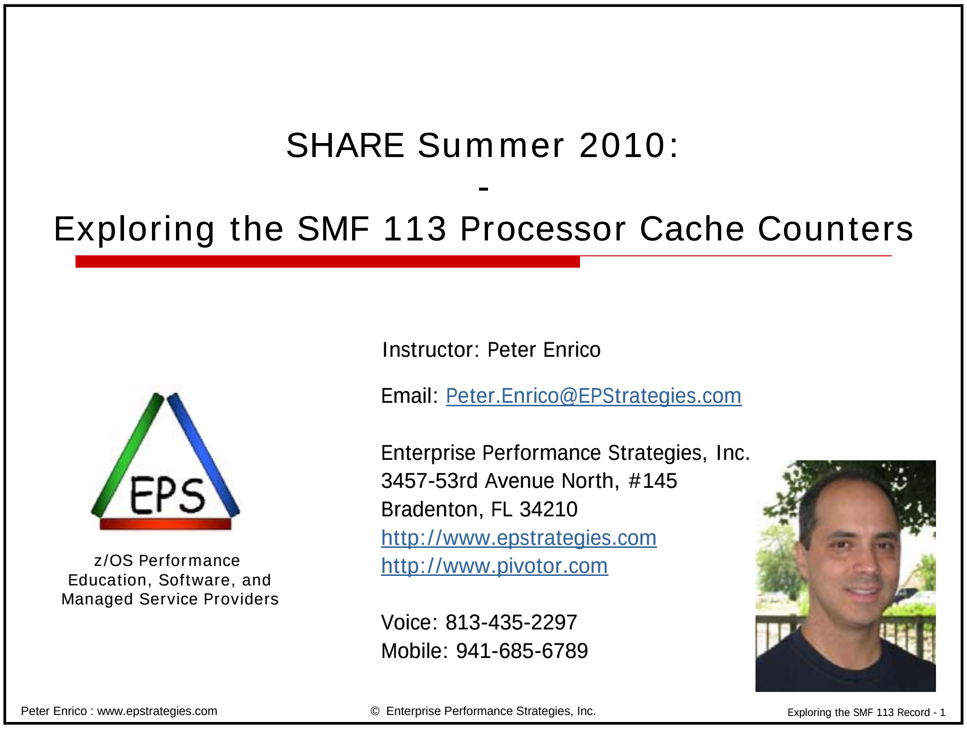


SHARE Summer 2010:
-
Exploring the SMF 113 Processor Cache Counters

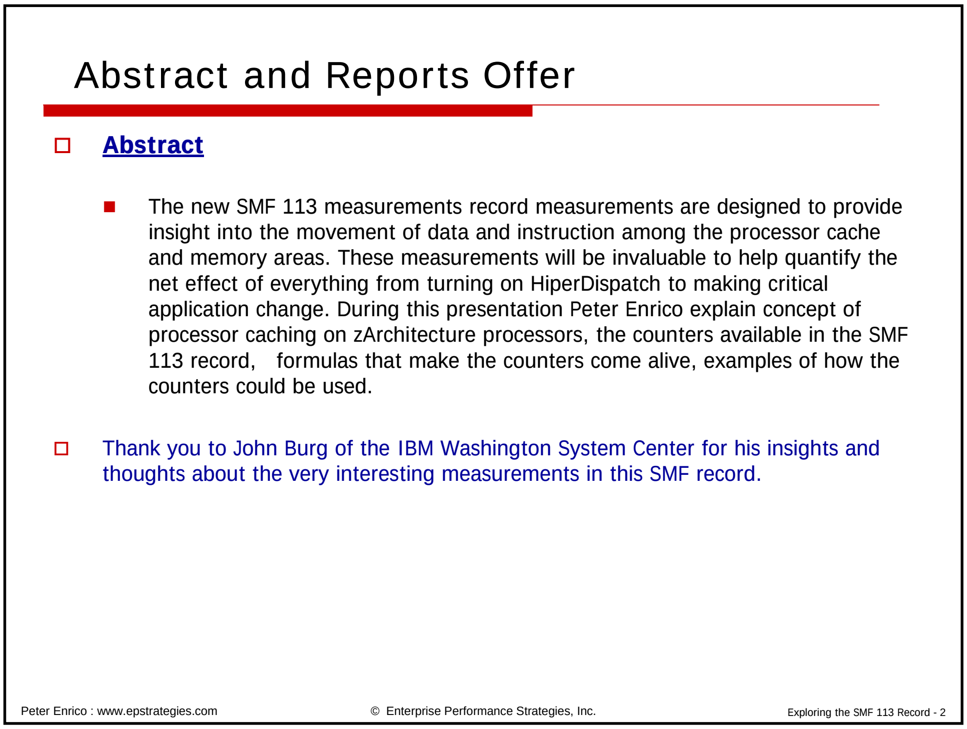
Instructor: Peter Enrico

Email: Peter.Enrico@EPStrategies.com

Enterprise Performance Strategies, Inc.
3457-53rd Avenue North, #145
Bradenton, FL 34210
http://www.epstrategies.com
http://www.pivotor.com

Voice: 813-435-2297
Mobile: 941-685-6789

z/OS Performance
Education, Software, and
Managed Service Providers

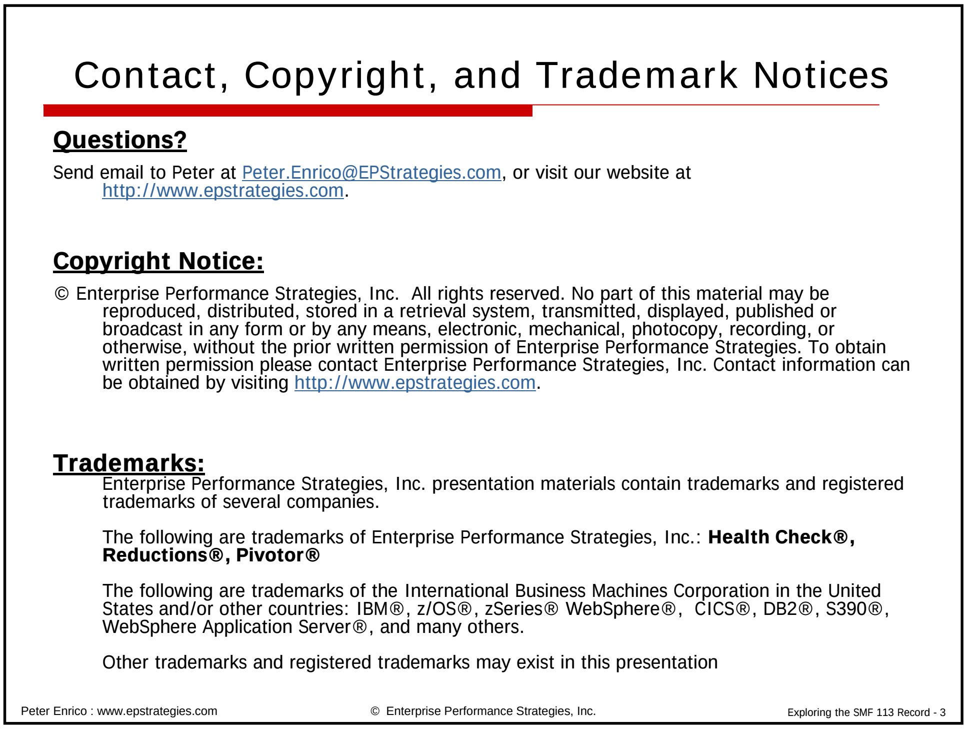


Abstract and Reports Offer

- **Abstract**

- The new SMF 113 measurements record measurements are designed to provide insight into the movement of data and instruction among the processor cache and memory areas. These measurements will be invaluable to help quantify the net effect of everything from turning on HiperDispatch to making critical application change. During this presentation Peter Enrico explain concept of processor caching on zArchitecture processors, the counters available in the SMF 113 record,   formulas that make the counters come alive, examples of how the counters could be used.

- Thank you to John Burg of the IBM Washington System Center for his insights and thoughts about the very interesting measurements in this SMF record.



Contact, Copyright, and Trademark Notices

Questions?

Send email to Peter at Peter.Enrico@EPStrategies.com, or visit our website at http://www.epstrategies.com.

