An introduction to tuning VSAM in CICS TS for z/VSE

Mike Poil
michaelalanpoil@gmail.com
Version: March 2022

http://www.ibm.com/zVSE
http://twitter.com/IBMzVSE

Notice Regarding Specialty Engines (e.g., zIIPs, zAAPs and IFLs):

Abstract

The aim is to explain some of the theory behind how VSE/VSAM works under CICS TS for z/VSE and to show how **Local file** performance can be tuned using IBM-provided free tools. *It is not designed to cover everything about the subject!* The products will be referred to as "CICS" and "VSAM" from now on. See the warnings about Function Shipping at the end of the presentation.

VSAM Redirector file access is subject to network, remote database, other delays and any inherent design constraints imposed by z/VSE, and will not be discussed. ISV VSAM redirection software may impact VSAM performance.

We will not cover the issue of tuning VSAM within the CICS applications. Developers need to be aware of the implications of the use of the various EXEC CICS requests, particularly when using KSDS log files where records are added to the end of the file, very long browses or using a series of random reads to find the correct record, inappropriate sequencing of requests for multiple files etc. In other words, what the CICS programs do can defeat any attempt to tune VSAM and can even cause operational problems.

The presentation assumes familiarity with EXEC CICS requests, VSAM dataset concepts, IDCAMS DELETE/DEFINE parameters and CICS CSD resource definitions.

Abstract

This presentation is based on my experience of working on customer performance problems, whether customer-reported problems or paid consultancy work. It also has information based on my own performance evaluation tests to find out how CICS and z/VSE really work rather than rely on what is published about the subject, which is not always accurate.

Some slides show the IBM Internal Use STAT Rexx analyser output, and are designed to show how DFH0STAT output can be used to highlight VSAM Best Practice and VSAM performance issues. An enhanced version of DFH0STAT that contains significantly more VSAM performance data than the version shipped with z/VSE can be found here:

http://www.vmworkshop.org/mikepoil/index.shtml

Please excuse some repetition.

Agenda

- VSAM Tuning for the Impatient
- Introduction.
- Definitions that affect VSAM performance:
  - IDCAMS.
  - CSD Local FILE definition.
  - CSD LSRPOOL definition.
- CICS Statistics data.
- File Tuning Principles
- Tuning LSR and case studies.
- Rexx DFH0STAT analysis tool.
- Tuning NSR files.
- Contention and resource definition limit issues.

Agenda

VSAM splits.

Tuning file contention.

Rexx DFH0STAT analysis tool File profiling.

Defining a local base KSDS as a Shared Data Table.

A free z/VSE I/O Performance Monitor

ISV tuning products.

Gotchas.

Warnings about Function Shipping.

Additional reference material is provided at the end and will not be discussed during the presentation itself.

VSAM Tuning for the Impatient

For those who need quick and simple rules right now. Don't forget to refer to the slide at the end about Function Shipping, which uses many times more CPU time than local VSAM.

Delete/Define

A 16K+ Data CISZ typically improves Browse and CA split performance. A Data CISZ value less than 4K is not normally very efficient for 3390 DASD usage. Specify Data **and** Index CISZ values and use the LSR buffer sizes of 0.5K, 1K, 2K, 4K, or multiples of 4K to 32K if possible. An Index CISZ greater than 8192 is not supported.

Avoid SHR(4) if possible. Consider using e.g., H&W SYSB-II if SHR(4) cannot be avoided.

Avoid compression if possible.

Use FSPC to reduce CI, but especially, CA splits, and don't go crazy with the percentage values. FSPC *doesn't* work when adding records to the end of file.

Avoid creating many secondary allocations because they are costly, and result in potentially fatal NOSPACE condition while CICS is active.

Don't initialise a KSDS with a **high** key record like Hex FFs because more CA splits occur adding records to the end of the file when that is done.

VSAM Tuning for the Impatient

CSD FILE Definition

Connect it to an LSRPOOL unless there is a *very* good technical reason not to. 99% of the time NSR gives *noticeably worse* performance than LSR.

Use DSNSHARING(ALLREQS) on the Base and Path CSD FILE definitions and when multiple FILE definitions use the same dataset. The reduction in CPU time and I/O can be considerable, but not always, and the use can avoid data integrity problems that can occur when updating via the Base and Path.

Defining too small a number of STRINGS for a local file may cause task waits and abends. Use STRINGS(5) as the minimum. When CICS statistics shows that these waits occur, add more STRINGS than the value for the peak number of waiting transactions. Do not use STRINGS(1) in an attempt to avoid VSAM issues like Exclusive Control waits.

A COLD start is normally required to install an updated CSD FILE definition in a production CICS system. That is, unless the customer can afford to close disable the FILE, use CEMT to discard it and CEDA to install the updated definition.

VSAM Tuning for the Impatient

LSRPOOL Definition

Always use LSRPOOL CSD resource definitions, but ignore the case where a COLD start dynamically defines an LSR Pool to open, read and close the CSD. Always allocate DATA **and** INDEX buffers in the LSRPOOL CSD definition. If only DATA buffers are allocated, duplicate the DATA buffer allocations as INDEX and use CICS statistics to tune the results. Don't accidentally exceed the available free GETVIS when increasing the number of buffers. Allocating more than about 2,000 buffers for a given buffer size (e.g. DATA 4096) might result in more CPU usage than is optimum, but it is very difficult to be sure. An Index buffer size greater than 8K is never used.

Consider placing SHR(4) files and their associations in a SHR(4) LSRPOOL to get better control. Consider placing ESDS in their own LSRPOOL.

It can be dangerous to define a small number of buffers when the buffer size is used for a lot of requests. This can cause task waits, abends and VSAM data integrity issues when the numbers are very low compared to what VSAM needs.

A COLD start is required to install updated CSD LSRPOOL definitions. To install an updated LSRPOOL while CICS is active, close all files in it and then CICS will delete the LSR Pool.

Introduction

VSAM tuning is important but needs to be reviewed in the context of the whole environment. VSAM performance impacts CICS, and a poorly tuned or overloaded CICS impacts VSAM.

z/VSE, z/VM and other product performance can impact CICS, and vice versa.

Tuning a 3-stage iterative process:

1. Collect an appropriate amount of performance data.

2. Analyze and look for performance problems that can be tuned.

3. Either tune, that is make changes, and start again at (1), or stop.

Performance analysis should not be based on one day's performance data because a CICS system does not behave uniformly on a daily basis. A week is a good start point. Tuning should preferably be done during known regular and seasonal workload peaks. Also, don't assume that tuning is needed once for the lifetime of the system because things change!

Remember the 80/20 rule and tune what will produce the most savings.

Stop if the tuning objectives have been met. However, if the last change did not achieve the expected results, either the tuning potential has been achieved or look for reasons elsewhere. If it is worse, undo the changes!

Introduction

I am looking at how to generically tune the VSAM workload by:

1. Reducing wait and CPU time from unnecessary EXCPs (I/O requests).

2. Reducing wait time and CPU time due to VSAM contention and resource definition limits imposed by CICS definitions.

Tuning at this level is done by adjusting values in the:

- CICS CSD FILE and LSRPOOL resource definitions.
- IDCAMS definitions.

The size of the VSAM datasets and the application access patterns can have a big effect on what can or can't be done in terms of improving VSAM performance.

Be careful, changing a parameter may impact performance in a positive way in one area, but in a negative way in another. For example, I tuned browse performance for a long-running transaction hog, but that allowed it to impact other transactions running at the same time more than it did before!

How much response time improvement is achieved cannot be quantified from the data that we will look at. It may be necessary to look in detail at unacceptable task response times and determine why there is a problem, which can be a challenge even with ISV tools.

Introduction

Tuning the workload given to VSAM may be required, which will require application changes. Understanding how the applications use files can be extremely important.

VSAM I/O elapsed time depends on the DASD model, features such as PAV, Flash, PPRC and the load, plus the Ficon channel speed. Use z/VSE 6.2 ZHPF support after installing PTF UD54386 for APAR DY47838, which should result in most VSAM I/O elapsed time being faster. I saw up to 25% reduction in I/O elapsed time and one of my customers reported that overnight batch ran an hour faster after enabling ZHPF.

The wait time experienced by the application EXEC CICS request can be impacted by how busy CICS is because of the delay between I/O completion and the task being re-dispatched. Even a CICS system that has very good access to a single CPU can start to provide degraded response times when it exceeds 50% busy for a time.

Important disclaimer:

All of my performance comparisons were done in a non-dedicated machine environment with I/O driver transactions and do not guarantee what will be seen in a production environment - YMMV applies in these cases. Other VSAM performance data provided is similar to what I have seen in customer production systems.

IDCAMS Definition

```
DEFINE CLUSTER                         -
     (NAME(TCOM.CICS2.IYBTZCCA.FILEB)          -
      INDEXED                          -
      FSPC(00 15)                      -
      RECORDSIZE(80 80)                -
      CYL(1500 150)                    -
      COMPRESSED                       -
      SPEED                            -
      VOLUME(VSAM01)                   -
      KEYS(6 1)                        -
      SHAREOPTIONS(2))                 -
    DATA                               -
     (NAME(TCOM.CICS2.IYBTZCCA.FILEB.DATA)     -
      CISZ(4096))                      -
    INDEX                              -
     (NAME(TCOM.CICS2.IYBTZCCA.FILEB.INDEX)    -
      CISZ(2048))                      -
    CATALOG(USERCAT)
```

An example of a KSDS ("INDEXED"). Other VSAM file types have very similar definitions.

The parameters that can have the most impact on performance are shown in red and are discussed on the following slides.

For the space allocation, avoid having secondary allocations while CICS is active if possible as each new allocation will cause a noticeable delay to the whole of CICS.

IDCAMS Definition

Parameter	Affects	Notes
CISZ Data	The number of EXCPs. Contention. DASD utilisation. Data transfer time.	Larger CISZ like 16K or higher can reduce the number of EXCPs for browse and reduce the cost of CA splits but might increase VSAM exclusive control waits when many updates occur. Less than 4K gives poor DASD utilisation. LSR Pool buffer size dependency. Fast channels mean that large CISZ values should be less of an issue.
CISZ Index	See VSE/VSAM reference material.	LSR Pool buffer size dependency. The maximum useable Index buffer size is 8K.
COMPRESSED	CPU time. The number of EXCPs to process the data. DASD utilisation.	Best to avoid, if possible, due to the CPU overhead. More records per CI through compression might reduce the number of EXCPs. Reduces DASD utilisation.

IDCAMS Definition

Parameter	Affects	Notes
FSFC(CI%,CA%)	KSDS and AIX CI and CA split activity. The number of EXCPs. CPU time. A split serializes all other access to the file until it is complete.	FSPC is reserved when the file is loaded or extended sequentially. Might reduce EXCPs when records are inserted before End Of File (EOF) and LISTCAT counts them as inserts. Records added after EOF do not benefit and LISTCAT counts them by just increasing the total number of records. A CI split requires about 4 to 7 EXCPs but has a small serialisation effect in CICS. A CA split can require hundreds of EXCPs and have an elapsed time up to 0.5 seconds or more. Try to avoid both as much as possible. Large FSPC values can waste a lot of DASD space for no gain. Beware of the 4.3GB limit for non-XXL files. LISTCAT will show the correct CI and CA split counts only after the file has been closed in CICS. ISV products and my DFH0STAT show counts for splits since the file was opened or CICS started.

IDCAMS Definition

Parameter	Affects	Notes
SHR(4)	Number of EXCPs. CPU time.	Normally produces the worst possible performance and does not respond to normal tuning techniques. An ISV product like H&W SYSB-II might enable SHR(4) to be avoided. VSAM will use an EXCP to read a Data and Index CI without attempting lookaside unless LSR buffering is used when VSAM caches the top Index CI only. A BROWSE operation is less affected for READNEXT. VSAM uses more CPU per request than normal as it has to manage SHR(4) locks using z/VSE LOCK/UNLOCK macros etc. If SHR(4) is used only to enable a PATH/AIX to be used, investigate the use of the DSNSHARING option in the PATH and the base FILE resource definition entries that allow SHR(2) to be used.

IDCAMS Definition

Parameter	Affects	Notes
SHR(3)	Data integrity.	If SHR(3) is defined for a file it does not affect performance as such, but it presents a data integrity issue because the file can be opened by many partitions in read/write mode without any attempt by VSAM to serialise concurrent write requests that could cause damage. I have never used SHR(3) and I do not know how integrity is managed by customer applications nor what types of problem might occur.

