CICS TS Tuning 101

Eugene S Hudders
C\TREK Corporation
787-397-4150
ehudders@ctrek.com
eshudders@aol.com

Overview

CICS Tuning 101

This session will provide information on what CICS areas should be reviewed and what resources can be optimized for better performance. Performance tuning of CICS systems has become a secondary objective in some installations with tuning being done when a problem arises. This presentation provides basic ideas that will provide a guide for tuning your system using the tools that you have available for tuning. It will concentrate on tuning the major resources such as CPU, I/O and Storage (virtual and real). The areas reviewed in this presentation should be analyzed by the persons responsible for CICS performance on a proactive basis.

Agenda

- Introduction to Tuning
  - Why Tune?
  - Response Time
  - Tuning Methodology
  - Identifying Constraints
  - Anatomy of a Response Time
  - Measurement Tools
  - Resources to Tune
- Tuning CPU Resources
  - ICV Parameters
  - BMS
  - Traces

Agenda

- LE Parameters
- MROLRM
- MXT
- SUDUMAX/TRDUMAX

- Tuning I/O Resources
  - What Are I/O Resources?
  - I/O Misconception
  - The Very Big I/O Picture
  - File Control Tuning
    - NSR
    - LSR
    - RLS

Agenda

- Tuning Storage Resources
  - CICS and Real/Virtual Storage
  - Tuning Real Storage
  - Tuning Virtual Storage
- Closing

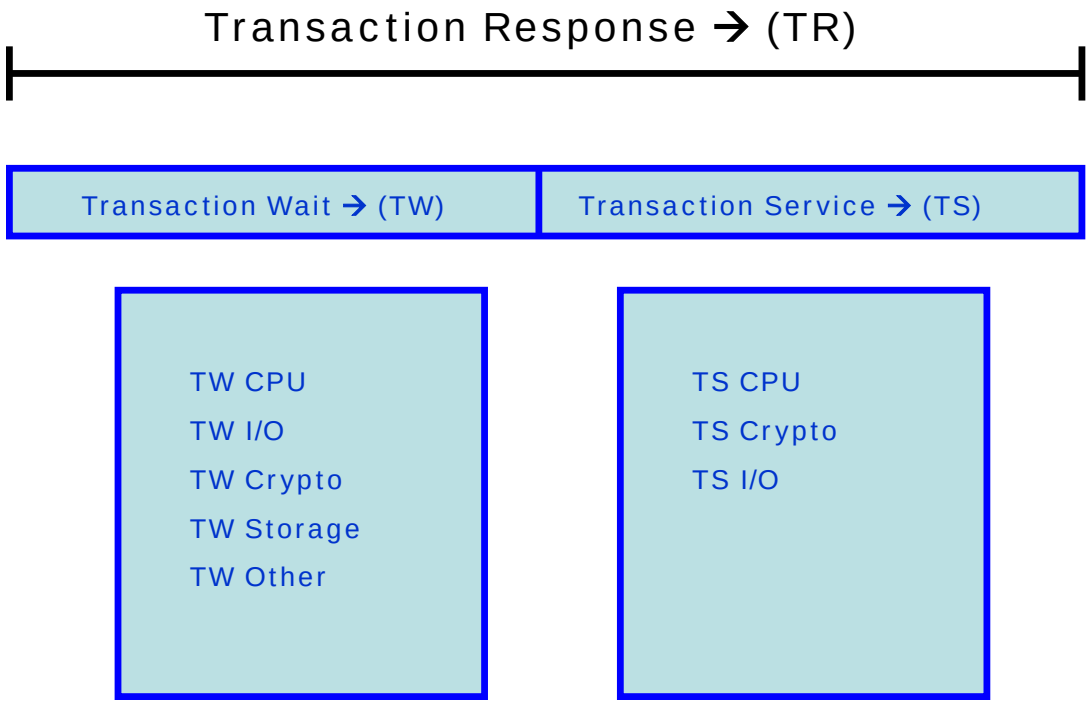
Introduction
Basic Performance Tuning Rules

- Rule # 1
  - No two CICS systems are alike
    - “What’s good for the goose may not be good for the gander”
- Rule # 2
  - Concentrate on “big hitters” first
- Rule # 3
  - You must know CICS and the applications that run in the particular CICS to properly determine performance
- Rule # 4
  - Don’t be afraid to ask
- Rule # 5
  - RTFM – there are many excellent presentations, technical articles and books that can be of use
    - Some books may have some obsolete information but still may contain valuable information

Introduction
Why Do You Want to Tune?