Copyright Notice:

Trademarks:

Current Class Schedule

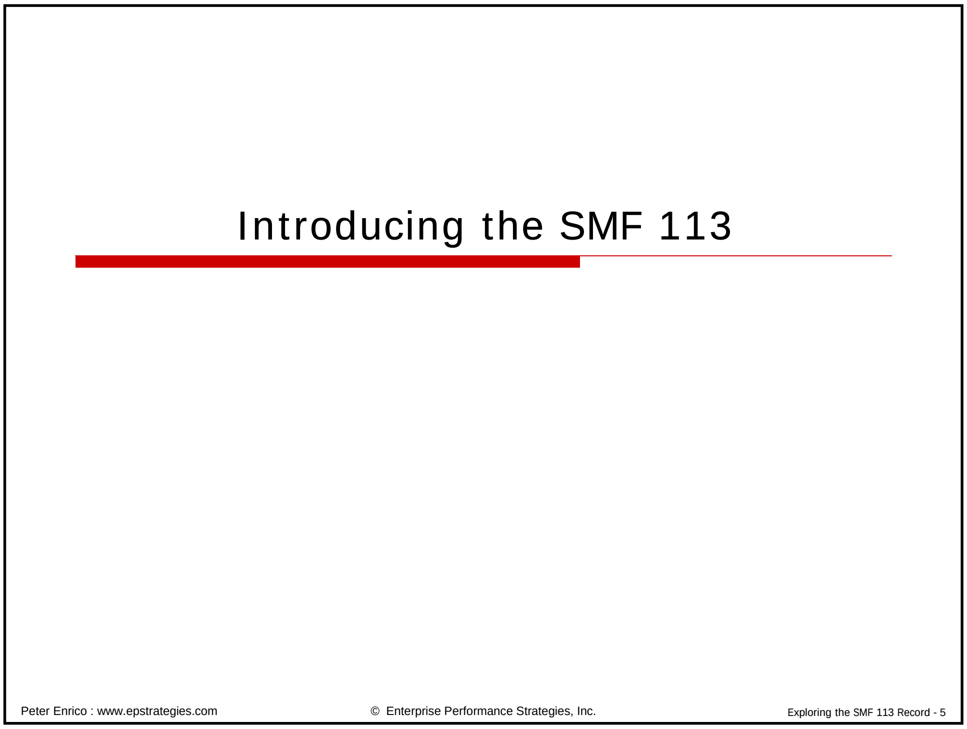
□ WLM Performance and Re-evaluating of Goals
- Instructor: Peter Enrico
- October 18 - 22, 2010, St. Paul, MN

□ Essential z/OS Performance Tuning
- Instructor: Peter Enrico and Tom Beretvas
- Currently not schedule

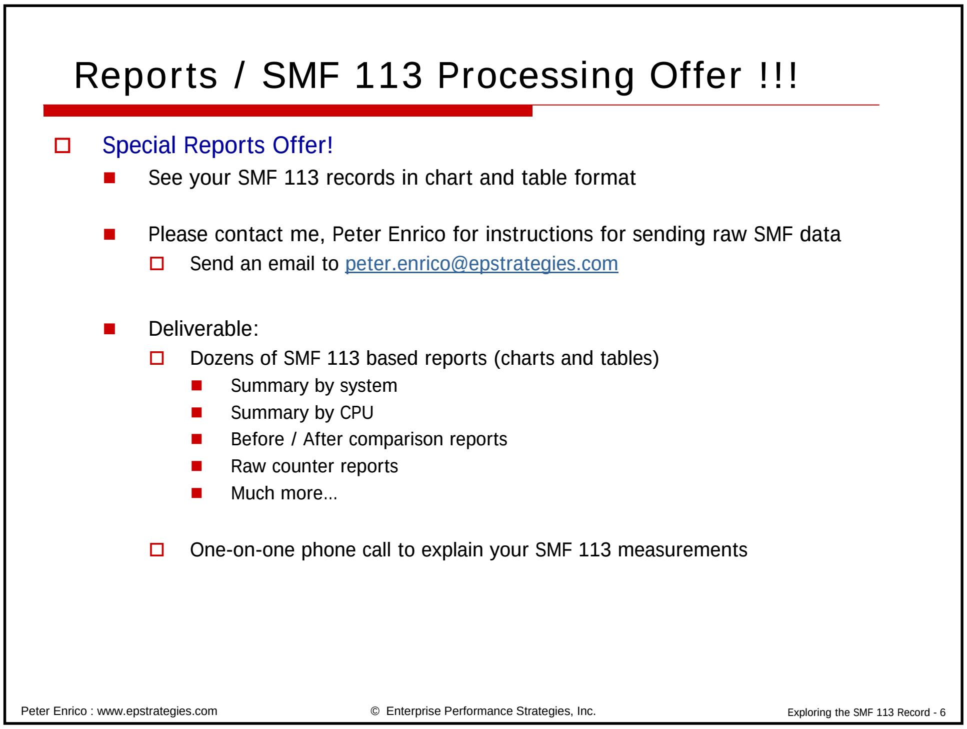
□ Parallel Sysplex and z/OS Performance Tuning
- Instructor: Peter Enrico
- September 13 - 17, 2010, Philadelphia, PA

Other classes are planned for Europe

Maybe some additional USA classes

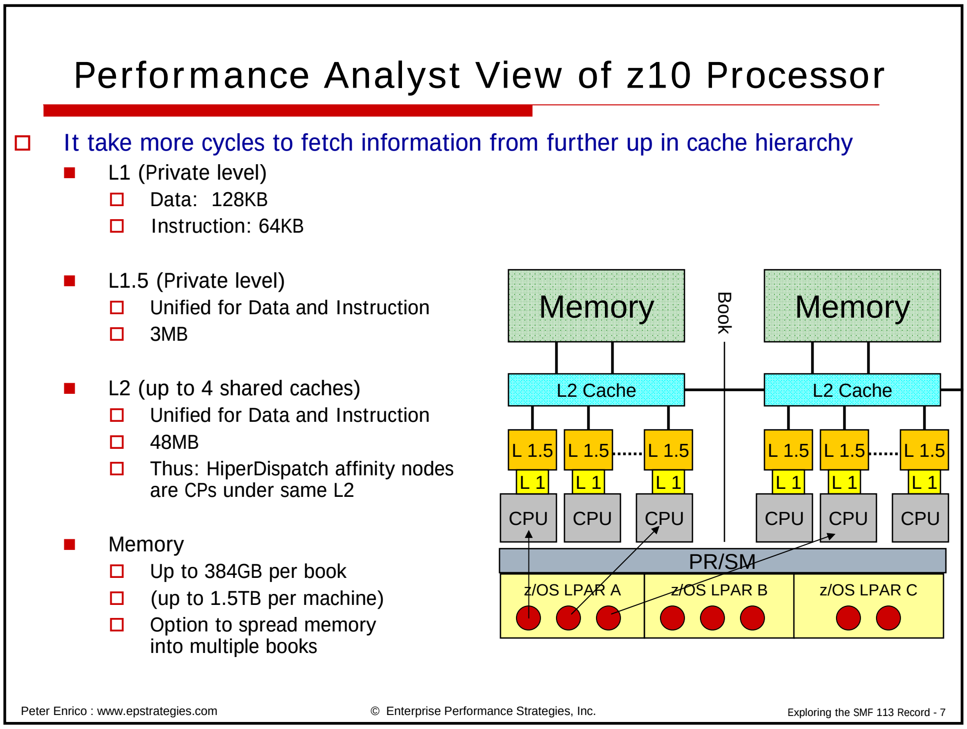


Introducing the SMF 113



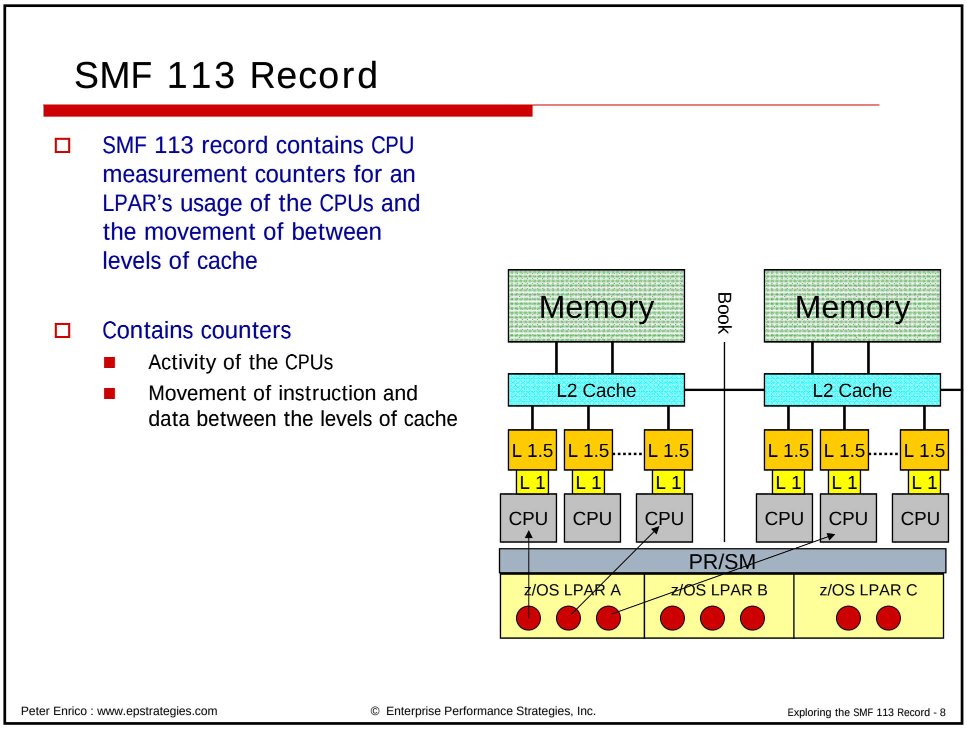
Reports / SMF 113 Processing Offer !!!

