ available while CICS LSR supports only 11 buffer sizes. Some data component fragmentation may be acceptable, such as using an 18K CISZ (non-VSAM/E) to obtain the best disk utilization. In this case, you would use a 20K buffer with a 2K or 10% cost of virtual/real storage fragmentation. Other fragmentation is not acceptable, such as using a 16K buffer to cover a 10K CISZ because the user did not define a 12K buffer. Adjusting the CISZ to a data component buffer size may have some advantages, for example, increasing a CISZ from 5.5K to use an 8K buffer. This takes advantage of 2.5K (31%) lost virtual/real storage. You can add or adjust free space without increasing the amount of virtual or real used by CICS LSR.

Some people think you should use as many pools as possible so files can be segregated to reduce buffer contention and/or interference. Other people think you should define as few pools as possible (preferably 1) so that resources can be used more efficiently. The things to bear in mind are:

- LRU algorithm works best with a larger number of buffers.
- Do you allocate a ‘fudge factor’ to each pool’s definition?
- Are the files continuously used? What happens to the resources when the file is in low activity?
- Unless you are using VSAM threadsafe, access to the different pools are single threaded via the QR TCB.

VSAM CA size (CASZ) is an important tuning option. CASZ is indirectly defined through the primary and secondary allocation. Bad CASZ can occur for any file that is incorrectly defined. However, it is more prone to happen for small files (less than 1 cylinder). Incorrect setting of the CASZ can result in unnecessary index CIs. Unnecessary index CIs would require buffers. The rule you must remember is that all data CAs must be the same size throughout the file (primary and secondary). If requested space is in CYLINDERS, the CASZ will be 1 cylinder. If requested in TRACKS, use the Highest Common Denominator. Be careful when using RECORDS, because it can lead to bad CASZ. Figure 3 illustrates good and bad CA size allocation.

CI/CA splits are the result of adding new records or extending the length of variable length records. Splits result in physical I/O operations. Splits mean that VSAM file can continue to operate accepting the split. However, many I/O operations can result, especially for a CA split. It can be particularly bad if an extent needs to be acquired. Besides the actual number of physical I/O operations required to service the split, there’s the cost of adding a new extent (data and/or index) because of additional I/O activity to the catalog, VVDS, and VTOC. CI/CA splits can create free space that cannot be used or has a very low possibility of being used.

There is a hidden cost to CI/CA splits that is not usually discussed. CI splits take a single CI and convert it to two CIs that are approximately 50% filled. Instead of being able to access the data in one buffer, you now need two buffers (half-full) to access the same amount of data. This is a hidden cost of CI splits and is important for most active files. CA splits come as a result of not having a free CI in the CA, causing the CA split. Splits can also affect the index component requiring additional indexes and, therefore, more LSR buffers.

VSAM allows for a file to have multiple extents. Extent processing has improved over time (Space Constraint Relief –DFSMS). The cost of an extent occurs when obtaining an extent because the process involves accessing and updating the catalog, VVDS, and the VTOC. The data set is serialized during a CI/CA split while the extent is being processed. The data set is serialized for CI/CA splits/reclaims for VSAM, and the data set is serialized for CA splits/reclaims for RLS. This can take time depending on the workload.

As you can see, the presentation was packed with useful information, far more than there is space for in this newsletter.

A copy of Eugene S Hudders’ presentation is available for download from the Virtual CICS user group Web site at fundi.com/virtualcics/presentations/LSRTuningNov19.pdf.

You can see and hear the whole user group meeting at https://youtu.be/midDLKwvPA4.

Sponsors wanted

In order for this user group to continue operating in 2020, we are looking for new sponsors. We are looking for an organization to pay for the administration costs, host the 10 years' worth of resources on this Web site, and also provide access to Webex or similar so that we can hold the virtual user group meetings.

If you would like to find out more about sponsorship, please contact Trevor Eddolls at trevor@itech-ed.com.

Otherwise, the Web site and this user group will cease to exist at the end of this year.

Meeting dates

The following meeting dates have been arranged for the Virtual CICS user group providing we find a sponsor for 2020:

- On 14 January 2020, Ezriel Gross, Principal Solutions Advisor at Rocket Software, will be presenting.

We are using Webex for the user group meetings.

CICS news

IBM has announced new automated unit testing capabilities for CICS COBOL programs using Db2 with IBM Developer for z/OS (IDz) V14.2.1. Users are now able to stub out CICS Db2 calls for COBOL programs through the enhanced record and playback functionality, as well as export results in new formats. The automation of testing is a key component for agile development.

More information can be found at https://developer.ibm.com/mainframe/2019/10/22/running-automated-unit-tests-in-isolation-now-possible-for-cics-db2-programs-with-ibm-developer-for-z-os-v14-2-1/

Recent CICS articles

CICS Bundle Maven plug-in 1.0.0 by Dave Nice on CICS Developer Center (14 November 2019). You can find the article at: https://developer.ibm.com/cics/2019/11/14/cics-bundle-maven-plug-in-1-0-0/

About the Virtual CICS user group

The Virtual CICS user group was established as a way for individuals using IBM's CICS TS systems to exchange information, learn new techniques, and advance their skills with the product.

The Web site at www.fundi.com/virtualcics provides a central point for coordinating periodic meetings (which contain technically-oriented topics presented in a webinar format), and provides articles, discussions, links, and other resources of interest to IBM CICS practitioners. Anyone with an interest in CICS is welcome to join the Virtual CICS user group and share in the knowledge exchange.

To share ideas, and for further information, contact trevor@itech-ed.com.

The Virtual CICS user group is free to its members.