- Poor response times
  - Internal or system response time
  - Network response time
  - DASD response time
- Increased workload
  - Consolidations
  - Increased volume
- Hardware considerations
  - Postpone hardware upgrades
- Learning Experience

Response Time



Introduction Methodology

- General performance tuning guidelines:
  - Observe
    - Understand start-up procedure
    - Understand your workload
    - Set realistic objectives
    - Develop a base line to which you can compare
      - CPU Utilization (overall and CICS)
      - # of tasks/day or hour (peak/average)
      - Response times
  - Measure
    - Identify area(s) to tune
    - Determine measurement timeline
    - Select tools to be used
      - IBM supplied (DFH0STAT, EOD Statistics, CICS tables/RDO information, LISTCAT etc.)
      - Third party monitors/tools

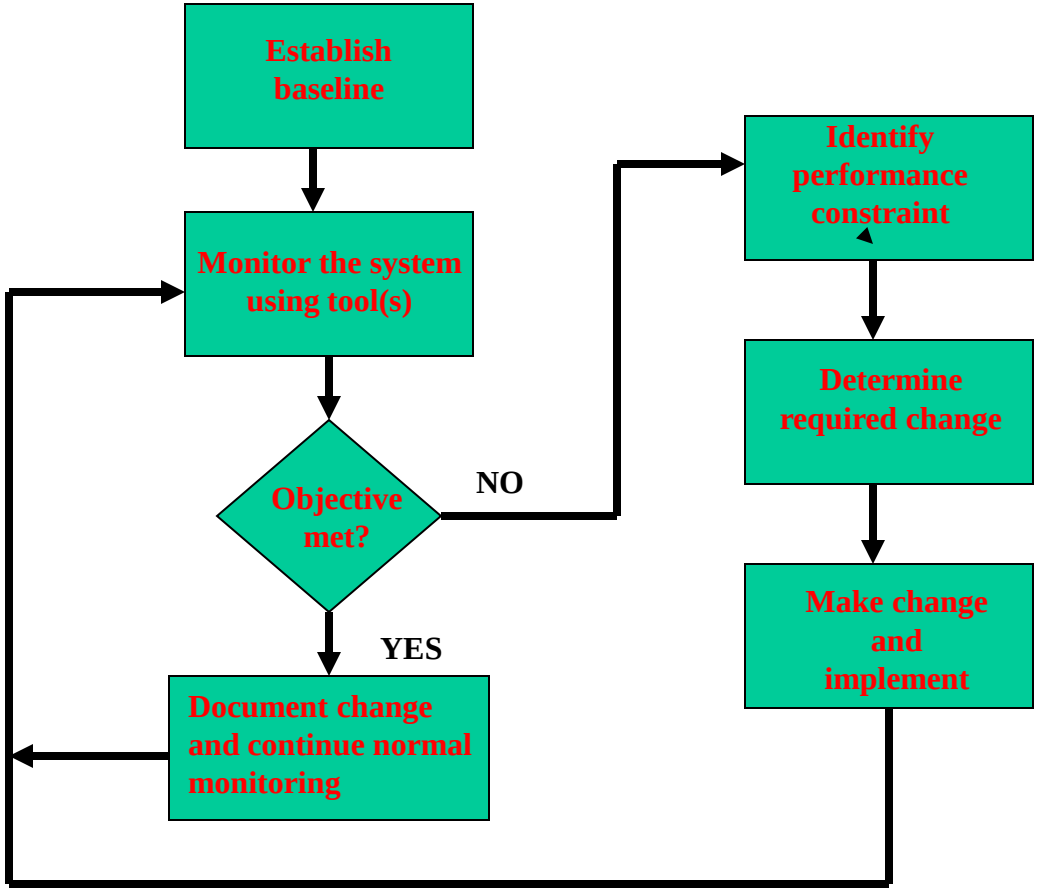
Introduction Methodology

- Analyze
  - Review outputs
  - Identify tuning opportunities
- React
  - Make appropriate changes
    - Use test/quality environments first
    - A word to the wise
      - Make major changes one at a time
      - Follow installation standards (change management)
    - Ensure backup/fallback plan is ready

Introduction Methodology

- Verify
  - Review results from changes
  - Make appropriate changes, as required
    - Some tuning may require several iterations (e.g., LSR pool tuning)
  - Go back to Measurement step until change(s) are meeting your objectives
- Implement
  - Move to production and go back to observe