- Special Reports Offer!
  - See your SMF 113 records in chart and table format
  - Please contact me, Peter Enrico for instructions for sending raw SMF data
    - Send an email to peter.enrico@epstrategies.com
  - Deliverable:
    - Dozens of SMF 113 based reports (charts and tables)
      - Summary by system
      - Summary by CPU
      - Before / After comparison reports
      - Raw counter reports
      - Much more…
    - One-on-one phone call to explain your SMF 113 measurements



Performance Analyst View of z10 Processor

- It take more cycles to fetch information from further up in cache hierarchy
  - L1 (Private level)
    - Data: 128KB
    - Instruction: 64KB
  - L1.5 (Private level)
    - Unified for Data and Instruction
    - 3MB
  - L2 (up to 4 shared caches)
    - Unified for Data and Instruction
    - 48MB
    - Thus: HiperDispatch affinity nodes are CPs under same L2
  - Memory
    - Up to 384GB per book
    - (up to 1.5TB per machine)
    - Option to spread memory into multiple books



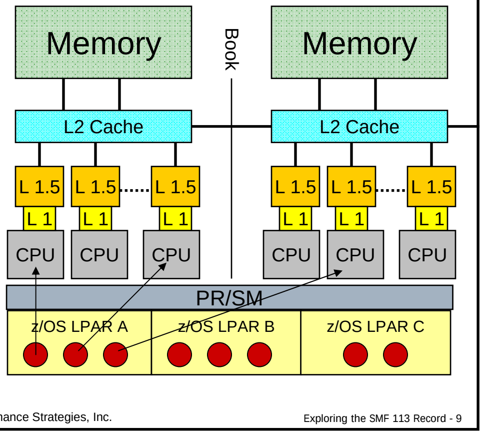
SMF 113 Record

- SMF 113 record contains CPU measurement counters for an LPAR's usage of the CPUs and the movement of between levels of cache
- Contains counters
  - Activity of the CPUs
  - Movement of instruction and data between the levels of cache

Memory
Book
Memory
L2 Cache
L2 Cache
L 1.5
L 1.5
L 1.5
L 1.5
L 1.5
L 1.5
L 1
L 1
L 1
L 1
L 1
L 1
CPU
CPU
CPU
CPU
CPU
CPU
PR/SM
z/OS LPAR A
z/OS LPAR B
z/OS LPAR C

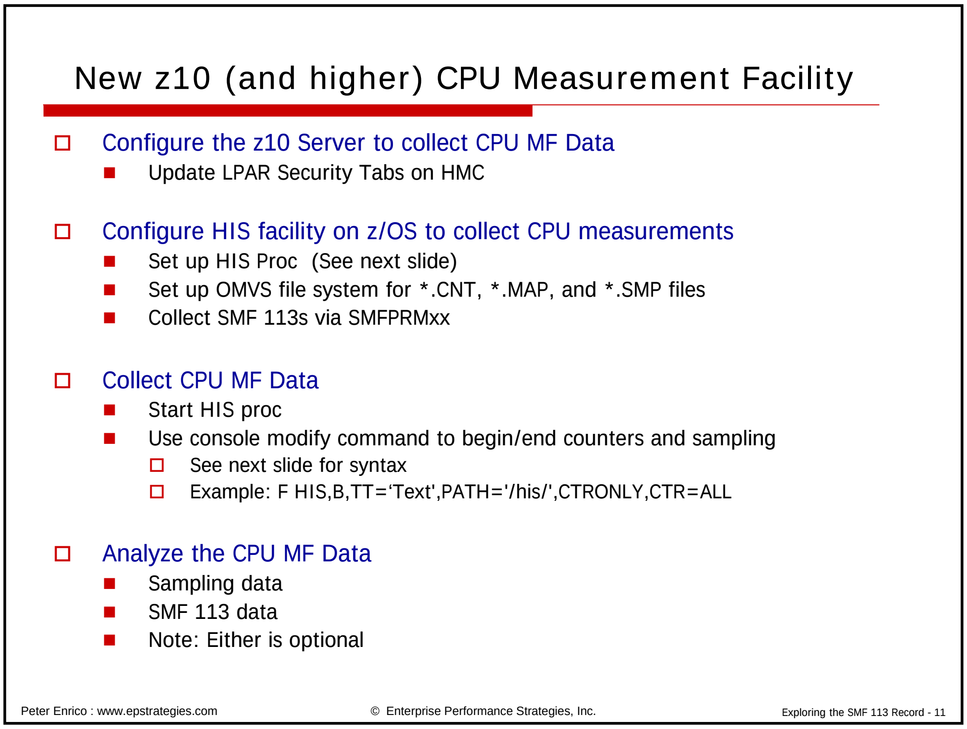
Greatest Usage of SMF 113

- □ Used to illustrate the workings of the z10 (and higher) processors to colleagues and management to assuage concerns about enabling facilities, such as HiperDispatch
- □ Usage of standard SMF records still required for full processor evaluations
  - ■ SMF 30
  - ■ SMF 70
  - ■ SMF 72.3
  - ■ Etc..



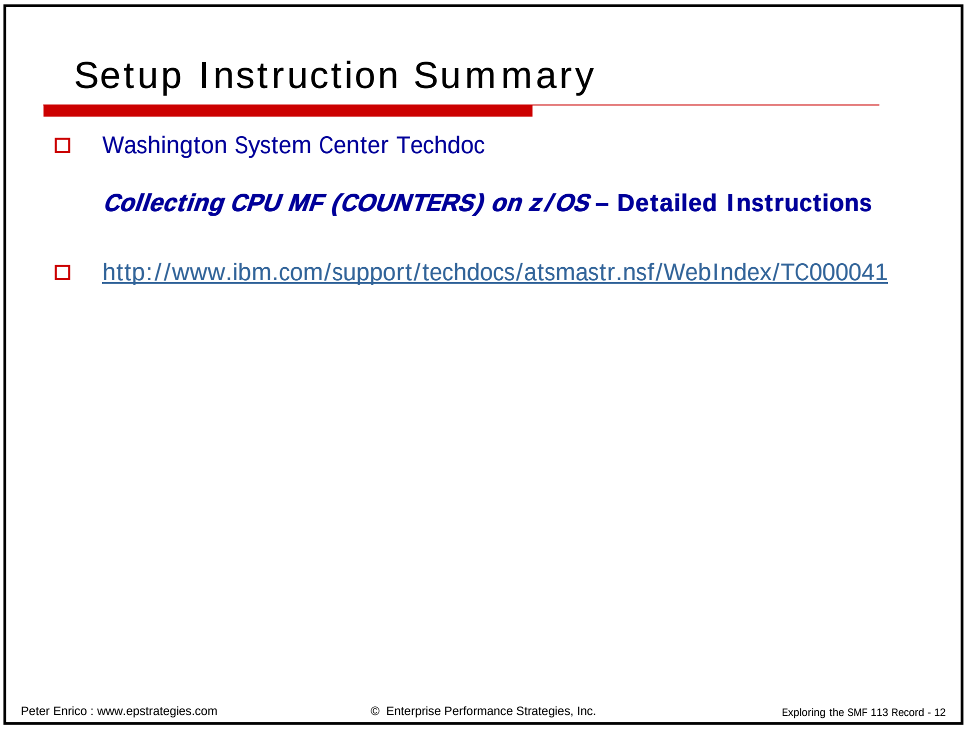
Possible Evaluations

- □ Changes due to hardware configuration
  - ■ Machine type, installed CPUs, memory sizes, book configuration
- □ Understanding workload mixture and characterization
  - ■ IBM plans on using this data to help select the workload mixture as input to zPCR
- □ Changes due to exploitation of certain hardware facilities
  - ■ Such as HiperDispatch and DCM, and any new or changed facility
- □ Changes due to LPAR configuration changes
  - ■ Weights, number of logical processors, number of LPARs, physical to logical, effects of HiperDisptach transitions (i.e. pool changes), etc.
- □ Changes due to WLM management or other software management and tuning algorithms that affect data access and CPU processing
- □ Changes due to changes in workload mixture, applications



New z10 (and higher) CPU Measurement Facility

- □ Configure the z10 Server to collect CPU MF Data
  - ■ Update LPAR Security Tabs on HMC
- □ Configure HIS facility on z/OS to collect CPU measurements
  - ■ Set up HIS Proc (See next slide)
  - ■ Set up OMVS file system for *.CNT, *.MAP, and *.SMP files
  - ■ Collect SMF 113s via SMFPRMxx
- □ Collect CPU MF Data
  - ■ Start HIS proc
  - ■ Use console modify command to begin/end counters and sampling
    - □ See next slide for syntax
    - □ Example: F HIS,B,TT=‘Text',PATH='/his/',CTRONLY,CTR=ALL
- □ Analyze the CPU MF Data
  - ■ Sampling data
  - ■ SMF 113 data
  - ■ Note: Either is optional