CSD Local FILE Resource Definition

FILE LSRPOOLID=	Buffering	Notes
NONE (LSRPOOL 0 is shown in CICS statistics output)	NSR	NSR performs badly compared to LSR. VSAM resources are used exclusively by the file. STRINGS defines the maximum number of concurrent requests CICS *and* VSAM can handle. When 5 or more are defined, 20% are reserved for simple READ requests. DATABUFFERS and INDEXBUFFERS say how many buffers are allocated in Partition Getvis just for this file. Uses (mostly) 31-bit Partition Getvis storage, but CLOSE/OPEN may cause Getvis fragmentation.
1 to 15	LSR	LSR provides READ caching and is much more efficient than NSR in terms of CPU time and number of EXCPs.

CSD Local FILE Resource Definition

FILE LSRPOOLID=	Buffering	Notes
DSNSHARING	Both	Allows access through the Base and Path to share the same buffers to reduce EXCPs (a single ACB is used for all access). Use it when multiple FILE definitions refer to the same physical dataset. Performance improvements may range from none to very significant, with at least some improvement likely, and should be considered as a Best Practice choice. It can avoid the use of SHR(4) for the Base when updating via the Path and make a significant difference. It may be required to avoid Base and Path data synchronisation errors.
STRINGS	Both	STRINGS at the LSR level defines the maximum number of concurrent requests CICS will allow. When 5 or more are defined, 20% are reserved for simple READ requests and cannot be used for file changes. File string wait (FCPSWAIT) is not handled well by the CICS Dispatcher, it can impact response time and can cause severe errors in extreme cases.

CSD LSRPOOL Resource Definition

Parameter	Notes
STRINGS	At the LSR level determines how many requests VSAM can handle before FCSRSUSP string waits occur. Always avoid LSR string waits.
DATA and INDEX BUFFERS 0.5K, 1K, 2K, 4K and 8K for both Data and Index, and multiples of 4K from 12K to 32K for Data buffers only.	Allocates buffers by CISZ ranges and the type of CI – Data and Index . Allocating separate Index buffers normally provides the best performance and tuning potential. If the DEFINE CISZ does not match, the next larger buffer size is used. DFHSTUP reports on the LSR Pool buffer sizes used by each file, but not the defined CISZ. My version of DFH0STAT reports both. Allocating a small number of buffers can cause errors when updating a file that has an AIX, for example an AEIU abend with VSAM error code X'90' RC X'08", or unusual EXEC CICS RESP values like DUPREC, plus it might cause many task FCBFWAITs. Allocating more than about 2,000 buffers of a given DATA/INDEC type size might start to become expensive in terms of the CPU time used for buffering, but YMMV. Uses (mostly) 31-bit Partition Getvis storage, see the reference material at the end for more details about Getvis.

CICS Statistics Data

I use this for overall CICS tuning, but it is not perfect.

Statistics data is always collected and cannot be stopped. Most counters are "reset" at one or more times per day. SIT STATRCD=OFF avoids the regular "interval" resets, which are every 3 hours by default, but there is always a "midnight" reset, the time of which can be changed from 24:00:00 by a user-written PLTPI program.

See the CICS Performance Guide chapter 5 for full details including what gets reset to what and when. FILE statistic counters are subject to timed resets to zero, but LSRPOOL statistic counters are not subject to timed resets.

Statistics data is sent to DMF before a reset and DFHSTUP can format it. Statistics are "lost" when a file is closed and when an LSR Pool is deleted after the last file is closed, however, USS data is sent to DMF, and the DFHSTUP SUMMARY parameter will combine timed reset and USS data, although some detail is lost in the report.

DFH0STAT (the STAT transaction) uses internal information from within CICS. For tuning VSAM, run it before closing files for batch or "midnight" to minimise data loss. I have a very enhanced version of DFH0STAT that produces much better VSAM data. I will only be showing DFH0STAT output because the output from DFHSTUP is similar, although now inferior.

CICS Statistics Data

It is important to correctly interpret File Statistics.

EXCP counts for a Path are only those that VSAM uses to access the Base Cluster.

AIX EXCPs are not counted in FILE statistics but are counted in LSRPOOL statistics, which is a VSAM and not a CICS restriction. If CISZ values for an AIX are defined so that they do not correspond to other files in the same LSR Pool, they can be identified uniquely.

The same VSAM Base Cluster may appear in multiple CSD FILE definitions and the EXCP counts will represent those made via that filename. The total EXCPs against the dataset will not be seen unless the counts are combined, which can be done by the STAT REXX program.

When using DSNSHARING, the EXCP counts that are provided by VSAM for each Base Cluster and its Path(s) are duplicated but still represent the EXCPs made to the Base Cluster. If DSNSHARING is used when there are multiple CSD definitions for the same Base Cluster, the EXCP counts are also duplicated for each filename and represent the total EXCP activity using all filenames.

DSNSHARING can avoid read integrity issues when using update Paths, and it can reduce EXCP counts through improved buffer sharing between the CICS File definitions. VSAM uses just one ACB for the Base Cluster no matter how many OPENs are performed.

File Tuning Principles

File access by the applications might be inefficient, and should be addressed first where possible. Look for obvious signs such as a high level of access against a file, then ask if it is reasonable. For example, I saw 175M requests to one file in 9 hours and asked applications if this was expected. They said "no", and fixed the bug. It would be pointless using normal tuning techniques with a bug like that. Maybe applications are already aware of problem files - ask them.

Using VSAM NSR is known to cause worse performance than LSR in most cases, so look for NSR usage and convert to LSR where possible.

Increase LSR buffering to reduce EXCPs, but very high buffer counts increase the CPU cost for potentially minimal EXCP savings compared to using fewer buffers.

If EXCPs are not reduced as expected, look at the files to see if you there is an obvious reason. For example, if the file has a lot of Update, Delete or Add activity a lot of EXCPs are used to write data and it is mostly only read EXCPs can be reduced. CA split EXCPs can be reduced by increasing FSPC CA% and/or made to work more efficiently by increasing the Data CI size.

At the end of the day, perhaps you can’t make the file perform any better!

Tuning LSR

Use CSD LSRPOOL resource definitions and define the numbers of Strings, Data and Index buffers in order to make tuning easier. Allowing CICS to dynamically build the LSR Pool results in no Index buffers and adjusting the numbers of strings and buffers requires changes to FILE definition counts!

How many LSR Pools should be used? There is no one "correct" answer, however:

- Consider placing SHR(4) datasets and associations in one SHR(4) LSRPOOL away from other files, but avoid SHR(4) if possible, e.g. use DSNSHARING for the FILE and PATH definitions if that is what resulted in SHR(4) being required in the first place, or consider using H&W SYSB-II to avoid the need for SHR(4).
- Consider using an ESDS-only LSRPOOL.
- Consider placing very active files in separate LSRPOOLs when a very large number of buffers appears to be required. The STAT REXX program LSRMAP option provides idea by file EXCP activity by LSR Pool buffer type and size.
- Consider putting VSAM files whose performance is critical in their own LSRPOOL.

Note: I may use the word "consider" because improvements are not guaranteed and there may be side effects from making a change.

Tuning LSR

The aim is to improve VSAM “Lookaside", which means increasing the number of "read" hits in buffers to reduce CPU and elapsed time for EXCPs, which is another word for improving data caching. By "read" I mean any request that includes a "read" under the covers like an ADD, DELETE and UPDATE.

"lookaside" is reported as a percentage of read hits compared to **all** read requests. A good LSR Pool overall lookaside target is 95% for Index I/O and 80% for Data I/O, although 80% for Data may be an impossible target. Within the LSR Pool, different Data and Index buffer sizes will perform differently to contribute to the total.

Increasing the number of buffers for an underperforming Data buffer size and/or Index buffer size will typically increase lookaside. Avoid using more than about 2,000 buffers for a given buffer size (but not the total for the LSR Pool). The number of buffers that belong to a file of that type of buffer size that are scanned affects the amount of CPU time because VSAM does **not** have the z/OS Buffer Hashing algorithm that avoids the buffer pool size issue, instead it uses a serial scan. As far as I know, it is impossible to determine how many Data and Index buffers are in use for a file, or in a way that could be helpful.

When currently using a large number of buffers to reduce EXCPs, it might be worth reducing the number to see if a similar lookaside is achieved with fewer.

Tuning LSR

The number added could be anywhere between +10% and +100% or more depending on how much improvement of the lookaside is required to match the target percentage and the number currently allocated. Index lookaside is normally much easier to improve than Data because every Index Component is much smaller than the Data Component.

About 5% more 31-bit Getvis storage is required than the sum of each (CI size * added buffers). Reference material at the end shows how to calculate available Getvis storage, but the enhanced DFH0STAT calculates the likely total GETVIS usage based on the +5% estimate.

SHR(4) files might consume additional buffers without any improvement.

A simple test to show the possible effects of an LSR buffering

On the next slide I have provided the results of performing 4 million EXEC CICS READ without UPDATE to a small KSDS with 100 Data and 101 Index CIs using its own LSRPOOL. The baseline eliminated all LSR buffer scanning by reading the same record every time, in which case VSAM was already pointing at the last used CI and there was an immediate cache hit.

Because the file is in its own LSR Pool, the defined number of buffers is how many are scanned every time, which would not be the case when you have multiple files in the same LSR Pool. Therefore, use it as an illustration of the cost of using EXCPs versus lookaside, and the potential impact when more buffers are allocated than can be effectively used. Do not use it to come to the conclusion that you should never allocate more than a certain number of buffers! YMMV.

A simple test to show the possible effects of an LSR buffering

50/50 etc. means 50 Data and 50 Index buffers.

Test	CPU seconds	Elapsed seconds	EXCPs	Lookaside
Baseline	12	12	0	100%
50/50 buffers	307.5	810	4,068,110	49% Data/74% Index
80/90 buffers	110.5	448	1,296,031	79% Data/94% Index
100/100 buffers	17.5	18	0	100%
500/500 buffers	24.5	27	0	100%
1000/1000 buffers	38.5	40	0	100%
4000/4000 buffers	95.5	98	0	100%
CICS Data Table	4	4	0	N/A

The reason that a CICS (Shared) Data Table is so fast is that all VSAM code is bypassed!

Tuning LSR Case Study 1

I realise that the case studies are based on one day's data, but I only have a short time to talk about the subject. However, it does provide an idea about how much time you would need to tune VSAM properly without automation.

```
LSR Pools

  Pool Number :    2       Time Created :    05:17:23.10465

     Maximum key length . . . . . . . . :          255

     Total number of strings  . . . . . :           65

     Peak concurrently active strings :             12

     Total requests waited for string :              0

     Peak requests waited for string. :              0
```

This is where to look for LSR contention issues.

The number of strings defined in the CSD LSRPOOL definition is a good value. Peak active is not close to the total number and there were no string waits. Don't worry about allocating too many strings.

Tuning LSR Case Study 1

```
Buffer Totals

     Data Buffers . . . . . . . . . . . :          288          Index Buffers. . . . . . . . . . :          320

       Successful lookasides . . . . . :   11,504,852             Successful lookasides . . . . . : 29,751,487

       Buffer reads . . . . . . . . . :    3,467,370             Buffer reads . . . . . . . . . :        58,223

       User initiated writes. . . . . :    3,210,888             User initiated writes. . . . . :       209,187

       Non-user initiated writes. . . :            0             Non-user initiated writes. . . :             0
```

This is where to start looking for potential EXCP savings.

Data and Index buffers are defined, which is Best Practice.

Lookaside % = (Successful lookasides*100)/(Successful lookasides + Buffer reads)

Data lookaside (11,504,852 * 100) / (11,504,852 + 3,467,370) = 77%, which is very close to the suggested 80%.

Index lookaside (29,751,487 * 100) / (29,751,487 + 58,223) = 99%, which is more than the suggested 95%.

The enhanced DFH0STAT now shows lookaside % values. Ignore writes.

The LSR Pool is working well overall.

Tuning LSR Case Study 1

Data Buffer Statistics

Size	Buffers	Look Asides	Reads	User Writes	Writes
2048	160	94,344	426,164	56,597	0
4096	128	11,410,508	3,041,206	3,154,291	0

Index Buffer Statistics

Size	Buffers	Look Asides	Reads	User Writes	Writes	
512	32	0	0	0	0	delete since it is not used
2048	128	14,593,314	484	0	0	
4096	128	12,628,859	55,989	203,712	0	
8192	24	2,529,314	1,750	5,475	0	
16384	8	0	0	0	0	

This is where to look to see which buffer sizes to tune. Tuning the 4K Data buffers is likely to save the most EXCPs - try 256 or more buffers. There is less potential saving from tuning the 2K Data buffers and a significant increase the number of buffers might be required to see any improvement.

Tuning LSR Case Study 2

```
LSR Pools

 Pool Number :    3      Time Created :    00:05:15.09522

     Maximum key length . . . . . . . :          100

     Total number of strings  . . . . :           90

     Peak concurrently active strings :           90

     Total requests waited for string :       42,987

     Peak requests waited for string. :           21   <== the size of the biggest queue
```

What is seen here is normally the result of the number of strings allocated being too small.