Introduction
Methodology



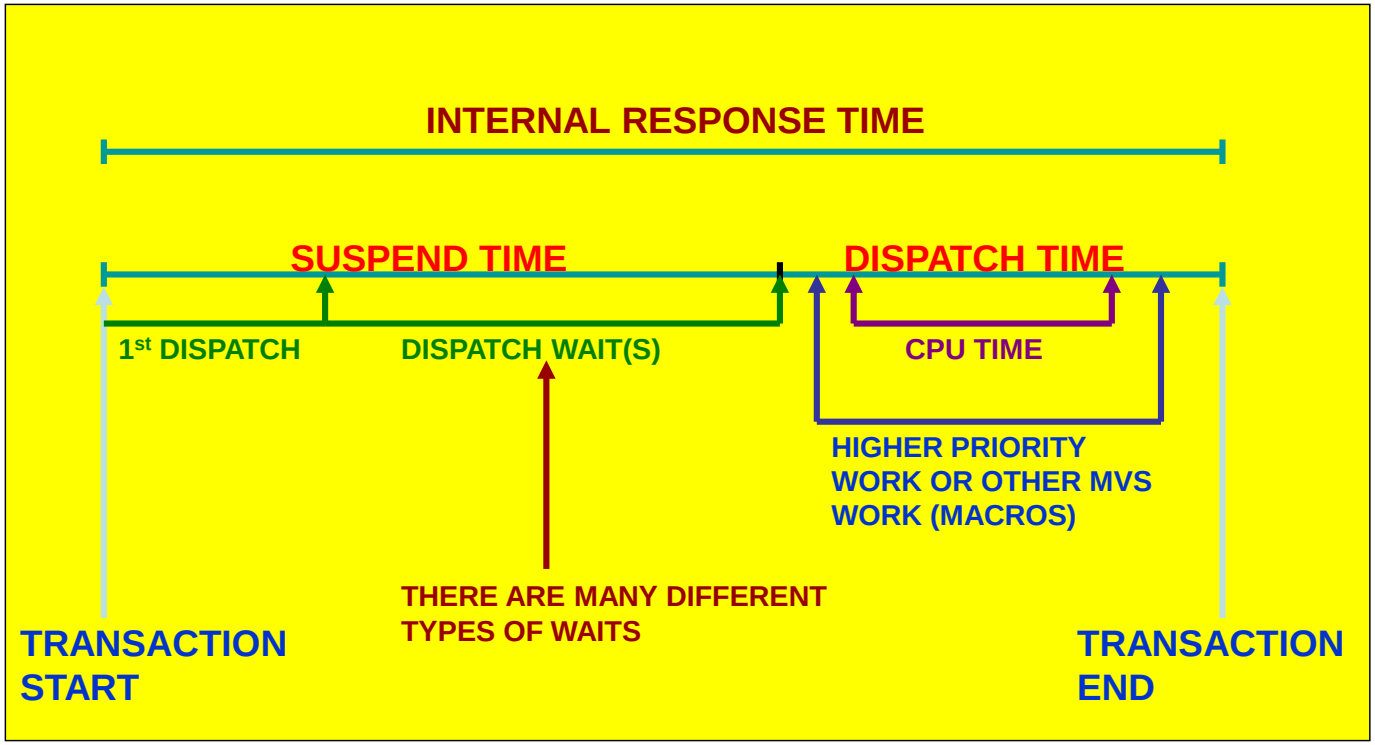
Performance Tuning

- Today, when is most performance tuning done?
  - When a problem occurs
- Why?
  - Lack of resources – budget cuts, system programming staff reductions
  - Lack of knowledge
  - Lack of interest – application/file tuning
  - Dependence on outside party
  - “If it ain’t broke, don’t fix it!” syndrome
  - Third party package and changes can only be done by the third party
- Types of queues in CICS
  - FIFO
  - LIFO
  - FISH

Identifying Constraints

- Major CICS constraints (limit conditions)
  - MXT
  - SOS (VS limits EDSA/DSA)
  - Paging (Real)
  - TCLASS limits
  - DASD problems (strings/buffers)
- Response times
  - Internal
  - External
- Hardware constraints
  - Processor cycles
  - Real Storage
  - I/O
  - Network
- Software
  - Data base design
  - Network design