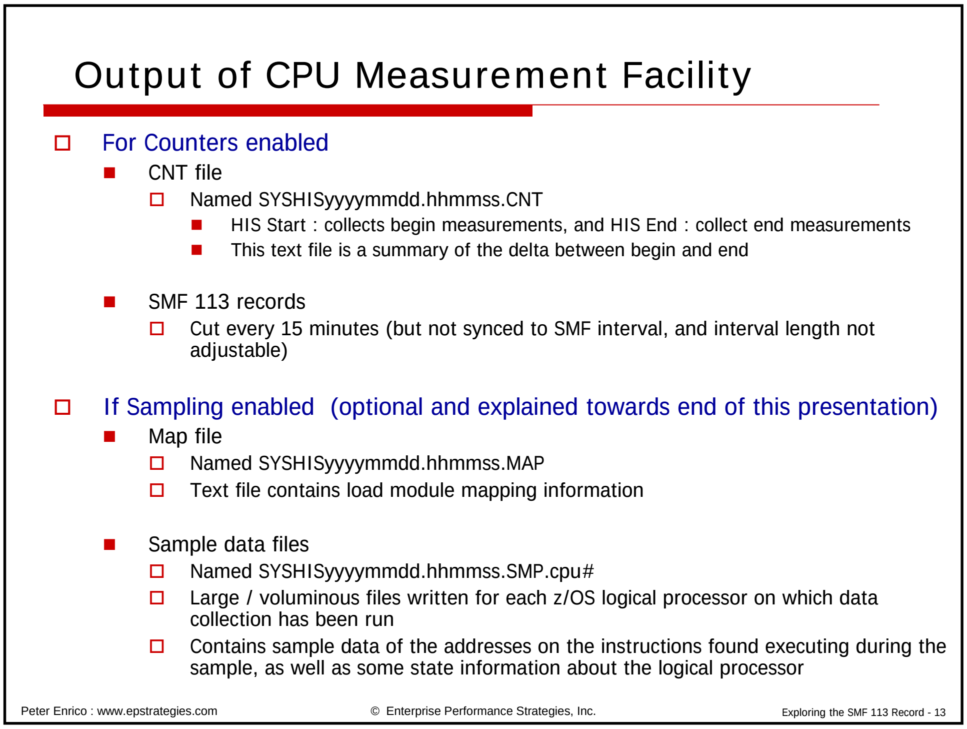


Setup Instruction Summary

- □ Washington System Center Techdoc

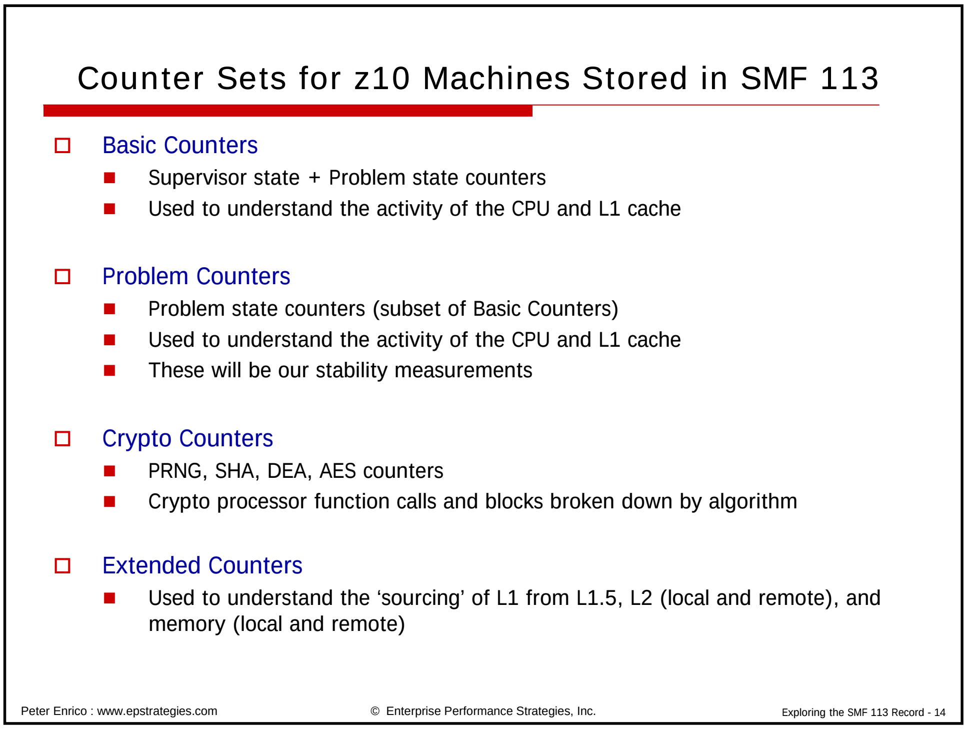
  ***Collecting CPU MF (COUNTERS) on z/OS* – Detailed Instructions**

- □ http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TC000041



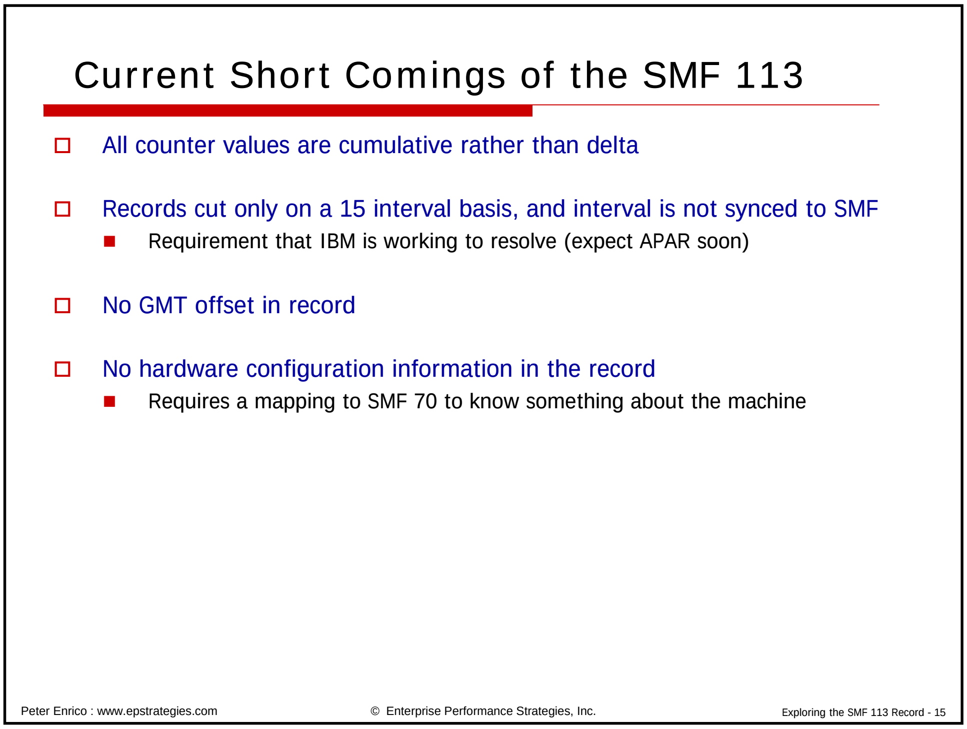
Output of CPU Measurement Facility

- For Counters enabled
  - CNT file
    - Named SYSHISyyyymmdd.hhmmss.CNT
      - HIS Start : collects begin measurements, and HIS End : collect end measurements
      - This text file is a summary of the delta between begin and end
  - SMF 113 records
    - Cut every 15 minutes (but not synced to SMF interval, and interval length not adjustable)
- If Sampling enabled (optional and explained towards end of this presentation)
  - Map file
    - Named SYSHISyyyymmdd.hhmmss.MAP
    - Text file contains load module mapping information
  - Sample data files
    - Named SYSHISyyyymmdd.hhmmss.SMP.cpu#
    - Large / voluminous files written for each z/OS logical processor on which data collection has been run
    - Contains sample data of the addresses on the instructions found executing during the sample, as well as some state information about the logical processor