The minimum number of strings needs to become 90 + 21 + more, but why not use the maximum VSAM-allowed value of 255? There may be a performance problem that is stopping strings being released fast enough, but how to identify that fact could be very difficult. Had there been a very small number of strings allocated, this level of LSR Pool string wait could have been disastrous and caused all types of potentially nasty VSAM problems.

If 255 strings is still not enough, create a second CSD LSRPOOL definition and split the files between them.

Tuning LSR Case Study 2

```
Buffer Totals

     Data Buffers . . . . . . . . . . . :            435          Index Buffers. . . . . . . . . . :          0

       Successful lookasides . . . . . :      44,689,132             Successful lookasides . . . . . :          0

       Buffer reads . . . . . . . . . . :      56,267,410             Buffer reads . . . . . . . . . :          0

       User initiated writes. . . . . . :       3,756,970             User initiated writes. . . . . :          0

       Non-user initiated writes. . . . :               0             Non-user initiated writes. . . :          0
```

Index buffers are not defined because CICS dynamically defined this LSR Pool.

Lookaside % = (Successful lookasides*100)/(Successful lookasides + Buffer reads)

Combined Data and Index = 44%, which is very bad.

Define an LSRPOOL in the CSD with the existing buffer numbers shown by statistics output as both Data **and** Index. It will use twice the amount of GETVIS storage, but this can be tuned later. To help with this case, the DFH0STAT analyser has an LSRMODEL option to show the Data and Index buffer sizes used by open VSAM files.

Ensure that every file in the LSR Pool that can be used has been opened so that all possible buffer sizes are used, then delete buffer sizes that are not used.

Tuning LSR Case Study 2

```
Data and Index Buffer Statistics

                      Look                   User
    Size  Buffers    Asides         Reads       Writes         Writes
   ___________________________________________________________________
    2048     110       451,493    7,401,268    1,453,394              0       6% lookaside
    4096     200    27,530,104   45,221,982      552,349              0       38% lookaside
    8192     110    16,701,475    3,642,161    1,750,503              0       82% lookaside
   16384      15         6,060        1,999          724              0       75% lookaside
```

The 4K buffers are the main problem as they require 45M EXCPs to read data.

The 2K buffers are a problem, but not as much as the 4K.

The 8K and 16K buffers are performing reasonably well.

Lookaside values will change when Index buffers are allocated. Only then tune the number of buffers.

Rexx DFH0STAT Analysis Tool

I developed a REXX program called STAT where the same code can run on z/VM and z/VSE, and which will analyse both standard and enhanced DFH0STAT output.

It is not perfect and it has to work with imperfect data. With the enhanced DFH0STAT output it is able to find a significantly higher number of issues and provides features like the LSRMAP option that shows LSR Pool usage in terms of the Data and Index buffer sizes and which open files use them.

To show what it is possible to do with DFH0STAT output, I have included its analysis of Case Study 2's data for LSR Pools 1 to 4, not just 3 and other examples may be included. The messages were produced on an old version of both DFH0STAT output and STAT REXX program and the latest versions will often produce better output.

Rexx DFH0STAT Analysis Tool

"Best Practice" messages show that something has been configured in a way that is not considered to be optimum. Some of the issues have less effect than others. For example, allocating buffers that are not used is a minor offence and has no impact other than to waste some storage, but not having Index buffers could have a noticeable impact.

```
Best Practice LSRPOOL  1 does not have index buffers allocated, LSR performance may be degraded

Best Practice LSRPOOL  2 does not have index buffers allocated, LSR performance may be degraded

Best Practice LSRPOOL  3 does not have index buffers allocated, LSR performance may be degraded

Best Practice LSRPOOL  4 data  CI size 01024 has no I/O activity, consider setting zero buffers

Best Practice LSRPOOL  4 does not have index buffers allocated, LSR performance may be degraded
```

Rexx DFH0STAT Analysis Tool

"Limit" messages show a configuration limit and that is likely to result in avoidable waits. How much will depend on how many requests are affected and a small number is likely to have a negligible effect. The output below is based on case study 2.

Increasing the reported "limit" to the suggested value may be the solution. However, this may not always make a significant difference because the limit may have been reached due to other problems slowing CICS or VSAM down.

```
Limit LSRPOOL  1 peak string usage 100%, consider increasing number of strings    80  (max. strings 255)

Limit LSRPOOL  3 peak concurrent string waits         21, total string waits      42987, consider increasing LSRPOOL strings to 116

Limit LSRPOOL  3 peak string usage 100%, consider increasing number of strings    90  (max. strings 255)
```

LSRPOOL 1 has reached the string limit but has not caused string waits this time as there is no string wait message. Consider increasing LSRPOOL 1 STRINGs by a small number.

The LSRPOOL 3 STRINGS value has resulted in a very large number of string waits and needs action to be taken.

Rexx DFH0STAT Analysis Tool

"Threshold" messages show that something is probably not working in the optimum way, or is warning of something that is close to becoming a "limit".

The output is based on case study 2 and shows that LSR Pool buffering is not optimum. These LSR Pools need to have Index buffers allocated first, then check again.

It shows lookaside ratios for the LSR Pool and buffers, and the potential EXCP savings help to determine the relative importance of tuning. LSRPOOL 3 and its 4K buffers are the obviously the worst performers at this point in time. Any text in blue is a note added by me.

```
Threshold  LSRPOOL  1 data  hit ratio  70% is < recommended 80%, Read EXCPs   52703156

Threshold  LSRPOOL  1 data  CI size 01024 hit ratio  60% EXCP saving at 80%     310801, consider increasing number of data  buffers   35

Threshold  LSRPOOL  1 data  CI size 02048 hit ratio  67% EXCP saving at 80%    5418263, consider increasing number of data  buffers  250

Threshold  LSRPOOL  1 data  CI size 04096 hit ratio  74% EXCP saving at 80%    5212766, consider increasing number of data  buffers  300

Threshold  LSRPOOL  1 data  CI size 08192 hit ratio  65% EXCP saving at 80%    6701366, consider increasing number of data  buffers   55

Threshold  LSRPOOL  1 data  EXCP saving at 80%   17643196  (about 33% of the LSRPOOL total EXCPs)
```

Rexx DFH0STAT Analysis Tool

```
Threshold  LSRPOOL  2 data  hit ratio  53% is < recommended 80%, Read EXCPs   14397267
Threshold  LSRPOOL  2 data  CI size 01024 hit ratio   2% EXCP saving at 80%    1667732, consider increasing number of data  buffers    70
Threshold  LSRPOOL  2 data  CI size 02048 hit ratio  69% EXCP saving at 80%     258956, consider increasing number of data  buffers   130
Threshold  LSRPOOL  2 data  CI size 04096 hit ratio  56% EXCP saving at 80%    5001964, consider increasing number of data  buffers   210
Threshold  LSRPOOL  2 data  CI size 08192 hit ratio  58% EXCP saving at 80%    1288335, consider increasing number of data  buffers    55
Threshold  LSRPOOL  2 data  EXCP saving at 80%    8216987  (about 14% of the LSRPOOL total EXCPs)

Threshold  LSRPOOL  3 data  hit ratio  44% is < recommended 80%, Read EXCPs   56267410
Threshold  LSRPOOL  3 data  CI size 02048 hit ratio   6% EXCP saving at 80%    5830716, consider increasing number of data  buffers   110
Threshold  LSRPOOL  3 data  CI size 04096 hit ratio  38% EXCP saving at 80%   30671565, consider increasing number of data  buffers   200
(The 16K buffer size was ignored due to it's usage being below an analysis trigger even though the lookaside is 75%.)
Threshold  LSRPOOL  3 data  EXCP saving at 80%   36502281    (about 65% of the LSRPOOL total EXCPs)
```

Rexx DFH0STAT Analysis Tool

This LSR Pool has Data and Index buffers defined.

```
Threshold LSRPOOL  1 data  CI size 01024 hit ratio   0% EXCP saving at 80%    270608, consider increasing number of data  buffers    50
Threshold LSRPOOL  1 data  CI size 02048 hit ratio   0% EXCP saving at 80%    239028, consider increasing number of data  buffers    50
Threshold LSRPOOL  1 data  CI size 08192 hit ratio  29% EXCP saving at 80%   7586403, consider increasing number of data  buffers   100
Threshold LSRPOOL  1 data  CI size 16384 hit ratio  17% EXCP saving at 80%   2768599, consider increasing number of data  buffers   100
Threshold LSRPOOL  1 data  EXCP saving at 80%   10864638

Threshold LSRPOOL  1 index hit ratio  58% is < recommended 95%, Read EXCPs  19748769
Threshold LSRPOOL  1 index CI size 01024 hit ratio  61% EXCP saving at 95%   2087616, consider increasing number of index buffers    40
Threshold LSRPOOL  1 index CI size 02048 hit ratio  59% EXCP saving at 95%  13211002, consider increasing number of index buffers   100
Threshold LSRPOOL  1 index CI size 04096 hit ratio  51% EXCP saving at 95%   1373873, consider increasing number of index buffers   100
Threshold LSRPOOL  1 index CI size 08192 hit ratio  36% EXCP saving at 95%    716877, consider increasing number of index buffers   100
Threshold LSRPOOL  1 index EXCP saving at 95%   17389368
```

Data lookaside is good overall because there was no threshold message. The 4K activity is not reported because there were 215M lookasides and only 19M EXCPs. Increase the 8K and possibly the 16K buffers to save some more EXCPs. The Index performance is very bad, try doubling the 2K and the 1K buffers.

Tuning NSR Files

Reported by DFH0STAT as LSR Pool 0 FILEs.

NSR files will only provide some lookaside, and adding buffers in the FILE definition may result in fewer EXCPs. The lookaside seems to be more costly in CPU time than LSR.

This example shows no lookaside as every read requires an EXCP to the Index and to the Data because the FILE is SHR(4) with only 1 Index level. This is normal SHR(4) behaviour and cannot be tuned.

```
         Access         LSR  Str Waits   Read     Get Update   Browse      Add      Update     Delete      Data       Index
Filename Method  Type   Pool Max Total  Requests   Requests   Requests  Requests  Requests   Requests     EXCPs      EXCPs
________________________________________________________________________________________________________________________________
VSMSHR4   VSAM   KSDS    0     0    0    156517          0          0          0         0          0    156517     156517
```

The best way to tune NSR files is to use LSR.

The DFH0STAT analyser shows:

```
Best Practice FILE VSMSHR4  is using NSR, GETVIS usage may cause a problem and performance may be poor compared to LSR
```

Tuning NSR Files

VFI0005 has very poor Index buffering, which is shown in its Index/Data EXCP ratio (3975192/849812). Using LSR should help.

VFI0009 has a very high EXCPs/Request ratio (1875284/246443) due to NSR buffering and split activity. Using LSR should help to reduce EXCPs.

Filename	Access Method	Type	LSR Pool	Str Max	Waits Total	Read Requests	Get Update Requests	Browse Requests	Add Requests	Update Requests	Delete Requests	Data EXCPs	Index EXCPs
VFI0005	VSAM	KSDS	0	0	0	254,913	214,777	8,940,197	76,754	148,154	149	849,812	3,975,192
VFI0009	VSAM	KSDS	0	0	0	17,616	114,721	0	105,354	55	8,697	1,091,308	783,976

The DFH0STAT analyser shows:

Profile EXCPs 4825004 NSR KSDS VFI0005 Index/Data EXCP ratio 4.68 EXCPs/Request 0.50 File requests 9634944 Read/Write ratio 41.81
Read 4.87% Browse 92.79% Update 1.54% Add 0.80% Delete 0.00%

Profile EXCPs 1875284 NSR KSDS VFI0009 Index/Data EXCP ratio 0.72 EXCPs/Request 7.61 File requests 246443 Read/Write ratio 1.16
Read 53.70% Browse 0.00% Update 0.02% Add 42.75% Delete 3.53%

A simple test to demonstrate the possible effects of NSR buffering

A repeat of the previous 4M random read test. I have included some LSR buffering (only one file in the LSR Pool) as a comparison. There are 2 levels in the Index.

Do not expect 0 Data EXCPs from NSR. I found that the number of Index EXCPs could be reduced by using 1,000 buffers, but the results varied a lot between runs and I have not included them - another reason to avoid NSR.

If nothing else, it shows the relative cost of a large number of EXCPs in terms of CPU and elapsed time even at only 0.25 milliseconds per I/O.

The results will be exaggerated compared to normal operation.