Anatomy of Response Time



Anatomy of Response Time

- Response time is comprised of two elements
  - Suspend Time – the time the task is not executing (waiting)
  - Dispatch Time – the time that CICS thinks the task is executing. This time can be divided into:
    - CPU time – the time the task is executing using the CPU
    - Wait time – the time the CPU has been taken away from the task without the knowledge of CICS
      - A higher priority work is ready to execute
      - Can also be caused by the task issuing operating system macros that may entail a wait (e.g., TD Extra-partition request)
      - WLM decides to dispatch that since CICS is exceeding its goal objectives (e.g., 90% of transactions response in less than 100 ms.), the CPU should be given to an address space that is not meeting its objectives
      - The resolution of a page fault

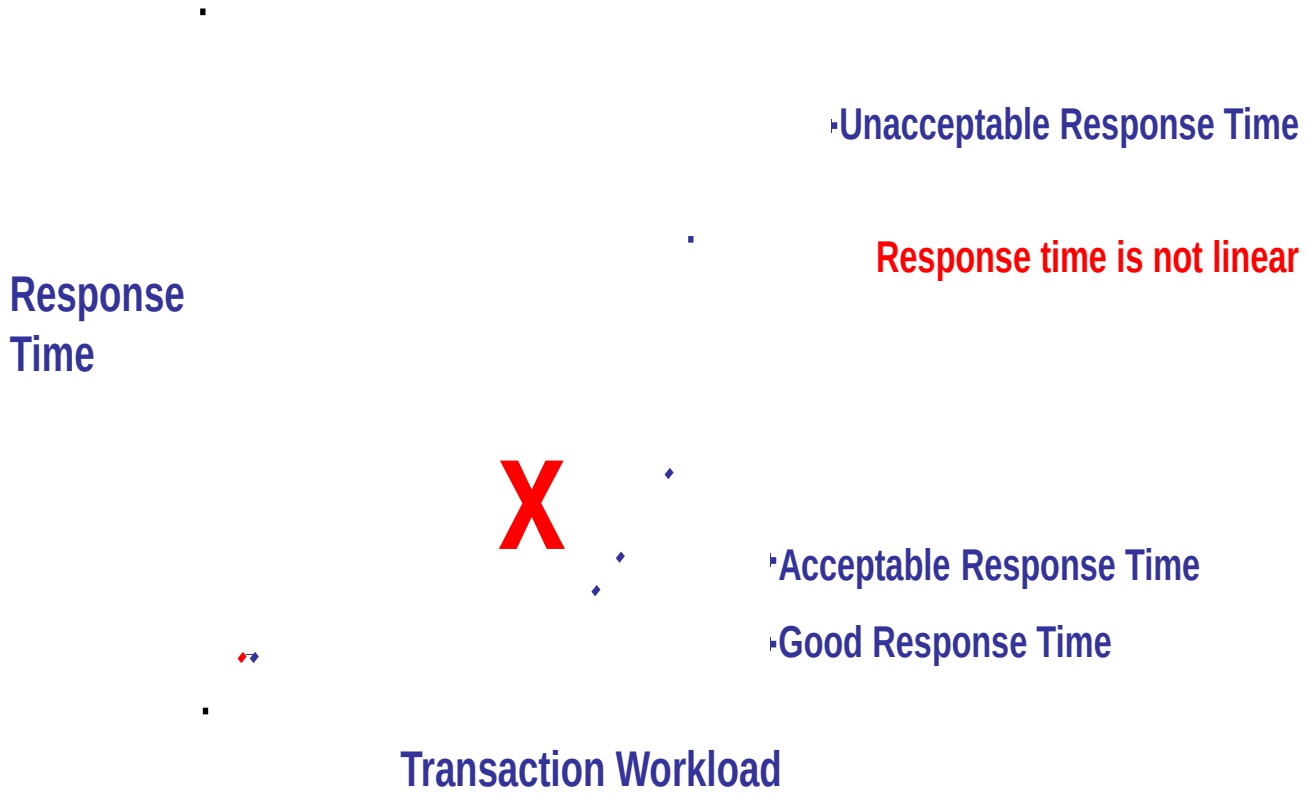
Anatomy of Internal Response Time

- Two types of “CPU” times
  - Dispatch Time – the amount of time that CICS thinks that your task is using the CPU
  - CPU Time – the actual CPU time your task is using the CPU
- CPU to Dispatch Ratio
  - Ratio = (CPU Time/Dispatch Time) * 100
  - Objective is 80% or higher
    - Used to measure the QR effectiveness
    - However, L8/L9 processing changes the picture
    - DB2 L8 priority assignments (low, equal or high) vs. QR