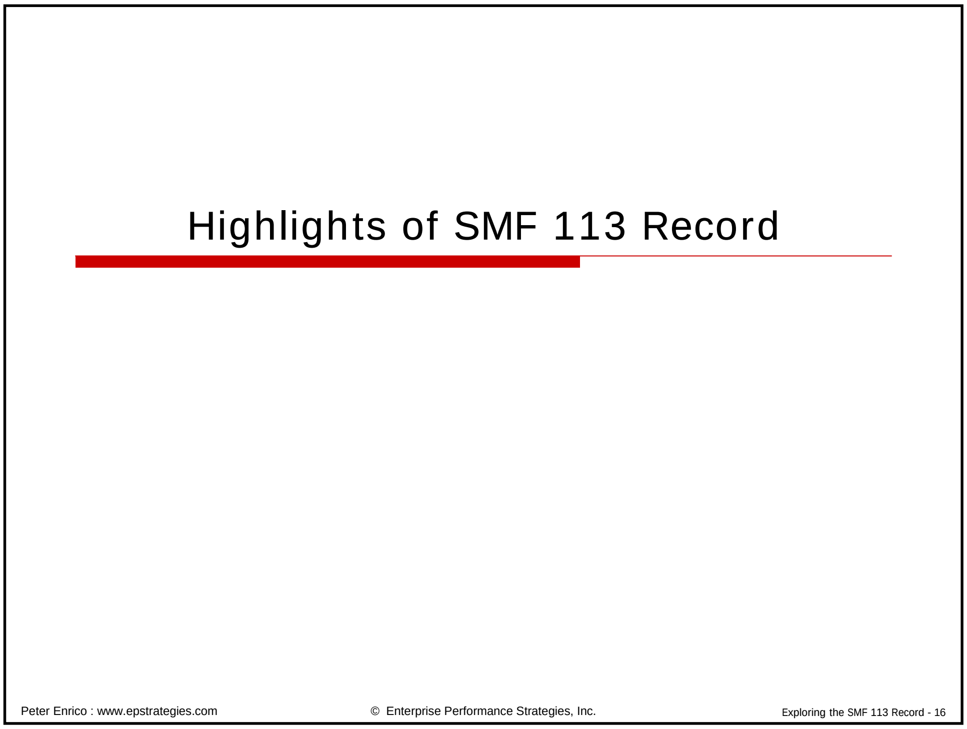
Counter Sets for z10 Machines Stored in SMF 113

- Basic Counters
  - Supervisor state + Problem state counters
  - Used to understand the activity of the CPU and L1 cache
- Problem Counters
  - Problem state counters (subset of Basic Counters)
  - Used to understand the activity of the CPU and L1 cache
  - These will be our stability measurements
- Crypto Counters
  - PRNG, SHA, DEA, AES counters
  - Crypto processor function calls and blocks broken down by algorithm
- Extended Counters
  - Used to understand the ‘sourcing’ of L1 from L1.5, L2 (local and remote), and memory (local and remote)

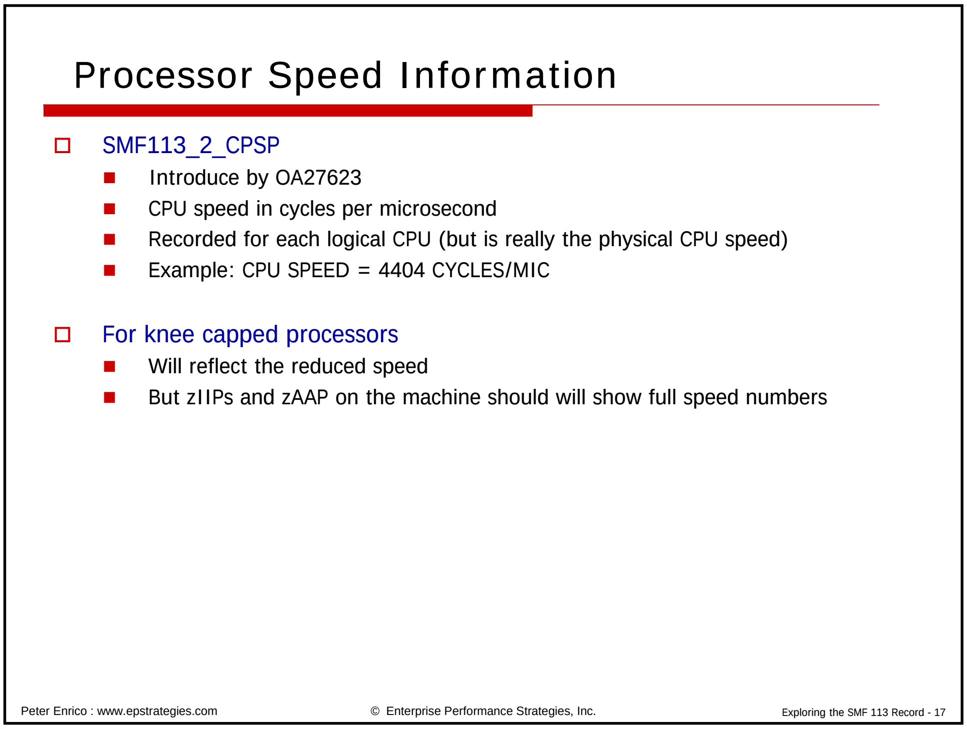


Current Short Comings of the SMF 113

- □ All counter values are cumulative rather than delta
- □ Records cut only on a 15 interval basis, and interval is not synced to SMF
  - ■ Requirement that IBM is working to resolve (expect APAR soon)
- □ No GMT offset in record
- □ No hardware configuration information in the record
  - ■ Requires a mapping to SMF 70 to know something about the machine

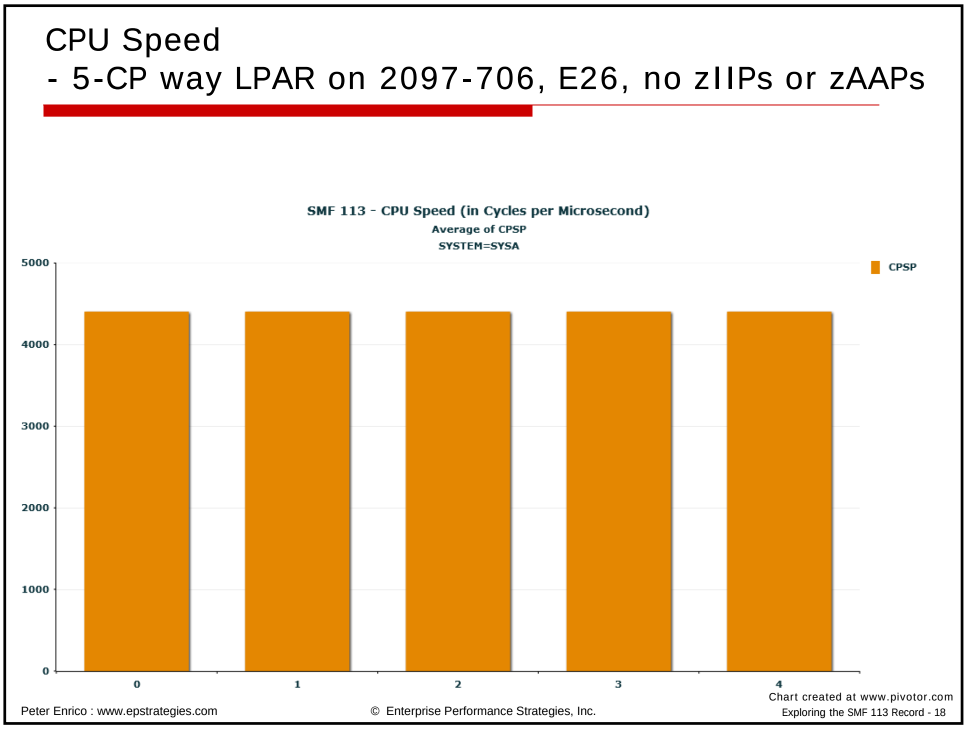


Highlights of SMF 113 Record



Processor Speed Information

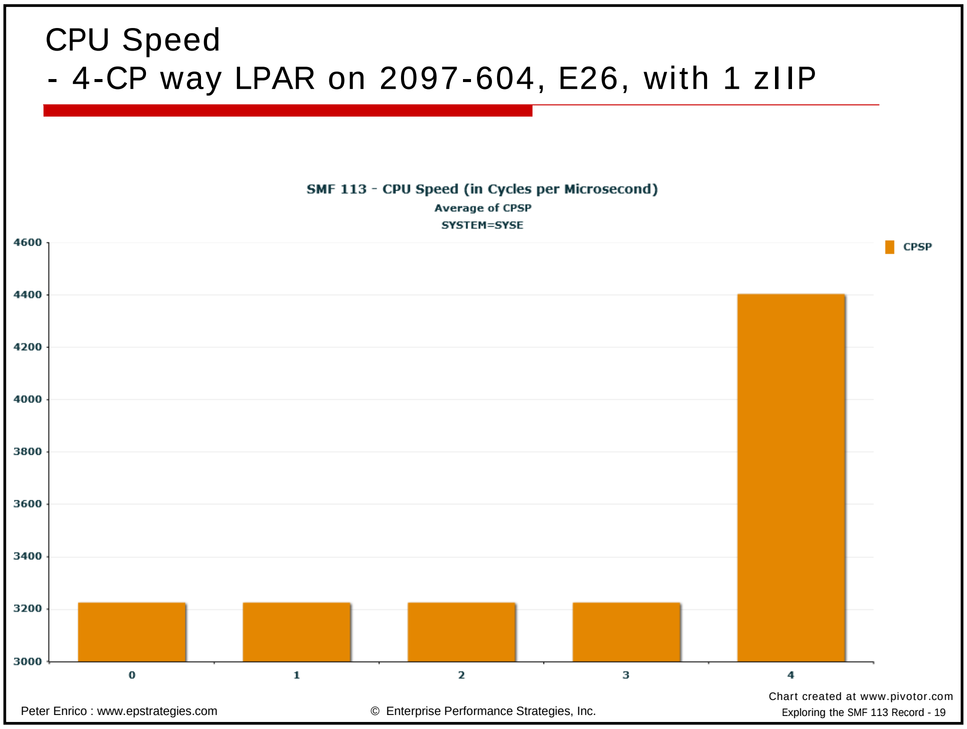
- SMF113_2_CPSP
  - Introduce by OA27623
  - CPU speed in cycles per microsecond
  - Recorded for each logical CPU (but is really the physical CPU speed)
  - Example: CPU SPEED = 4404 CYCLES/MIC
- For knee capped processors
  - Will reflect the reduced speed
  - But zIIPs and zAAP on the machine should will show full speed numbers