LSR Test	CPU seconds	Elapsed seconds	EXCPs	lookaside
50/50 buffers	307.5	810	4,068,110	49% Data 74% Index
100/100 buffers	17.5	18	0	100%

NSR Test	CPU seconds	Elapsed seconds	Data EXCPs	Index EXCPs
50/50 buffers	375	1735	2,160,000	3,648,000
100/100 buffers	225.5	812	0	3,245,999

Tuning NSR by using LSR

A customer NSR KSDS was by far the busiest file in CICS, with up to 180 EXCPS per second on average over the CICS business day. Access was predominantly Read for Update and Rewrite, but with Read requests.

The KSDS was migrated to its own LSRPOOL with 1,000 Data and 200 Index buffers.

Before: 8.5M EXCPs, with 2M EXCPs from Rewrites, leaving 6.5M Read EXCPs split almost 50/50 between the Data and Index components.

After: 2.2M EXCPS with 2M EXCPs from Rewrites, saving 6.3M Read EXCPs and a lot of elapsed time plus CPU time from each EXCP. The file then averaged about 45 EXCPs per second, most of which are Write EXCPs that can't be reduced.

Contention and Resource Definition Limit Issues

CICS, and not VSAM, enforces the number of concurrent I/O requests for a file according to the FILE definition STRINGS value and the task will encounter an FCPSWAIT if the request would cause STRINGS to be exceeded. The wait will be resolved later when a string is released by the completion of another I/O request, but it is slow to resolved by CICS.

If STRINGS is 5 or higher, 80% of the number can be used for any type of request, but when that 80% are used, the remaining 20% can only be used by READ (no update) requests a file update request will encounter an FCPSWAIT. The enhanced DFH0STAT code shows both values.

CICS needs to handle Recoverable File record changes, which must either be all committed by a Syncpoint or End Of Task or backed out after an abend and means that other task VSAM requests for the same record(s) will end up in an KC_ENQ wait until a DEQ is performed. This will impact response time, and in some cases, quite badly and can even cause CICS hangs. This is not a CICS defect as such, it is a function of how records are accessed by the customer applications and may result in an AFCG deadlock abend. Some changes to non-recoverable files may result in short-term record locks which do not need to wait until Syncpoint of End of Task to be released. These events are not counted, but ENQ wait time is available in CICS Monitoring data.

Contention and Resource Definition Limit Issues

A task enters FCIOWAIT while any I/O, including split I/O, is running and CICS can dispatch other tasks until it sees that I/O is complete, when the task can be re-dispatched. Reference material at the end of the presentation contains a simplified description of the process. But other wait states can occur, e.g.

• FCCIWAIT - access to the **FILE** is blocked by VSAM until the (CA and) CI split for another task's WRITE request is complete.

• FCXCWAIT - access to the **CI** is blocked by VSAM until other VSAM activity completes, and the blocked request could any type of request, even a READ.

CICS does not directly report the number of FCCIWAIT and FCXCWAIT waits.

• FCSRSUSP - **LSR Pool** access is blocked by VSAM until an LSR Pool string becomes available.

• FCPSWAIT - **FILE** access is blocked by CICS until a FILE string becomes available.

CICS shows FCSRSUSP waits in the LSR Pools report and FCPSWAIT waits in the Files report but does not say how long the delays were. See the CICS Problem Determination Guide chapter 6 for full details.

Contention and Resource Definition Limit Issues

CICS forms a queue of tasks waiting on FCXCWAIT. When the issue is resolved, **all** of the tasks in the queue are resumed to retry the VSAM request, and the highest priority task will be dispatched first, leaving the others dispatchable. As the maximum queue depth grows, wait times become less linear and there is more chance of a retry failure for a task. For example, tasks A and B are resumed when an FCXCWAIT is resolved. Task A's update becomes an FCIOWAIT. Task B may be dispatched to retry its request before task A frees the CI. Task B goes into another FCXCWAIT having used CPU time in CICS and VSAM to achieve nothing. This is not a bug. Now imagine 10 tasks after B who will have been resumed and will retry the request only to go into FCXCWAIT again!

CICS will perform 2 VSAM requests before it enters the first FCXCWAIT in a series for the same request. This is not a bug, the second one is used to pass a ECB address to VSAM for it to POST when the FCXCWAIT is resolved.

General or task performance problems that slow things down and/or CICS getting busy may result in these waits occurring more frequently and for longer. This effect is seen in a later slide. CICS Task Performance Data and possibly TMON will show that FCAMCT, the number of VSAM requests, is bigger than the number of requests when FCXCWAIT occurs.

My VM Workshop page has an unofficial, but working, fix to count FCXCWAIT by file and the enhanced DFH0STAT will show the count.

Contention and Resource Definition Limit Issues

An output-only ESDS will receive FCXCWAITs because the writes go to the same CI.

A KSDS file that only has records added to the end can suffer much FCXCWAIT and FCCIWAIT time from records not being added in ascending key order, even if the application was supposed to avoid it. Do **not** use a dummy high-key record to load the file, even if it is deleted, as this will tend to promote higher split activity. Try:

- EXEC CICS ENQ the file
- Build the key
- EXEC CICS WRITE
- EXEC CICS DEQ the file

FCXCWAITs from updates may be reduced by using a smaller Data CISZ in order to reduce the potential for concurrent updates to the same CI.

Using STRINGS(1) to avoid the FCXCWAITs is not a good idea in z/VSE because the FCPSWAIT waits typically result in longer response times, even though less CPU is used.

CICS Task Performance Data records report FCIOWTT, which is only FCIOWAIT and FCCIWAIT. This value is reported by ASG TMON as I/O wait.

Contention and Resource Definition Limit Issues

An example of the impact on VSAM from running a notoriously bad "batch" transaction on the same day with the same input "File X", based on about 2 seconds of internal trace.

```
 Run1:     File X    READNEXT 6736 FCIOWAIT 1371   FCIOWAIT/READNEXT  0.25
                     FCIOWAIT 0.000589  CICS busy  82.13%
           All FCIOWAIT        1.5194239217  7.40% of total wait, Count 2152 Avg. 0.000706
           File Y FCIOWAIT     0.1850103277  0.90% of total wait, Count  162 Avg. 0.001142
           File Y FCXCWAIT     0.0186860698  0.09% of total wait, Count   12 Avg. 0.001557
           File Y FCCIWAIT     0.0010488125  0.01% of total wait, Count    1 Avg. 0.001049

Run 2:     File X    READNEXT 6911 FCIOWAIT 1135   FCIOWAIT/READNEXT  0.16
                     FCIOWAIT 0.000553  CICS busy   96.05%
           All FCIOWAIT        4.9590373068  9.85% of all wait,  Count  2377 Avg. 0.002086
           File Y FCIOWAIT     1.4118249483  2.81% of total wait, Count  295 Avg. 0.004786
           File Y FCXCWAIT    10.0815792752 20.03% of total wait, Count  371 Avg. 0.007353
           File Y FCCIWAIT     1.1993173926  2.38% of total wait, Count  101 Avg. 0.011874
```

Contention and Resource Definition Limit Issues

For run 2, the transaction's I/O performance was slightly better due to improved buffering, but CICS was running close to 100% while trying to run more work at the same time.

The transaction affected other transactions running at the time more than it did before, causing average FCIOWAIT time seen by the transactions to increase by nearly 3x.

The use of "File Y", which is a KSDS log file where new records are added close to or after EOF, was significantly impacted. In run 2, there were about 200 EXEC CICS WRITEs executed (look at the total wait for 200 WRITEs!!), which compares with about 170 for run 1.

CICS Statistics over even a short time would probably not have highlighted any problem, even though the customer knew that this particular transaction always had an impact, sometimes much worse that others.

Sometimes there is a need to dig deeper to look at performance. Don't expect an ISV CICS Performance Monitor to always explain why response is bad, it didn't help me to get a definitive answer here, and I had to resort to using edited EI=1 and DS=1 internal trace data taken from an AR DUMP command.

Don't use CICS Auxtrace to look at a performance problem because it will cause a performance problem just by using it!

Contention and Resource Definition Limit Issues

DFHSTUP reports on VSAM buffer waits (FCBFWAIT) by file name in the "LSRPOOL FILES" section, but the standard DFH0STAT does not have access to the same data.

Ensure that a small number of buffers is not defined.

The Enhanced version of DFH0STAT contains extra data for open VSAM files, and provides FCBFWAIT counts (shown in red):

```
          Access         LSR  Str Waits     Read    Get Update   Browse       Add     Update     Delete        Data        Index

 Filename Method    Type  Pool Max Total  Requests   Requests   Requests   Requests   Requests   Requests      EXCPs        EXCPs
 ________________________________________________________________________________________________________________________________

 xxxxxxx   VSAM    KSDS    3     0    0     988711        371   13752695     129777          0       1124    1061286      214164

 xxxxxxx  BUFFER WAIT HWM       0 TOTAL               0 LSRPOOL BUFFER SIZE DATA 16384 INDEX  4096

 xxxxxxx  SPLITS CI       40,468 CA             537 DEFINED CISZ        DATA 16384 INDEX  4096 EXCC            0 SHAREOPTIONS 2
```

"EXCC" is the count of Exclusive Control waits and needs the fix to capture the value.

VSAM Splits

Adding records to an indexed file will often cause splits.

A CI split in itself can normally be considered as a minor overhead, but a CA split is a major overhead, and the impact is inversely proportional to the Data CISZ.

For example, using Data CISZ(2048) may require 800 EXCPs and take more than 0.5 of a second to perform, during which time no other I/O is permitted on the file! The handling of the 800 I/O events also requires a lot of time in the CICS EXCPAD code. Consider using a large CISZ, e.g. 16K (or bigger) to reduce the overhead.

A larger Data CISZ may cause FCCXWAIT from update activity, but that may be a lesser evil than the impact of the CA splits.

I have seen a CA split cause SOS Below at times in a CICS system that was constrained by 24-bit DSALIM and was having problems handling the concurrent workload.

CA split activity is seen in LISTCAT after the file is closed, but ISV CICS performance monitors can show it while CICS has the file open as can the enhanced DFH0STAT.

VSAM Splits

STAT analysis for a file that has a large number of CA splits. 2K+ per day was not uncommon.

"Before" is with a Data CISZ(2048) Index CISZ(8192) and "After" is Data CISZ(16384) Index CISZ(4096). EXCPs/request compared to a day that had a very similar file access profile were reduced to about one third of what it was before.

Before:

```
LSRPOOL  4 KSDS  LOGFILY EXCPs    2769037 EXCPs/second    32.79 Index/Data EXCP ratio  0.15 EXCPs/Request  4.44
```

After:

```
LSRPOOL  4 KSDS  LOGFILY EXCPs    1275605 EXCPs/second    15.02 Index/Data EXCP ratio  0.09 EXCPs/Request  1.49
```

Tuning FILE Contention

The data that is available is not good for proper statistical analysis. It suggests that there could be a problem and that changing FILE definition values may help.

Filename	Access Method	LSR Type	Str Pool	Waits Max	Total	Read Requests	Get Update Requests	Browse Requests	Add Requests	Update Requests	Delete Requests	Data EXCPs	Index EXCPs
MSTR001	VSAM	KSDS	1	79	>999	211234	213416	0	5445	213416	0	304412	59993
MSTR002	VSAM	KSDS	1	155	>999	8	278	864650	27107	277	3	213887	62001

The files had *more than* 999 FCPSWAIT string waits, probably due to the number of STRINGS in the FILE definition being too small. At one point in time, 79 and 155 tasks were queued in FCPSWAIT, which *is* a major concern. The DFH0STAT analyzer reports:

```
Limit         FILE MSTR001 peak concurrent string waits        79, total string waits       999+ consider
increasing the FILE definition STRINGS value

Limit         FILE MSTR001 peak string waits is high, please check the file and overall VSAM performance

Limit         FILE MSTR002 peak concurrent string waits       155, total string waits       999+ consider
increasing the FILE definition STRINGS value

Limit         FILE MSTR002 peak string waits is high, please check the file and overall VSAM performance

Threshold     MONITOR exception records were produced, 97289 SOS/DFHTEMP/VSAM resource limit waits occurred
(if CICS exception recording is active, this message shows the extent of the problem)
```

Rexx DFH0STAT Analysis Tool File Profiling

Local and Remote File profiling is performed when using the FILE option. The Enhanced version of DFH0STAT differentiates between Remote files ("REMOTE") and Remote Shared Data Table ("REMTBL") files.