Anatomy of Response Time

- There are many different types of waits within a CICS system that is attributed to Suspend Time – some wait type examples are:
  - File I/O Wait
  - Terminal Wait
  - TS/TD Wait
  - Journaling Wait
  - Logger Wait
  - Inter-Region Wait
  - Other Wait types
    - ENQ Wait
    - Interval Control Wait
    - Lock Manager Wait
    - External Wait
    - CICS Waits (SOS/MXT)
    - String Waits

Anatomy of Response Time



Measurement Tools

- CICS Statistics
  - DFH0STAT
  - EOD
- SMF
- RMF
- Auxiliary Trace
- Third Party Performance Monitors
- GTF Trace
- LISTCAT
- System Dump (a real headache)

Measurement Tools

- What do you need to tune the system?
  - A tool that provides timely and accurate information
    - CPU usage
    - I/O activity and files
    - Storage usage
  - A tool that can identify potential problems
    - Highlighting potential problems
    - Calculates and relates information
  - A tool that can make recommendations

Measurement Tools

- EOD statistics
  - Advantages
    - Provide information about CICS resources
    - Provides numeric information about CICS resources that can be used to tune
  - Disadvantages
    - Fairly large output that is not easy to digest
      - So, people concentrate on a few areas
      - Hard to identify problem areas
        - » Many calculations have to be done manually
        - » Does not provide information about file definitions that may require attention
        - » You have to know the formulas to be applied

Resources to Tune

- CPU
  - Payback is harder to measure with today's fast processors
    - What showed a significant CPU improvement may not demonstrate much
      - Saving 10K instructions on a 20 MIPS system may not be measureable in an 800 MIPS system
  - So, the payback comes from saving of CPU resources that is repeatable across several/many CICS systems
  - Application tuning is still important but remember that between 50 to 70% of the CPU used by a task is attributable to CICS management programs

Resources to Tune

- I/O
  - Probably the biggest payback item for tuning the system
    - Tuning I/O
      - Can save CPU resources
      - Can improve response times
      - Can reduce enqueues for resources
  - Achieved via proper buffering of files
  - There is a misconception associated with new disk technology – DASD cache

Resources to Tune

- Storage – Virtual and Real
  - VS below continues to be a challenge
    - Outside of moving applications above the line, options to resolve are limited
  - SOS conditions above the line
    - Can be caused by rogue transactions
    - Increased activity
    - Bottlenecks
    - Too low of an EDSALIM requested
  - SOS above the Bar is new and probably rare for most installations but resolution means CICS recycle
  - Real storage problems are reflected by increased paging in the system
    - Availability of inexpensive real storage has reduced problems

CPU

- Today's processor speeds can mask the effectiveness of CPU tuning
  - You need to concentrate on repeatable conditions across multiple CICS systems
  - Most generalized CPU savings come from fixing SIT parameters
    - ICV, ICVTSD and ICVR (not directly)
    - BMS NODDS
    - Trace (Internal and Auxiliary)
    - HPO
    - FEPI
    - LE parameters
      - RUWAPOOL
      - AUTODST
    - MROLRM
    - SUDUMAX and TRDUMAX

CPU

- Interval Control Parameters
  - ICV → determines how long CICS will sleep when there is no work to do – 3000 ms. and up
  - ICVTSD → obsolete parameter should be set to zero
  - ICVR → used to indicate how long a task can execute w/o returning control to CICS default is too high – 5000 ms.

CPU

- BMS NODDS
  - Used to identify the use of Device Dependent Suffix BMS maps (DDS) – this is the default
  - If DDS maps are not used, there are two calls made to locate the desired map. First call for the DDS map and when a not found condition occurs, a second call for the generic map
  - The vast majority of the accounts use the generic map, so avoid the call for the specific map by coding NODDS

CPU

- Traces
  - Internal trace can add up a lot of CPU cycles from transactions that execute a lot of Command Level commands
    - Can use between 6 to 30% more CPU cycles when internal trace is on
    - Overhead much greater with Auxiliary trace on
  - Unfortunately, some performance monitors require the internal trace to be turned on
  - If not needed, turn it off
  - Can be manually turned on using CETR

CPU

- HPO
  - Reduces the overhead in the communication between CICS and VTAM
  - Requires an SVC
  - May be useful in CICS regions that have communications with VTAM (e.g., TOR with 3270 emulation support)
- FEPI
  - Uses CPU cycles when defined even if you do not use FEPI
  - If not needed, turn it off

CPU

- LE parameters
  - RUWAPOOL can reduce CPU overhead when linking to the same subroutine multiple times within one task
    - Working Storage area is not cleared on reuse
  - AUTODST can reduce the CPU overhead by consolidating GETMAIN commands by monitoring the high allocated storage by task
    - Can over-allocate storage if you enter an exception program that uses a lot of storage
      - Can be fixed by issuing a new copy for the program