CPU Speed
- 5-CP way LPAR on 2097-706, E26, no zIIPs or zAAPs

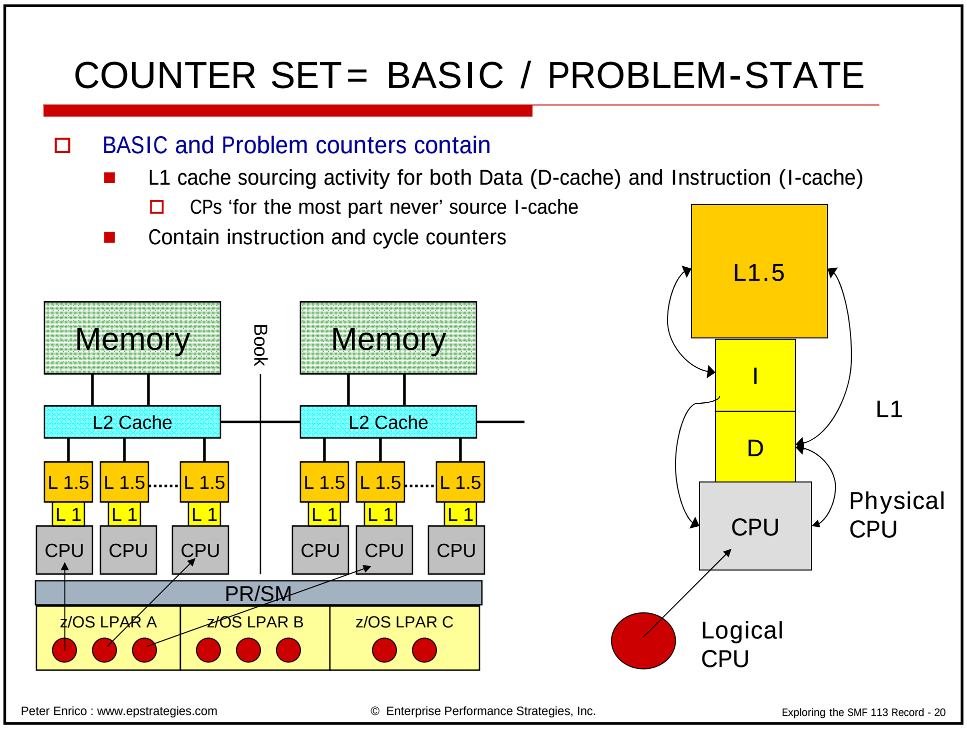
SMF 113 - CPU Speed (in Cycles per Microsecond)
Average of CPSP
SYSTEM=SYSA

Chart created at www.pivotor.com



CPU Speed
- 4-CP way LPAR on 2097-604, E26, with 1 zIIP

Peter Enrico : www.epstrategies.com © Enterprise Performance Strategies, Inc. Exploring the SMF 113 Record - 19



COUNTER SET= BASIC / PROBLEM-STATE

- □ BASIC and Problem counters contain
  - ■ L1 cache sourcing activity for both Data (D-cache) and Instruction (I-cache)
    - □ CPs 'for the most part never' source I-cache
  - ■ Contain instruction and cycle counters

Peter Enrico : www.epstrategies.com © Enterprise Performance Strategies, Inc. Exploring the SMF 113 Record - 20

COUNTER SET= BASIC

- Activity count for CPU when in both problem and supervisor state
  - Counters for general purpose processors, zIIPs, and zAAPs
- 0: CYCLE COUNT
  - Number of CPU cycles, excluding the number of cycles CPU is in wait state
- 1: INSTRUCTION COUNT
  - Number of supervisor and problem state instructions executed by the CPU
- 2: L1 I-CACHE DIRECTORY-WRITE COUNT
  - Number of writes to instruction cache (and includes data cache if unified cache)
- 3: L1 I-CACHE PENALTY CYCLE COUNT
  - Instruction cache penalty cycle count (and includes data cache if unified cache)
- 4: L1 D-CACHE DIRECTORY-WRITE COUNT
  - Number of writes to data cache (and zero if unified cache)
- 5: L1 D-CACHE PENALTY CYCLE COUNT
  - Data cache penalty cycle count (and zero if unified cache)

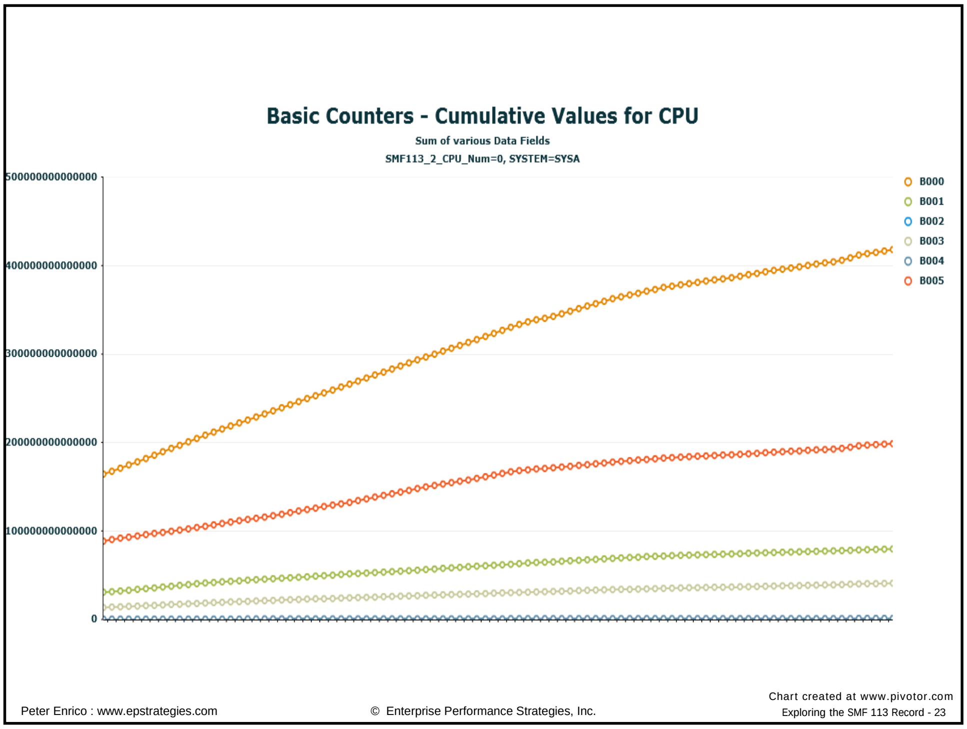
COUNTER SET= BASIC *.CNT file excerpt example

- CNT file contains delta values from begin to end of HIS modify begin/end
  - Note a very easy report to use, so it is recommended to use SMF 113 since will get measurements on an interval basis rather than begin/end delta

```
COUNTER SET= BASIC
COUNTER IDENTIFIERS:
   0: CYCLE COUNT
   1: INSTRUCTION COUNT
   2: L1 I-CACHE DIRECTORY-WRITE COUNT
   3: L1 I-CACHE PENALTY CYCLE COUNT
   4: L1 D-CACHE DIRECTORY-WRITE COUNT
   5: L1 D-CACHE PENALTY CYCLE COUNT

START TIME: 2010/03/16 11:25:21  START TOD: C5AFCE3D7E54909C
END TIME:   2010/03/16 14:44:15  END TOD:   C5AFFAB2674B2F8C
COUNTER VALUES (HEXADECIMAL) FOR CPU 00 (CPU SPEED = 4404 CYCLES/MIC):
  0-  3 000007316AE25823 000000BE06CB4472 00000003CA983E8B 000001DE4B1B3543
  4-  7 00000004C2AE05DC 000003E785375A3C ---------------- ----------------

START TIME: 2010/03/16 11:25:21  START TOD: C5AFCE3D7E586F9C
END TIME:   2010/03/16 14:44:15  END TOD:   C5AFFAB2674C708C
COUNTER VALUES (HEXADECIMAL) FOR CPU 01 (CPU SPEED = 4404 CYCLES/MIC):
  0-  3 0000072B802120E8 000000B9C8DD0373 00000003CCC89351 000001E6B45F71C7
  4-  7 00000004BA302723 000003E608E79159 ---------------- ----------------
```