It profiles files other than ESDS that exceed an EXCP threshold. As seen previously, this may help to explain why performance is not as expected.

```
Profile   EXCPs    4587897 LSRPOOL  3 KSDS MAS0002 Index/Data EXCP ratio  0.92 EXCPs/Request 1.92
File requests    2388259 Read/Write ratio       9.99 Read    90.90% Browse    0.00% Update    9.10% Add
0.00% Delete    0.00%
```

FILE MAS0002 has a high EXCPs/Request ratio even with a high random Read %. High Index EXCP counts could be due to poor LSRPOOL 3 buffering that we saw earlier. Check the difference after tuning LSR.

```
Profile   EXCPs   26869764 LSRPOOL  3 KSDS TAB0001 Index/Data EXCP ratio  0.93 EXCPs/Request 1.14
File requests   23583307 Read/Write ratio    6105.50 Read    54.43% Browse   45.55% Update    0.00% Add
0.01% Delete    0.01%
```

FILE TAB0001 (27M EXCPs!) is similar, but has a lot of browse requests that could cause Data and Index EXCPs and may not be so easy to tune.

Rexx DFH0STAT Analysis Tool File Profiling

```
Profile   EXCPs   13110280 LSRPOOL  3 RRDS MASTRR1  Index/Data EXCP ratio   N/A EXCPs/Request  1.00
File requests    13110280 Read/Write ratio 9999999.99 Read  100.00% Browse    0.00% Update    0.00% Add
0.00% Delete    0.00%
```

FILE MASTRR1 has a 1:1 EXCPs/Request ratio even with a high Read %. This RRDS appears to have no lookaside whatsoever. Is it too large to benefit from LSR, are there too few buffers in LSR Pool 3, or what? It may be an LSR "hog"!

```
Profile   EXCPs   12013393 LSRPOOL  2 KSDS J01F002 Index/Data EXCP ratio  1.00 EXCPs/Request  2.00
File requests     6007728 Read/Write ratio 9999999.99 Read     0.00% Browse  100.00% Update    0.00% Add
0.00% Delete    0.00%
```

This file is always browsed and shows no obvious lookaside with every request reading one Index and Data record. Browse uses the Sequence Set, which is the biggest part of the Index Component, so just a small increase in LSR Pool 2 Index buffers may not help.

```
Profile   EXCPs 4759295 LSRPOOL  3 KSDS LOGJNX0 Index/Data EXCP ratio  0.15 EXCPs/Request  6.25
File requests    761911 Read/Write ratio        0.16 Read  14.02% Browse    0.02% Update   12.19% Add
73.78% Delete    0.00%
```

This "log" file has a high percentage of Adds and a very high EXCPs/Request. LISTCAT confirms that there are a lot of CA splits.

Rexx DFH0STAT Analysis Tool File Profiling

```
Profile       EXCPs     333823 LSRPOOL  1 KSDS ATM0001 Index/Data EXCP ratio  0.01 EXCPs/Request  0.08 File
requests    4308182 Read/Write ratio   59834.86 Read     0.00% Browse  100.00% Update     0.00% Add     0.00%
Delete     0.00%

Profile       EXCPs    4192010 LSRPOOL  4 KSDS SUBMIT1 Index/Data EXCP ratio  0.24 EXCPs/Request  0.05 File
requests   81588207 Read/Write ratio     186.38 Read     0.00% Browse   99.46% Update     0.00% Add     0.53%
Delete     0.00%
```

As a contrast, these files are working well even for the high percentage of browse activity.

Reminder: The number of EXCPs reported for a PATH in the FILE statistics does not include the EXCPs that are used internally by VSAM to access the AIX, the EXCPs are those that VSAM uses to access the Base Cluster mapped by the PATH. But, EXCP counts for an AIX accessed by a PATH are included in the LSRPOOL statistics.

Defining a Local KSDS as a Shared Data Table (SDT)

A "small" base KSDS, and not a PATH, that is predominantly (at least 50%?) accessed by READ without UPDATE and by BROWSE may benefit from being defined as an SDT even if not shared with another CICS system. SDT files have their records cached in a z/VSE Data Space with a limit of 2GB per Partition for all SDTs (not 2GB per z/VSE system), but can require a significant amount of Partition 31-bit GETVIS to index the SDT.

A CICS-Maintained Table (CMT) works like any KSDS and it must use an LSR Pool. All changes to the SDT are made to the source KSDS. A User-Maintained Table (UMT) is slightly restricted in terms of the API and no changes are made to the source KSDS.

There is a one-time cost for loading the records when the file is opened - always specify OPENTIME(STARTUP). Subsequently, CICS uses significantly less CPU time to access the SDT compared to accessing the data in an LSR Pool. Any file change uses more CPU time with a CMT because CICS has to update its cache and call VSAM to write the changes to the KSDS using normal LSR access. Data Table misses will also access the dataset using normal LSR processing. EXEC CICS request counts in the FILE statistics for this CICS system will normally just be from the initial load.

For full details, see the CICS TS for VSE/ESA Shared Tables Guide.

An SDT results in huge CPU time savings for REMOTE files and a noticeable amount even for Local access.

A free z/VSE I/O Performance Monitor

z/VSE has a free I/O monitor - the "SIR SMF" console (AR) command. See the z/VSE publication "Hints and Tips from L2" for SIR SMF and other undocumented commands. An ISV monitor might provide similar data.

This data from a Hursley DS8000 shows how fast I/O might be even without 100% DASD cache hits (there is disconnect time). SIR SMF=ON to start monitoring and SIR SMF=OFF to stop it and reset the counters. SSCH is the instruction that starts I/O.

```
sir smf=32d
AR 0015 TIMING VALUES FOR 32D BASED ON    2053754 I/O INSTRUCTIONS
AR 0015   QUEUED      PENDING     CONNECT     DISCONN     DEV.BUSY     TOTAL
AR 0015  msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH
AR 0015      0.001       0.004       0.128       0.108       0.000       0.242

sir smf,vse,32d   (what the z/VSE Supervisor sees as the DASD Service time)
AR 0015 TIMING VALUES FOR 32D BASED ON    2053677 I/O INSTRUCTIONS
AR 0015 MAXIMUM I/O QUEUE   8
AR 0015   QUEUED      PENDING     CONNECT     DISCONN     DEV.BUSY     TOTAL
AR 0015  msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH
AR 0015      0.001       0.000       0.354       0.000       0.000       0.355
```

A free z/VSE I/O Performance Monitor

SYSDEF SYSTEM,ZHPF=START is used to start the z/VSE 6.2 ZHPF support, and SYSDEF SYSTEM,ZHPF=STOP to stop it and reset the counters. The console command SYSDEF SYSTEM,ZHFP=RESETCNT will reset the counters while it is active.

SIR SMF,ZHPF shows how well z/VSE is converting the normal I/O requests to ZHPF format, and is based on every type of I/O request to each candidate volume. ZHPF will not convert sequential file I/O requests and possibly other types of request.

In this case it is very effective! (329 is the only candidate z/VM full pack minidisk on our Hursley z/VSE 6.2 system.)

Rejected ZHPF format I/O requests will cause the original unconverted I/O request to retried by z/VSE. Report non-zero rejected counts to VSE Support.

```
sir smf,zhpf

AR 0015 ZHPF I/O REQUEST STATISTICS

AR 0015   CUU         OVERALL      ZHPF   REJECTED

AR 0015   329         2735962   2735906          0

AR 0015 1I40I  READY
```

ISV Tuning Products

The information here has been supplied by those ISVs who responded to my request for details about how their product might help with tuning VSAM under CICS.

The presentation will not explain how to interpret the data values in the following slides.

ISV Tuning Products

C\TREK from the C\TREK corporation in the USA is available from various business partners, and is a product that some customers use instead of, or in a complimentary fashion with other ISV CICS tuning products to tune CICS systems and VSAM files.

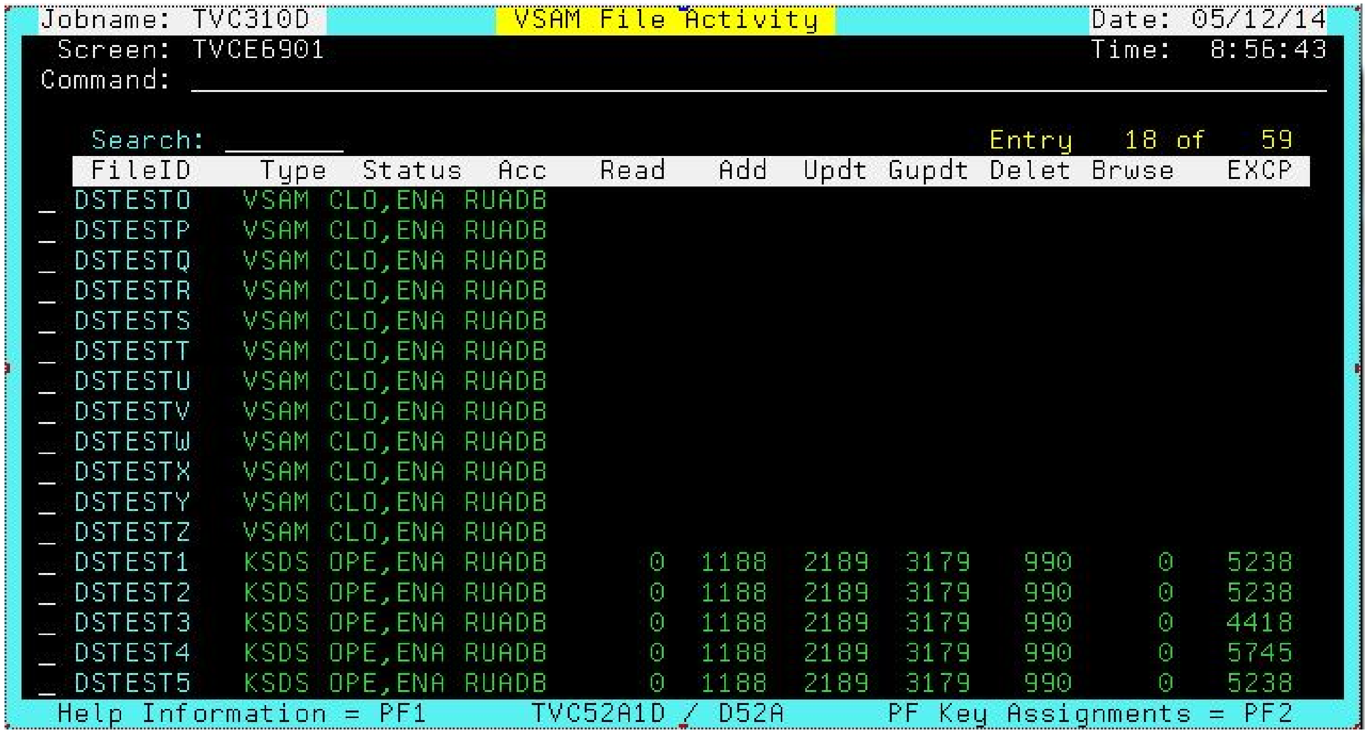
It is primarily on online tool and is able to identify approximately 150 different performance problems within CICS as a whole and provide suggestions on how to fix them.

It has the ability to view CICS control blocks to look at various issues, and is able to help with certain types of CICS task transaction abends.

Control block displays provide information about LSR Pools and VSAM files that CICS is unable to provide.