CPU

- MROLRM
  - Indicates whether or not you want a long running mirror or not
    - The mirror task is not detached when the request is completed
    - Allows for the caller to reuse the mirror task without having to go through the attach/detach process
    - Reduces overall CPU utilization where mirrors execute
    - Elapsed time will reflect a lot of wait time

CPU

- MXT
  - There is one Performance Block (PB) per MXT entry
    - MXT + 1 are the total of user PBs
  - System PBs are 10% of MXT or 20 whichever is higher
  - PBs are scanned every so often by the WLM address space
    - Every 250 ms. for transaction controls (Goal Mode)
    - Every 2.5 seconds for region controls (Velocity)
  - An excessive over assignment of MXT can result in high CPU usage when using WLM transaction controls
    - CPU cost is in the WLM address space
  - Try to set MXT so that the peak number of transactions are around 50 to 70% of MXT
  - CICSTS52 default is 250

CPU

- SUDUMAX and TRDUMAX
  - Used to indicate the number of a particular type of system and/or transaction dumps you want to take
  - Excessive dump activity can result unnecessarily
    - AP0001/SR0001 versus ASRA dumps
    - Resource overhead
    - Disk space requirements for system dumps
  - Default is 999 or unlimited
  - Reduce figures – write routine at PLT that can set individual type of dumps

I/O

- Objective is to reduce physical I/O activity to the DASD farm
- Tuning entails adjusting buffers (Data and Index) so that the desired information is found in virtual storage and a physical I/O operation is avoided
- There is a misconception of no need to tune because DASD is faster and backed by cache that avoids the slow part of disk access (seek, rotational delay and finally the transfer of the data)
- VSAM data set tuning is still valid