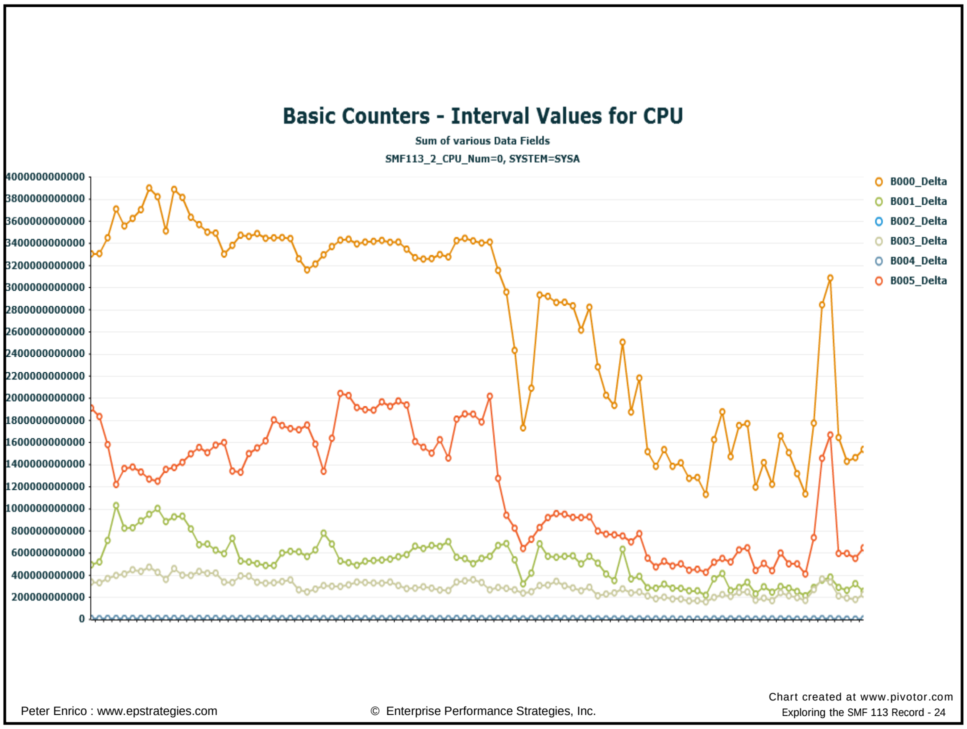


Basic Counters - Cumulative Values for CPU

Sum of various Data Fields

SMF113_2_CPU_Num=0, SYSTEM=SYSA

Chart created at www.pivotor.com

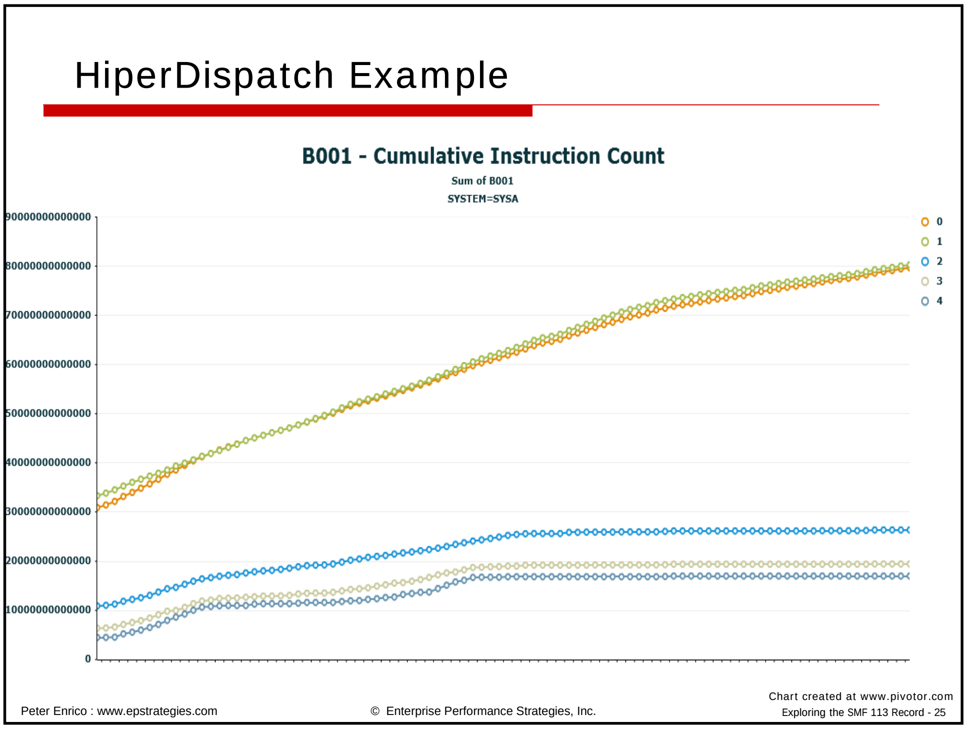


Basic Counters - Interval Values for CPU

Sum of various Data Fields

SMF113_2_CPU_Num=0, SYSTEM=SYSA

Chart created at www.pivotor.com



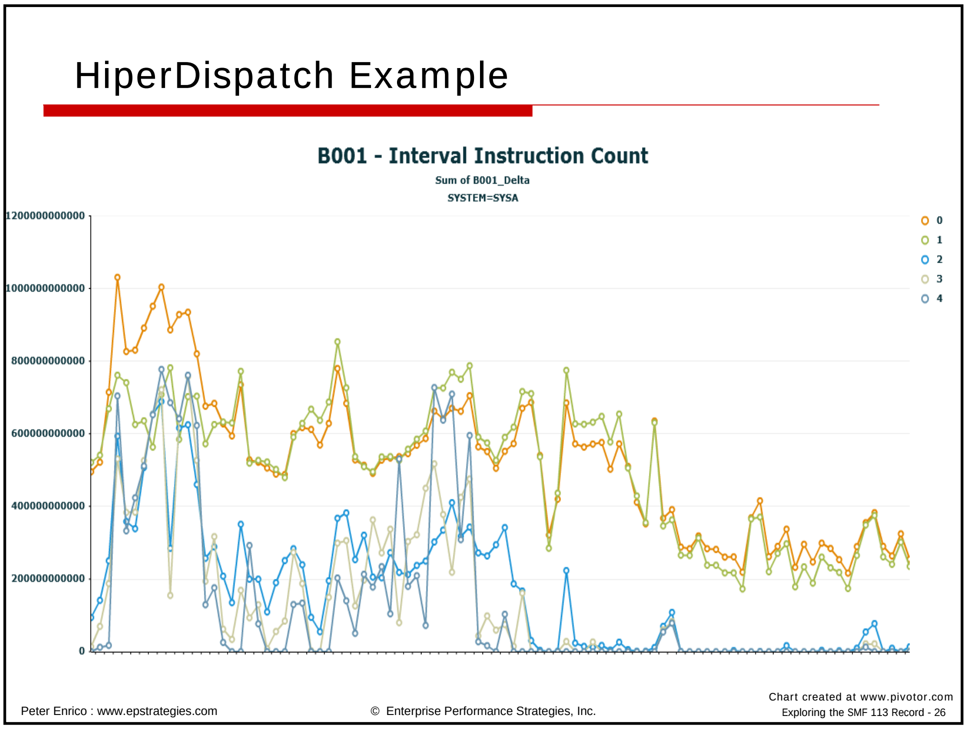
HiperDispatch Example

Chart created at www.pivotor.com

Peter Enrico : www.epstrategies.com

© Enterprise Performance Strategies, Inc.

Exploring the SMF 113 Record - 25



HiperDispatch Example

Chart created at www.pivotor.com

Peter Enrico : www.epstrategies.com

© Enterprise Performance Strategies, Inc.