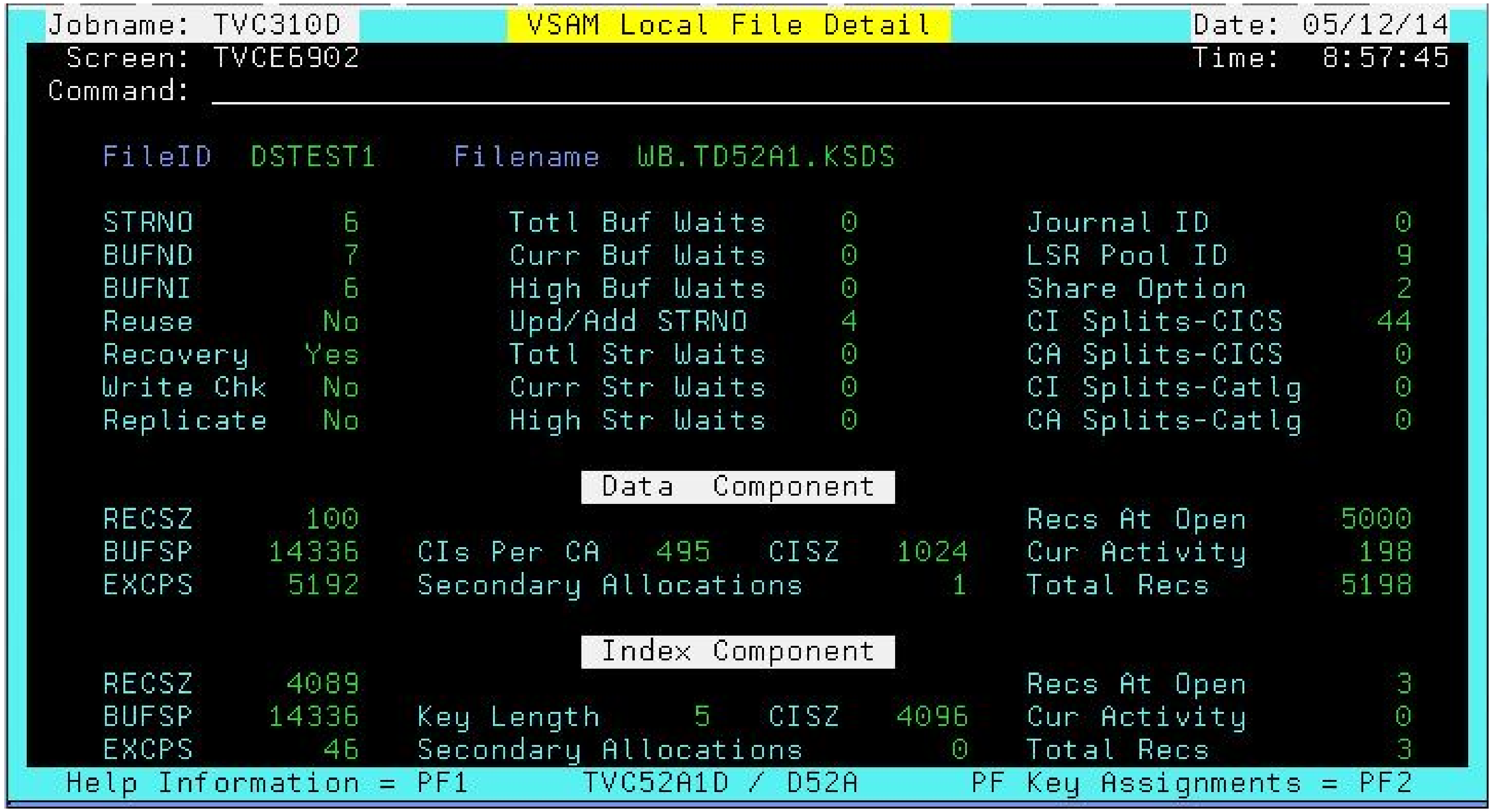
ISV Tuning Products

ASG TMON has both online and batch reporting capabilities. The slides contain sample screen and report output. Reported I/O wait times are often lower than actual, which is due to the uses of poor data provided by CICS. The best detail for CICS task performance data is seen from the Transaction Roster, but this is time-consuming to review.



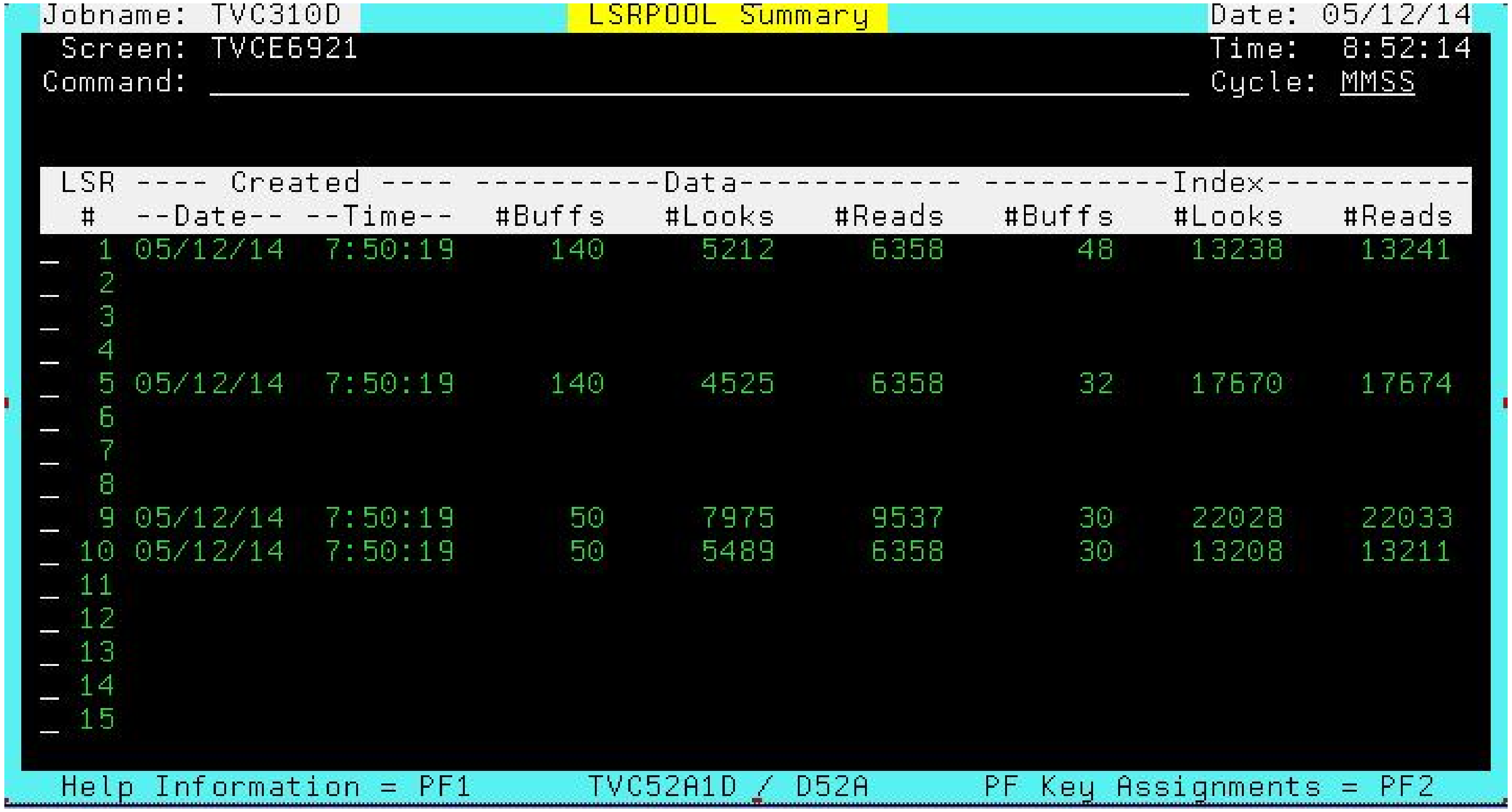
```
Jobname: TVC310D              VSAM File Activity             Date: 05/12/14
 Screen: TVCE6901                                            Time:  8:56:43
Command: ___________________________________________________________________

   Search: _______                                      Entry   18 of    59
   FileID    Type  Status   Acc    Read   Add  Updt Gupdt Delet Brwse   EXCP
_ DSTESTO   VSAM CLO,ENA RUADB
_ DSTESTP   VSAM CLO,ENA RUADB
_ DSTESTQ   VSAM CLO,ENA RUADB
_ DSTESTR   VSAM CLO,ENA RUADB
_ DSTESTS   VSAM CLO,ENA RUADB
_ DSTESTT   VSAM CLO,ENA RUADB
_ DSTESTU   VSAM CLO,ENA RUADB
_ DSTESTV   VSAM CLO,ENA RUADB
_ DSTESTW   VSAM CLO,ENA RUADB
_ DSTESTX   VSAM CLO,ENA RUADB
_ DSTESTY   VSAM CLO,ENA RUADB
_ DSTESTZ   VSAM CLO,ENA RUADB
_ DSTEST1   KSDS OPE,ENA RUADB      0  1188  2189  3179   990     0   5238
_ DSTEST2   KSDS OPE,ENA RUADB      0  1188  2189  3179   990     0   5238
_ DSTEST3   KSDS OPE,ENA RUADB      0  1188  2189  3179   990     0   4418
_ DSTEST4   KSDS OPE,ENA RUADB      0  1188  2189  3179   990     0   5745
_ DSTEST5   KSDS OPE,ENA RUADB      0  1188  2189  3179   990     0   5238
 Help Information = PF1      TVC52A1D / D52A      PF Key Assignments = PF2
```

ISV Tuning Products



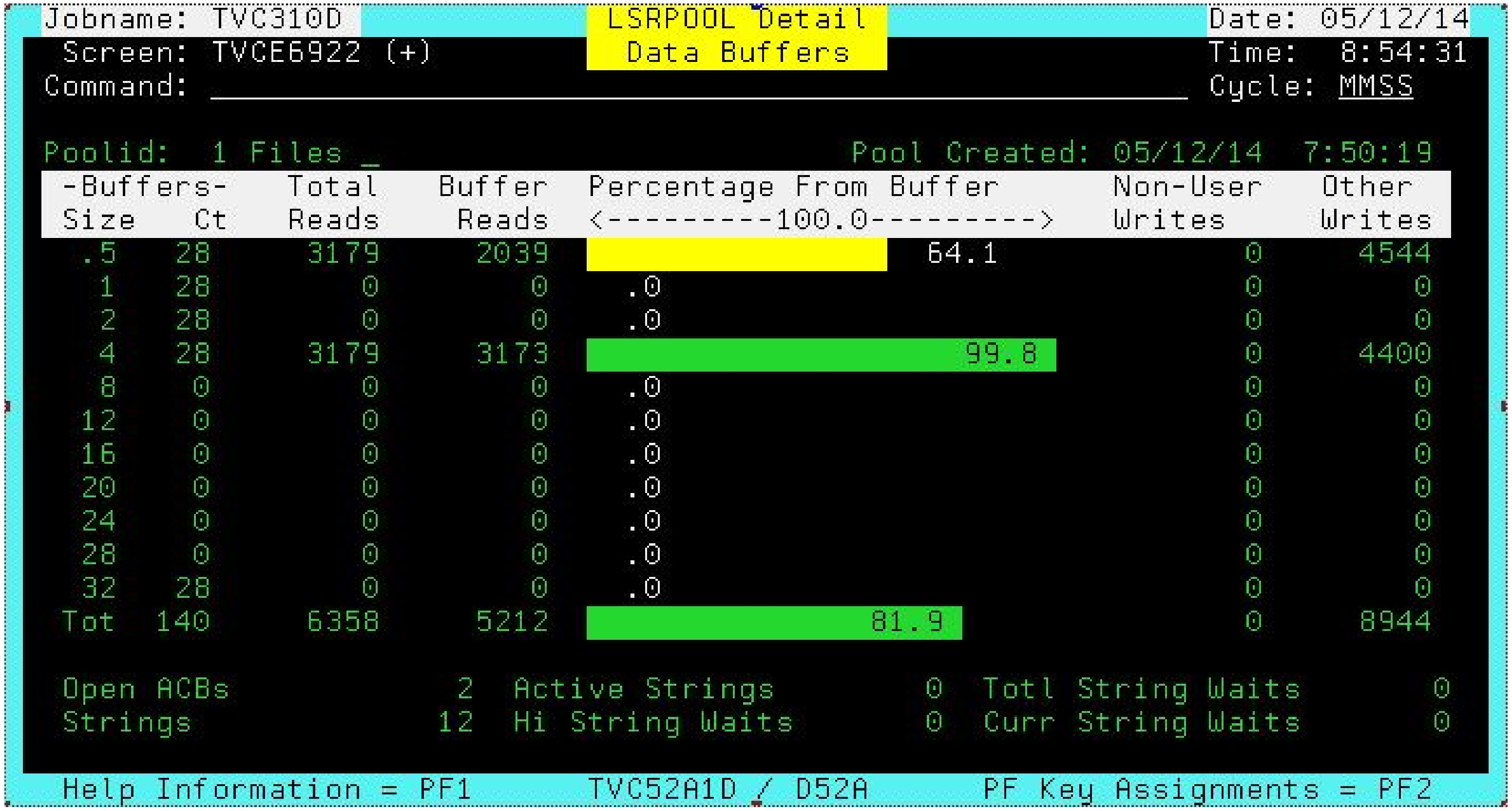
```
 Jobname:  TVC310D           VSAM  Local  File  Detail           Date:  05/12/14
  Screen:  TVCE6902                                              Time:   8:57:45
 Command: ___________________________________________________________________

    FileID   DSTEST1     Filename  WB.TD52A1.KSDS

    STRNO          6         Totl  Buf  Waits     0         Journal  ID        0
    BUFND          7         Curr  Buf  Waits     0         LSR  Pool  ID      9
    BUFNI          6         High  Buf  Waits     0         Share  Option      2
    Reuse         No         Upd/Add  STRNO       4         CI  Splits-CICS   44
    Recovery     Yes         Totl  Str  Waits     0         CA  Splits-CICS    0
    Write  Chk    No         Curr  Str  Waits     0         CI  Splits-Catlg   0
    Replicate     No         High  Str  Waits     0         CA  Splits-Catlg   0

                               Data   Component
    RECSZ        100                                        Recs  At  Open  5000
    BUFSP      14336    CIs  Per  CA    495    CISZ   1024  Cur  Activity    198
    EXCPS       5192    Secondary  Allocations           1  Total  Recs     5198

                               Index  Component
    RECSZ       4089                                        Recs  At  Open     3
    BUFSP      14336    Key  Length       5    CISZ   4096  Cur  Activity      0
    EXCPS         46    Secondary  Allocations           0  Total  Recs        3
  Help  Information  =  PF1     TVC52A1D  /  D52A      PF  Key  Assignments  =  PF2
```