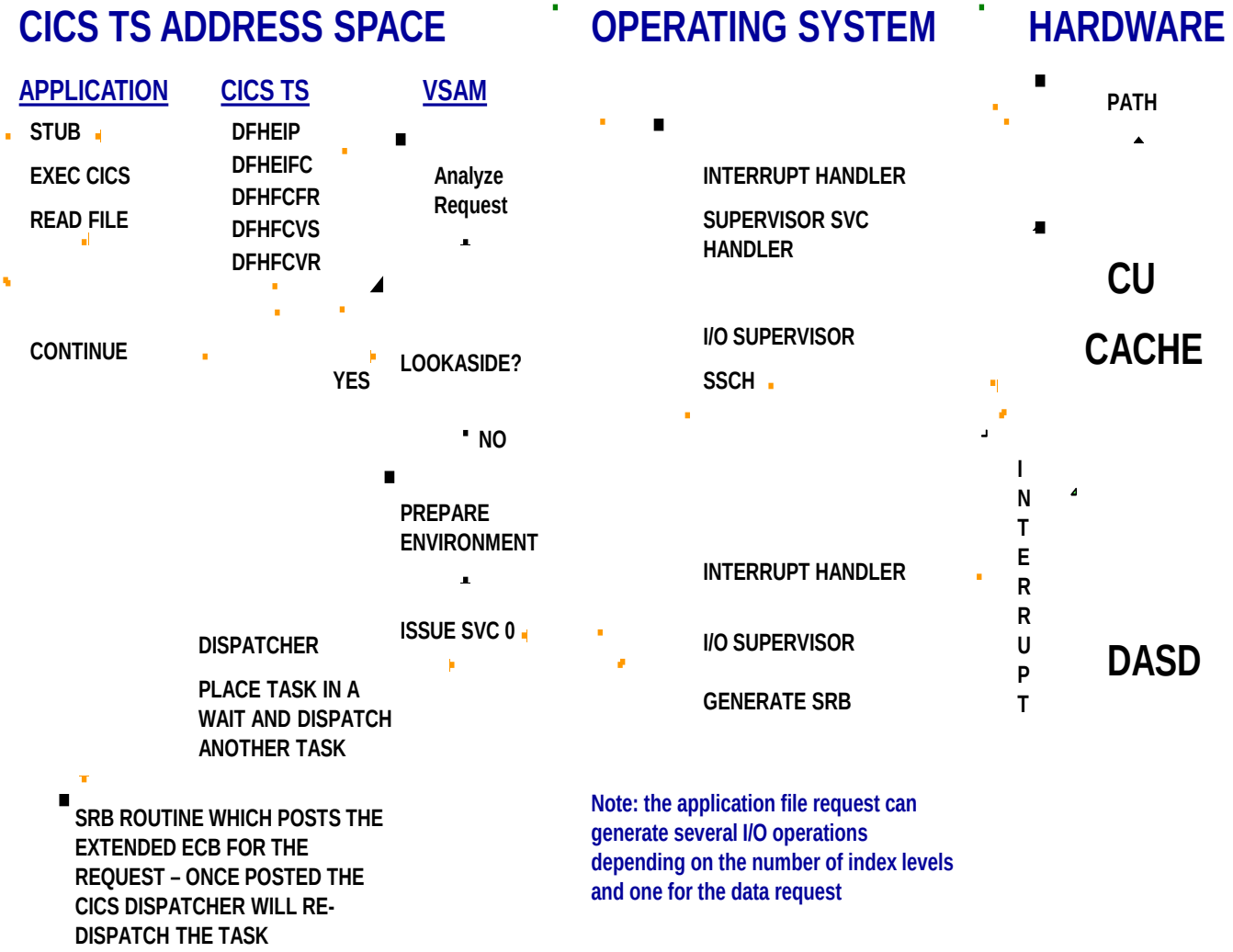
I/O

- There is a major difference if the requested information is found in virtual storage versus if the requested information is found in the disk cache
- To get to the record in the cache you still have to go through a lot of CPU instructions
- This process generates more instructions than if the desired record was found in virtual storage
- This process is called a “look-aside” hit

I/O Misconception

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

The Very Big I/O Picture



File Control Tuning

- Select buffering Technique
  - Non-Shared Resources (NSR)
    - Can provide for better performance for sequential operations (a.k.a. batch)
    - Best suited for SHROPT 4 files
  - Local Shared Resources (LSR)
    - Self tuning as resources are allocated by activity
    - More efficient use of VS
    - Better buffer look-aside
    - Better read integrity
    - Allows for
      - 255 separate pools (usually one or two)
      - Separate index and data buffers
  - Record Level Sharing (RLS)
    - Self tuning as resources are allocated by activity
    - Buffers are outside the region
    - Tuning is a “black hole” environment
    - Best for file sharing across MVS complexes

File Control Tuning

- String allocation provides concurrency
  - One string required per request
  - Short on strings indicates a request that was queued
  - Key to number strings is the look-aside hit ratio
    - NSR
      - Represents the maximum number of concurrent requests that can be issued to that file
      - Strings belong to the file – unused strings cannot be used by any other file
      - Will require one data and one index (KSDS) buffer
        - » Strings can be expensive VS wise if over allocated
      - Usually 3 to 10 strings are sufficient for most files

File Control Tuning

- LSR
  - Two types of string allocations
    - » File – maximum number of concurrent requests that can be issued to the file (usually 3 to 10)
    - » Pool – the actual number of strings that will be shared by the files assigned to the pool (usually 20 to 50)
    - » Two possible short on string conditions: pool and file
  - Pool strings belong to the files assigned to the pool
  - String number is independent of buffer allocations
    - » String buffers are dynamically allocated when needed
- RLS
  - String availability – 1024
  - Data Space used for buffers (31-bit) or in the SMSVSAM address space (64-bit)

File Control Tuning

- NSR Tuning
  - Main tuning item is to ensure that there are sufficient index buffers to hold the index set buffers (ISI—second level and above) in virtual storage
    - Finding the index CI in virtual storage avoids a physical I/O operation that improves response time and lowers CPU utilization
  - Tuning of lowest level of indices called the sequence set index (SSI—lowest level of indices) depends on string buffers assigned
    - Look-aside at this level is not good
  - Look-aside for the data depend on the string buffer assigned
    - Look-aside at this level is not good
  - NSR is Batch

File Control Tuning

- LSR
  - Tuning should include
    - Ensure static definition of buffer pool(s)
    - Separate data and index buffers
    - Allocate sufficient pool strings
    - Concentrate on index buffers first
      - Try to achieve 95% or better look-aside for index
      - Try to standardize index buffer sizes
        - » Round index CISZ to buffer size
        - » Be careful with default index CISZ for data sets with large key sizes
        - » Be suspicious when data and index CISZ are equal
        - » Small data CISZ tend to generate larger index CISZ
    - Allocate data buffers after tuning index buffers
      - Try to achieve 80% or better look-aside for data
    - Try to achieve a combined look-aside of 93% or better