ISV Tuning Products



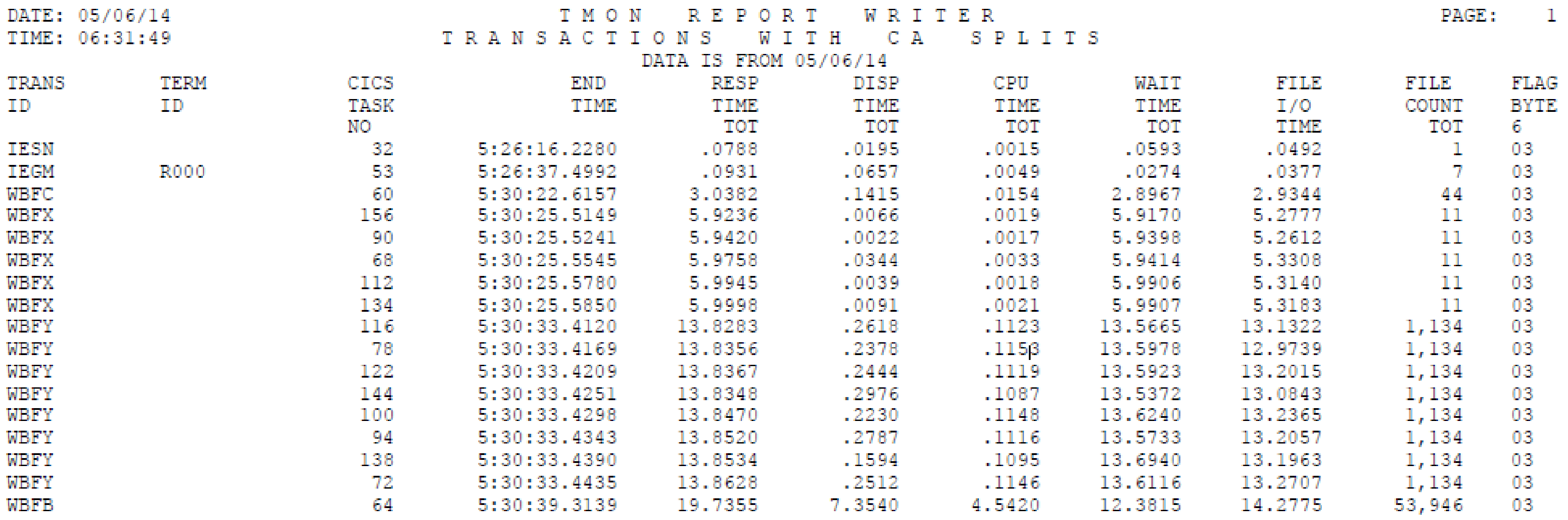
```
 Jobname:  TVC310D          LSRPOOL Summary                  Date: 05/12/14
  Screen:  TVCE6921                                          Time:   8:52:14
 Command:  ____________________________________________________ Cycle: MMSS


  LSR ---- Created ---- ----------Data------------ ----------Index----------
   #   --Date-- --Time--   #Buffs   #Looks   #Reads   #Buffs   #Looks   #Reads
 _   1 05/12/14  7:50:19      140     5212     6358       48    13238    13241
 _   2
 _   3
 _   4
 _   5 05/12/14  7:50:19      140     4525     6358       32    17670    17674
 _   6
 _   7
 _   8
 _   9 05/12/14  7:50:19       50     7975     9537       30    22028    22033
 _  10 05/12/14  7:50:19       50     5489     6358       30    13208    13211
 _  11
 _  12
 _  13
 _  14
 _  15

  Help Information = PF1     TVC52A1D / D52A     PF Key Assignments = PF2
```

ISV Tuning Products



```
 Jobname:  TVC310D              LSRPOOL Detail                Date:  05/12/14
  Screen:  TVCE6922 (+)         Data Buffers                  Time:   8:54:31
 Command:  ____________________________________________________ Cycle: MMSS

 Poolid:   1 Files _                              Pool Created: 05/12/14  7:50:19
  -Buffers-     Total     Buffer    Percentage From Buffer     Non-User     Other
  Size    Ct    Reads      Reads    <---------100.0--------->   Writes      Writes
    .5    28     3179       2039                   64.1              0        4544
     1    28        0          0     .0                              0           0
     2    28        0          0     .0                              0           0
     4    28     3179       3173                       99.8          0        4400
     8     0        0          0     .0                              0           0
    12     0        0          0     .0                              0           0
    16     0        0          0     .0                              0           0
    20     0        0          0     .0                              0           0
    24     0        0          0     .0                              0           0
    28     0        0          0     .0                              0           0
    32    28        0          0     .0                              0           0
  Tot    140     6358       5212                  81.9               0        8944

  Open ACBs                  2  Active Strings       0  Totl String Waits      0
  Strings                   12  Hi String Waits      0  Curr String Waits      0

  Help Information = PF1      TVC52A1D / D52A        PF Key Assignments = PF2
```

ISV Tuning Products



```
DATE: 05/06/14                       T M O N   R E P O R T   W R I T E R                                         PAGE:    1
TIME: 06:31:49             T R A N S A C T I O N S   W I T H   C A   S P L I T S
                                              DATA IS FROM 05/06/14
TRANS       TERM          CICS               END        RESP        DISP        CPU        WAIT        FILE        FILE     FLAG
ID          ID            TASK               TIME       TIME        TIME        TIME       TIME        I/O         COUNT    BYTE
                          NO                            TOT         TOT         TOT        TOT         TIME        TOT      6
IESN                        32     5:26:16.2280      .0788       .0195       .0015      .0593       .0492           1    03
IEGM        R000            53     5:26:37.4992      .0931       .0657       .0049      .0274       .0377           7    03
WBFC                        60     5:30:22.6157     3.0382       .1415       .0154     2.8967      2.9344          44    03
WBFX                       156     5:30:25.5149     5.9236       .0066       .0019     5.9170      5.2777          11    03
WBFX                        90     5:30:25.5241     5.9420       .0022       .0017     5.9398      5.2612          11    03
WBFX                        68     5:30:25.5545     5.9758       .0344       .0033     5.9414      5.3308          11    03
WBFX                       112     5:30:25.5780     5.9945       .0039       .0018     5.9906      5.3140          11    03
WBFX                       134     5:30:25.5850     5.9998       .0091       .0021     5.9907      5.3183          11    03
WBFY                       116     5:30:33.4120    13.8283       .2618       .1123    13.5665     13.1322       1,134    03
WBFY                        78     5:30:33.4169    13.8356       .2378       .1153    13.5978     12.9739       1,134    03
WBFY                       122     5:30:33.4209    13.8367       .2444       .1119    13.5923     13.2015       1,134    03
WBFY                       144     5:30:33.4251    13.8348       .2976       .1087    13.5372     13.0843       1,134    03
WBFY                       100     5:30:33.4298    13.8470       .2230       .1148    13.6240     13.2365       1,134    03
WBFY                        94     5:30:33.4343    13.8520       .2787       .1116    13.5733     13.2057       1,134    03
WBFY                       138     5:30:33.4390    13.8534       .1594       .1095    13.6940     13.1963       1,134    03
WBFY                        72     5:30:33.4435    13.8628       .2512       .1146    13.6116     13.2707       1,134    03
WBFB                        64     5:30:39.3139    19.7355      7.3540      4.5420    12.3815     14.2775      53,946    03
```

Gotchas - Excessive VSAM Catalog EXCPs with XXL Datasets

Use no more than 2 volumes for the Index Component.

Use 27 or less volumes for the Data Component.

Gotchas - random SDUMP in CICS

0S24I AN SDUMP OR SDUMPX MACRO WAS ISSUED with no DFHxxnnnn message.

An abend at about IKQBFC+X'392' is due to using a huge XXL SHR(4) KSDS where VSAM has duplicate LOCK resource names for multiple CI used at the same time. Using a larger CISZ may help, as would SHR(2), but ultimately needs a design change.

```
// EXEC INFOANA,SIZE=300K
 SELECT DUMP MANAGEMENT
     DUMP NAME SYSDUMP.F2.DF200000
          RETURN
   SELECT DUMP SYMPTOMS
          PRINT DATA
/*

ENVIRONMENT:
    CPU MODEL ....... 8561
    CPU SERIAL ...... 33BBF8
    TIME ............ 14:14:29:00
    DATE ............ 21/06/01
    SYSTEM ID ....... 5686VS606
    RELEASE ......... 6
    FEATURE ......... 2C
    DUMPTYPE ........ SDUMP(X)
    PROBLEM NUMBER .. .......

REQUIRED SYMPTOMS:
    PRCS/00000012
    PIDS/5745SCVSM
    RIDS/IKQBFC     VSAM module produced an SDUMP (IKQIXS is for an Index problem that resolves itself)
    REGS/FFFFF
```

Warnings and Notes about Function Shipping

Function Shipping will always result in a significant increase in CPU and elapsed request times compared to using local files, probably 10 times the CPU times and maybe 20 or more times elapsed. If AORs and a FOR are used extensively, 2, 3 or even more times the total CICS Partition CPU utilisation may occur compared to using the equivalent Local files in a single CICS partition.

Using SDTs will reduce CPU and elapsed time for Remote Base KSDS Read and Browse access. Please read the CICS TS for VSE/ESA Shared Data Tables Guide to understand how they should be defined and what the limitations are. Briefly, all that is required is to change the CSD FILE definition in the file-owning CICS partition to say it is a CICS or USER table and the maximum number of records to cache, over-estimating the number, and the CICS partitions with the REMOTE FILE definitions will start to use the SDT when you next COLD start the CICS systems. Remember to also define OPENTIME(STARTUP).

When deciding if a base KSDS is suitable for use as an SDT **and** the CSD FILE definition options are for read/only access so that the file is opened for INPUT and it is defined as SHR(2), be sure to check that there is no add/delete/update activity in batch while the file is open in CICS or the SDT will not contain the batch updates. I would NOT recommend updating in batch while CICS has a SHR(2) file open, because there is the potential for the Data and Index CIs used by CICS to out of synchronisation with the Cluster.

Warnings and Notes about Function Shipping

Moving VSAM files from the FOR to the AOR that uses them the most, and then connecting the other AORs is something that I have successfully used to reduce Function Shipping. Assimilating the AORs and FOR into fewer or a single CICS partition would also solve the problem. In both cases, there will be less overall resilience to problems.

Thank You

Please forward your questions or remarks to

poilmike@uk.ibm.com or michaelalanpoil@gmail.com

z/VSE Live Virtual Classes

z/VSE @ http://www.ibm.com/zvse/education/

LINUX + z/VM + z/VSE @ http://www.vm.ibm.com/education/lvc/

Read about upcoming LVCs on @ http://twitter.com/IBMzVSE

Join the LVC distribution list by sending a short mail to alina.glodowski@de.ibm.com

References

CICS Library	http://www-03.ibm.com/systems/z/os/zvse/documentation/#cics
Ingolf's Blog	https://www.ibm.com/developerworks/mydeveloperworks/blogs/vse/?lang=en

The effect of Paging on Performance

Tuning normally considers the role of storage in performance as there may be delays from paging that are a result of a poor Virtual/Real ratio. They are not normally VSAM's fault unless you have allocated a ridiculously large amount of buffers. Paging delays will also affect the whole CICS partition, not just VSAM.

Paging should be monitored at the z/VSE level *and* at the partition level if possible, and at the z/VM level. If NOPDS is defined in the IPL procedure, z/VSE does not page.

At the z/VM level, ensure that the z/VSE Guest VM's working set is not impacted by stealing due to other VMs' real storage demands, e.g. use CP SET RESERVE on the z/VSE Guest.

Paging at the z/VM level will not be obvious within a z/VSE system when using standard performance monitor outputs, and can cause random erratic response times that appear to have absolutely no explanation when looking at CICS and even z/VSE performance data.

Index CISZ Considerations

The required Index CISZ is related to two values.

The Data CISZ: the bigger this is, the smaller the Index CISZ tends to be because VSAM needs to keep fewer CI high keys in each Sequence Set CI, which is the lowest level of the Index; e.g. a 4K CISZ needs to map CI/CA=180 high keys per cylinder but an 18K CISZ only needs to map CI/CA=45.

The size of the keys: the bigger the keys, the more space is needed to map the keys.