Real/Virtual Storage

- CICS performance can be affected by the lack of
  - Real storage – paging
    - Loss of control by CICS
    - Erratic response
      - Same type of transaction provides a wide and varied response
    - Information is available in RMF reports
  - Virtual storage – the “Achilles Heel”
    - SOS
    - System slowdown
      - Response for all transactions slows down
- Lack of MVS storage will have a very negative effect on CICS

Real Storage

- Possible contributor to paging
  - Over use of TS MAIN
  - Unused resources
    - Unused table definitions
    - Excessive buffer/string allocations
    - Use of TEST functions by application programs
    - Not using the compiler optimizer function
    - Unused TWA/TCTUA specifications
  - Excessive MXT specification allowing many tasks to be attached

Real Storage

- Possible solutions
  - These two solutions may not be possible if the hardware is out of marketing
    - Buy more storage ($)
    - Get a faster CPU ($$)
  - Tune for real storage
    - Improve Page Data Set (PDS) response
    - Storage protect via WLM CICS region definitions
    - CICS management modules in LPA
    - Reduce MXT
    - TCLASS big real consuming storage transactions
    - Eliminate unused resources (time consuming)
    - Speed up transaction response

Virtual Storage

- Virtual Storage (VS) is the “Achilles Heel” of CICS
- Two types of VS:
  - VS managed by CICS
    - DSA/EDSA/GDSA
    - Obtained by CICS from the operating system
  - VS managed by the Operating System
    - Size controlled by
      - REGION=
      - MEMLIMIT=
    - Allocation can be controlled by
      - IEFUSI
      - IEALIMIT (obsolete)

Virtual Storage

- CICS managed storage are called dynamic storage areas (DSA)
  - The title depends on where the storage is allocated
    - DSA → storage allocated below the 16 MB line
      - CDSA/UDSA/RDSA/SDSA
    - EDSA → storage allocated above the 16 MB line
      - ECDSA/EUDSA/ERDSA/ESDSA/ETDSA
    - GDSA → storage allocated above the 2 GB Line called the Bar
      - GCDSA/GUDSA/GSDSA

Virtual Storage

- When there is insufficient storage to satisfy an unconditional GETMAIN request, the system is considered under stress and an SOS condition is raised
  - Messages
    - DFHSM0131 – below the 16 MB line (DSA)
    - DFHSM0133 – above the 16 MB line (EDSA)
    - DFHSM0606 – above the 2 GB Bar (GDSA)
- The first warning usually received is that programs are being “compressed” out of storage and are being reloaded into VS

Virtual Storage

- The major problem with VS is that people say → VS is free
  - However, you cannot buy any like you do with real storage
    - Try buying VS below the line
- The VS “cost” is space on disk and the overhead of translating virtual to real addresses

Virtual Storage

- Possible solutions depend on where the SOS occurred
  - Below the line
    - Best defense provided by CICS is to define transactions with SPURGE(YES) with a DTIMOUT value
    - Ensure DSALIM is at maximum size possible (CEMT)
    - Migrate programs to 31-bit addressing
    - Link BMS maps above the line
    - Reduce MXT
    - TCLASS high VS below the line transactions
    - Eliminate unused TWAs
    - If possible, change TCTUAs to above the line and/or reduce/eliminate allocation
    - Split CICS region using MRO
    - Storage violations
    - Ensure that each shared GETMAIN have an offsetting FREEMAIN
    - Reduce task residency time (tuning)
    - Ensure proper storage use by task

Virtual Storage

- Above the line
  - Best defense provided by CICS is to define transactions with SPURGE(YES) with a DTIMOUT value
  - Ensure EDSALIM is at maximum size possible (CEMT)
  - Reduce MXT
  - TCLASS high VS above the line transactions
  - Split CICS region using MRO
  - Storage violations
  - Ensure that each shared GETMAIN have an offsetting FREEMAIN (SDSA/ESDSA)
  - Reduce task residency time (tuning)
  - Ensure proper storage use by task
  - Eliminate unneeded resource definitions

Virtual Storage

- Above the Bar
  - Fortunately, from a non-Java environment, the VS use above Bar is not big
  - Increasing MEMLIMIT size requires a recycling of the CICS region
  - VS storage above the Bar is only supported by Assembler at this time
  - Major user type VS usage above the Bar is for Container storage
  - CICS usage is for control blocks and certain resource definitions such as Loader Domain program resource definitions

Closing

- To tune CICS you need the right tool that can identify areas that require attention and if possible recommendations
- Start by tuning the SIT parameters
  - Easy to implement
  - Easy to monitor
- Biggest payback is I/O reduction

Summary

Thank You for Your Attention