VSAM has always compressed keys in the Index, and IDCAMS assumes a certain amount of compression when it checks that your Index CISZ is not too small or defaults the Index CISZ. Keys less than about 10 bytes often do not compress well as each compressed key needs 3 bytes of control information, and VSAM adds additional bytes at regular intervals within the CI. See the VSE/VSAM User's Guide and Application Programming Chapter 7.

If the Index CISZ is not big enough, some of the CA's will not be fully populated with Data CI's because there will be no room for the last "n" CI high keys in the matching Sequence Set CI. This will affect the amount of DASD space that is used and may even increase the occurrence of CA splits, although it will depend on how many CAs are affected. The Index Set comprises all Index CIs above the Sequence Set and is not affected by a CI size that is too small.

Index CISZ Considerations

If this occurs, it is difficult to identify as there are no tools to show it that I am aware of. You could use DITTO to browse or IDCAMS to print the Index component and have a look when you have a lot of time to spare. The Index component can be read as a dataset as each CI contains one logical record. LISTCAT tells you how many records and hence how many CI's are in the Index - the size of the Sequence Set is the number of CA's, normally cylinders, so the difference must be the size of the Index Set.

For safety, define the Index component CISZ by using this formula when keys are 64 bytes long or bigger.

Index CISZ = Data CI/CA * (Key length/3) and round *up* to the next LSR boundary.

For keys of length 10 bytes to 63, you could substitute (Key length/2) for safety.

LSR boundaries are 0.5K, 1K, 2K, 4K and multiples of 4K to 32K. I would use 4K unless it is grossly inefficient, or you wanted to be able to select the CISZ for special handling.

Using a value that is larger than required should reduce the total number of CIs in the Index Set and improve the performance slightly. The whole Index will use more DASD space in this case, but this is normally small compared to the amount of DASD space that is required for the Data component.

How to Calculate Available Getvis Storage

In this case, GETVIS F2,RESET was done after CICS started:

```
GETVIS F2
AR 0015 GETVIS USAGE    F2-24     F2-ANY                F2-24     F2-ANY
AR 0015  AREA SIZE:   11,260K   122,876K   (122,876K = ALLOC 120MB - 4K for SIZE=DFHSIP)
AR 0015  USED AREA:    9,828K   104,068K MAX. EVER USED:   9,828K  104,068K
AR 0015  FREE AREA:    1,432K    18,808K LARGEST FREE:     1,432K   18,808K
```

If no RESET was done, the F2-24 MAX. EVER USED = F2-24 AREA SIZE.

F2-ANY includes F2-24, but DFH0STAT output shows 24-bit and pure 31-bit.

Only 4K pages are counted. Actual USED will be less than shown as you cannot see free storage within the used pages.

MAX. EVER USED is the High-Water-Mark (AREA SIZE - LARGEST FREE is an approximate 24-bit HWM if no RESET was done).

LARGEST FREE is contiguous and may be less than FREE AREA.

"Available" contiguous storage is the smaller of (AREA SIZE - MAX. EVER USED) and LARGEST FREE.

Assuming that the data is representative, 24-bit usage could be increased by a maximum of about 1,024K and 31-bit usage by about 16MB (18,808K - 1,432K).

How to Calculate Available Getvis Storage

An LVC presentation in June 2013 has a lot of detail about z/VSE and CICS storage concepts.

Simplified CICS, VSAM, EXCP and DASD Processing

The EXEC CICS request to a local file results in the execution of one or more VSAM macros, each of which calls VSAM and normally results in one or more VSAM I/O requests:

- If it is a read, the I/O wait time is at CPU speeds if the record is in an appropriate buffer, this is a "lookaside". (SHR(4) lookaside is *very* restricted, and AIX appears to be limited as well.)
- Otherwise VSAM issues an EXCP to read or write and, in relative terms, a *lot* of CPU time is required *and* the wait time is longer:
  - EXCP causes z/VSE to build an I/O request and adds it to the device queue, it is started by SSCH if the device is not busy and VSAM gets control back. (The request is started later if the device is busy, which adds more wait time.)
  - VSAM passes control to the CICS EXCPAD exit, which issues a SUSPEND for FCIOWAIT on an ECB and the CICS task waits for I/O completion; other tasks can be dispatched while the I/O is active.
  - Meanwhile, the I/O operation is running and ends with an I/O interrupt; a read presents I/O interrupt quickly for a cache hit, otherwise the wait time increases while the DASD reads the data; a write is normally written to cache with an immediate I/O interrupt, however, a PPRC/Metro Mirror device will delay the I/O interrupt until the data is in the remote cache thus adding to the wait time.

Simplified CICS, VSAM, EXCP and DASD Processing

- The I/O interrupt stops z/VSE dispatching the current task and the waiting CICS task's ECB is POSTed; if another EXCP request is queued for the device, it is started.
- CICS is re-dispatched by z/VSE.
- CICS RESUMEs the task when it next looks for POSTed I/O requests, and dispatches it when all higher priority CICS tasks are not active; the EXCPAD exit completes and returns to VSAM.

VSAM issues more I/O requests as required. For example, an EXEC CICS READ for a KSDS uses a VSAM I/O request to read a CI from every level in the Index component and then to read a CI from the Data component. (Now imagine 150 VSAM I/Os from a CA split!)

Control returns after the VSAM macro in CICS, which will eventually return to the application after its EXEC CICS request.

z/VM uses CPU time when it intercepts the z/VSE SSCH so that it can build and issue the real SSCH. But Mini Disk Cache (MDC) may be able to read the data from its cache and avoid the real I/O, thus making some I/O faster than is possible for Native z/VSE!

z/VM CPU time is used to handle the I/O interrupt before it passes it to z/VSE.

The CPU time is accounted for as CP overhead.

CICS Monitor Data

It is not really the correct type of data for tuning VSAM, but I have included some information for reference.

See the CICS Performance Guide chapter 6 and the Customization Guide for full details.

When enabled, data records are written to DMF. However, the data may be intercepted by ISV software and written elsewhere. It can require up to a 10% CPU *delta* to collect it, e.g. 50% average over 15 minutes when disabled could mean up to 55% when enabled.

IBM provides DFH$MOLS to format it, and it produces one page per task by default!

DFH$MOLS output shows GMT, which is the assumed time zone of the real or virtual TOD clock from STCK.

CICS Monitor Data Task Performance records contain request counts and wait times, but *not by file,* which is where ISV products can help. FCIOWAIT and FCCIWAIT time is summed in FCIOWTT. Other FCxxyyyy waits are added to SUSPTIME (total task wait time).

You see the number of EXEC CICS requests and the number of VSAM API requests, so contention can be deduced.

CICS Monitor Data Task Exception records provide string and buffer wait counts.

CICS Monitor Data

This task had a *lot* of Exclusive Control FCXCWAITs, which is seen in the ratio of VSAM requests to EXEC CICS requests, and is a big component of the SUSPTIME field. CICS can perform 2 VSAM requests before entering FCXCWAIT.

You don't get activity by file, so which one(s) is a guess!

```
-----FIELD-NAME-------------------------UNINTERPRETED-------------------------------INTERPRETED---------------

DFHTASK C001     TRAN         E3D9C1F1                          TRA1
     ...
     DFHCICS T005   START        CAE81843D15C819A              13/02/11 11:30:13.1378 G.M.T.
     DFHCICS T006   STOP         CAE8184A92BA8000              13/02/11 11:30:20.2213 G.M.T. Response 7.08
     ...
     DFHFILE A036   FCGETCT       000003E8                     1000  READ or READ UPDATE
     DFHFILE A037   FCPUTCT       000003E8                     1000  REWRITE
     ...
     DFHFILE A093   FCTOTCT       000007D0                     2000 EXEC CICS file requests
     DFHFILE A070   FCAMCT        00000FA3                     4003 Actual VSAM requests are 2x bigger!
     ...
     DFHJOUR A058    JCPUWRCT      000007D6                       2006
     ...
     DFHTASK S007    USRDISPT      0000247B00000FA3              00:00:00.14942    4003  Dispatch time and count
     DFHTASK S008    USRCPUT       0000034B00000FA3              00:00:00.01348    4003  CPU time
     DFHTASK S014    SUSPTIME      00069CE200000FA3              00:00:06.93404    4003  Total wait time
     DFHTASK S102    DISPWTT       0000368700000FA2              00:00:00.22334    4002  Included in FCIOWTT etc.
     ...                                                so don't count twice
     DFHFILE S063   FCIOWTT       0000AE0E000003E4              00:00:00.71292    996  Includes any CI/CA split
     DFHJOUR S010   JCIOWTT       0001A4E9000007D1              00:00:01.72404    2001  Journal I/O
     ...
     DFHTASK S125    DSPDELAY      000002AA00000001              00:00:00.01091      1  Wait for first dispatch
     ...
     DFHTASK S129    ENQDELAY      0000002D00000001              00:00:00.00072      1  Wait for ENQ
```

CICS ESDS Notes

A standard WRITE uses RPL option DIR.

WRITE MASSINSERT uses RPL option SEQ, which might use less EXCPs if you are writing many records at a time. An UNLOCK is required to terminate the MASSINSERT.

VSAM Performance Data in a DUMP from FCP=3 Formatting

```
 FCTE.CFAL 022AFB70 FCT ENTRY  (mapped by DFHFCTDS DSECT)

    0000  C3C6C1D3 40404040 0222D380 00000000  0000001D 00EABA06 8001C400 3E000000 *CFAL    ..L...............D.....*
          FILENAME                                          BA READ/UPDATE/ADD/DELETE/BROWSE
                                                              06 FIXED/BLOCKED
                                                                 80 VSAM
                                                                   01 SYSTEM LOG
                                                                     C4 OPEN ENABLED
                                                                       00 reserved
                                                                          3E KEYLEN
    0020  40400000 00000002 00000000 7F620000  00000000 00000004 00000001 000984A7 *  ..........".................dx*
                                               READs    ADDs     UPDATEs
    0040  0006355A 00000003 00000A3E D643D92F  BCC88E42 02259940 022B0030 022B0030 *...!........O.R..H....r ........*
                   GETUPDs  BROWSEs
    0060  0221F204 024CF990 00030003 00000003  00000000 0504A844 01000000 00000000 *..2..<9...............y.........*
                   Buffer wait chain (FCTDSBWC)
                            0003 waiting for buffer to access this file
                                0003 highest waiting for buffer
                                     00000003 total buffer waits
                                                        05 LSR Data buffer size 8K
                                                          04 LSR Index buffer size 4K
                                                            A8 = KSDS + shared + LSR
                                                              .4 = Base access
                                                                 01 = LSRPOOL 1
    0080  00030003 00000000 00000002 00000001  00500000 00040005 0094A7B0 00000000 *.................&.......mx.....*
          String   String   Deletes  0000     LRECL   Data Index ACB
          limits   waits             max.              buffers
                                     string
                                     waits
    00A0  00000000 00000000 D7D9D6C4 F1F24000  40404040 40404040 00000000 00000000 *........PROD12 .        ........*
    00C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000 *................................*
    00E0  00000000 00000000 0000                                                   *..........                      *

LSR Buffer Size values:

00 Not LSR     01 0.5K     02 1K     03 2K     04 4K     05 8K     06 12K     07 16K     08 20K     09 24K     0A 28K     0B 32K
```

VSAM Performance Data in a DUMP from FCP=3 Formatting

```
SHRCTL.LSRPOOL1 0222C030 SHARED RESOURCES CONTROL BLOCK     (DFHFCTSR DSECT)

   0000  D3E2D9D7 D6D6D3F1 40000000 010100D1  024CF5D0 00000000 00320000 00050000  *LSRPOOL1 ......J.<5.............*
                                    20 bit is set if CICS dynamically calculated the number of buffers (it didn't)
                                         D1 = 209                              5 active strings
                                              ACBs                                 0 waits for string
   0020  0040000A 00000000 D643D922 F918F842  00000000 00000000 0040000A 00060000  *. ......O.R.9.8.......... ......*
                                                                Key   10 strings
                                                             length        6 strings used max.
   0040  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
   DATA buffers:
   0060  0200000C 0000000C 00000000 00000000  00000170 00000001 00000000 00000000  *................................*
          512  12 buffers                    Lookaside Read      User     Non-user
        bytes                                                    Write    Write
   0080  00000000 00000000 00000000 00000000  04000006 00000006 00000000 00000000  *................................*
                                              1024   6 buffers
                                             bytes
   00A0  00000952 0000027C 00000070 00000000  00000000 00000000 00000000 00000000  *.......@........................*
        Lookaside Read     User     Non-user
                           Write    Write
   . . .
   0260  00000000 00000000 00000000 00000000  02000000 00000000 00000000 00000000  *................................*
         INDEX buffers:                        512 no buffers
                                             bytes
   0280  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
   02A0  04000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
         1024
        bytes
```