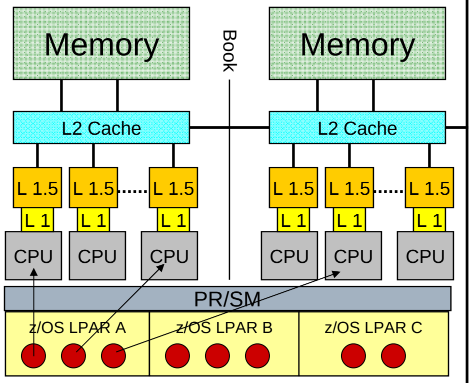
Exploring the SMF 113 Record - 26

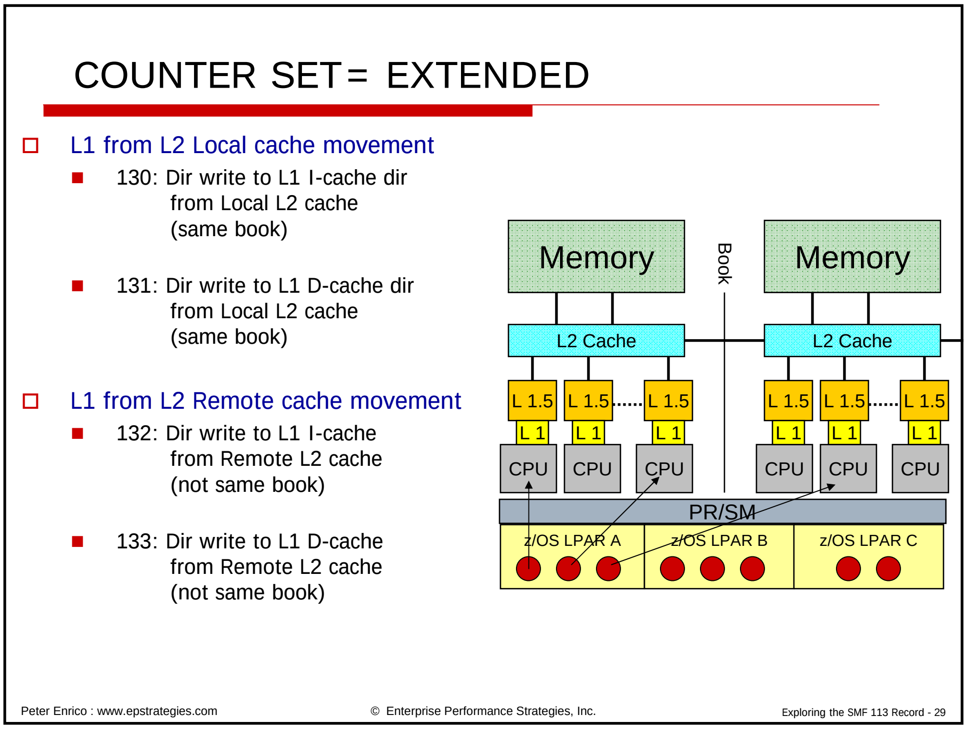
COUNTER SET= PROBLEM-STATE

- Activity count for CPU when in both problem state
  - Counters for general purpose processors, zIIPs, and zAAPs
- 32: PROBLEM-STATE CYCLE COUNT
  - Number of CPU cycles, excluding the number of cycles CPU is in wait state
- 33: PROBLEM-STATE INSTRUCTION COUNT
  - Number of problem state instructions executed by the CPU
- 34: PROBLEM-STATE L1 I-CACHE DIRECTORY-WRITE COUNT
  - Number of writes to instruction cache (and includes data cache if unified cache)
- 35: PROBLEM-STATE L1 I-CACHE PENALTY CYCLE COUNT
  - Instruction cache penalty cycle count (and includes data cache if unified cache)
- 36: PROBLEM-STATE L1 D-CACHE DIRECTORY-WRITE COUNT
  - Number of writes to data cache (and zero if unified cache)
- 37: PROBLEM-STATE L1 D-CACHE PENALTY CYCLE COUNT
  - Data cache penalty cycle count (and zero if unified cache)

COUNTER SET= EXTENDED

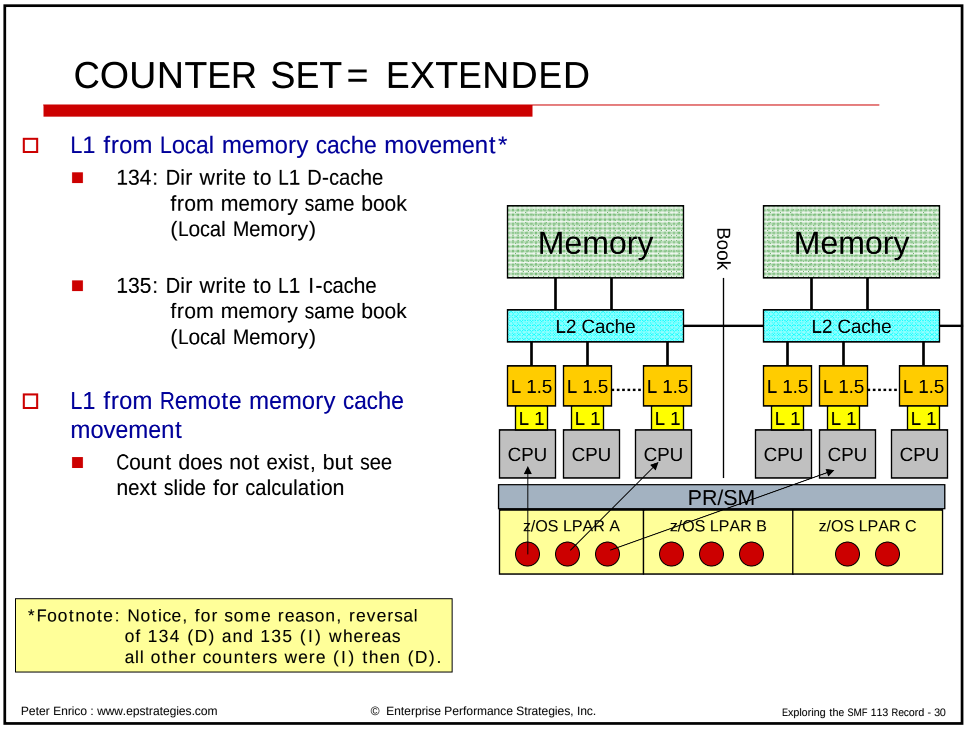
- L1 from L1.5 cache movement
  - 128: Dir write to L1 I-cache dir from L1.5 cache (Instruction)
  - 129: Dir write to L1 D-cache dir from L1.5 cache (Data)





COUNTER SET= EXTENDED

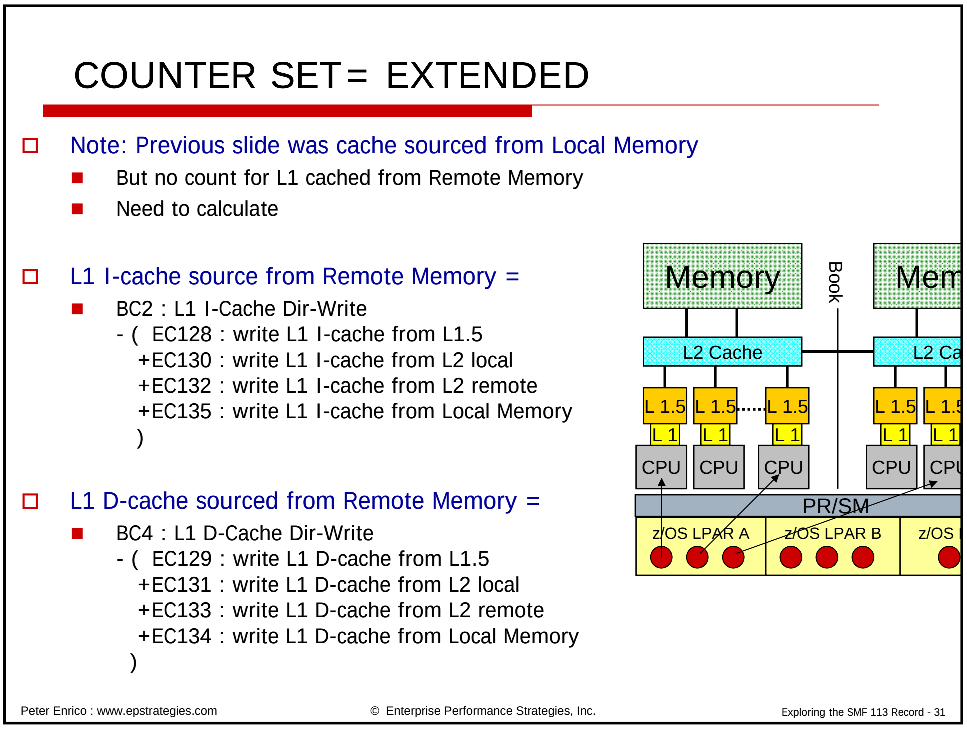
- □ L1 from L2 Local cache movement
  - 130: Dir write to L1 I-cache dir from Local L2 cache (same book)
  - 131: Dir write to L1 D-cache dir from Local L2 cache (same book)
- □ L1 from L2 Remote cache movement
  - 132: Dir write to L1 I-cache from Remote L2 cache (not same book)
  - 133: Dir write to L1 D-cache from Remote L2 cache (not same book)



COUNTER SET= EXTENDED

- □ L1 from Local memory cache movement*
  - 134: Dir write to L1 D-cache from memory same book (Local Memory)
  - 135: Dir write to L1 I-cache from memory same book (Local Memory)
- □ L1 from Remote memory cache movement
  - Count does not exist, but see next slide for calculation

*Footnote: Notice, for some reason, reversal of 134 (D) and 135 (I) whereas all other counters were (I) then (D).

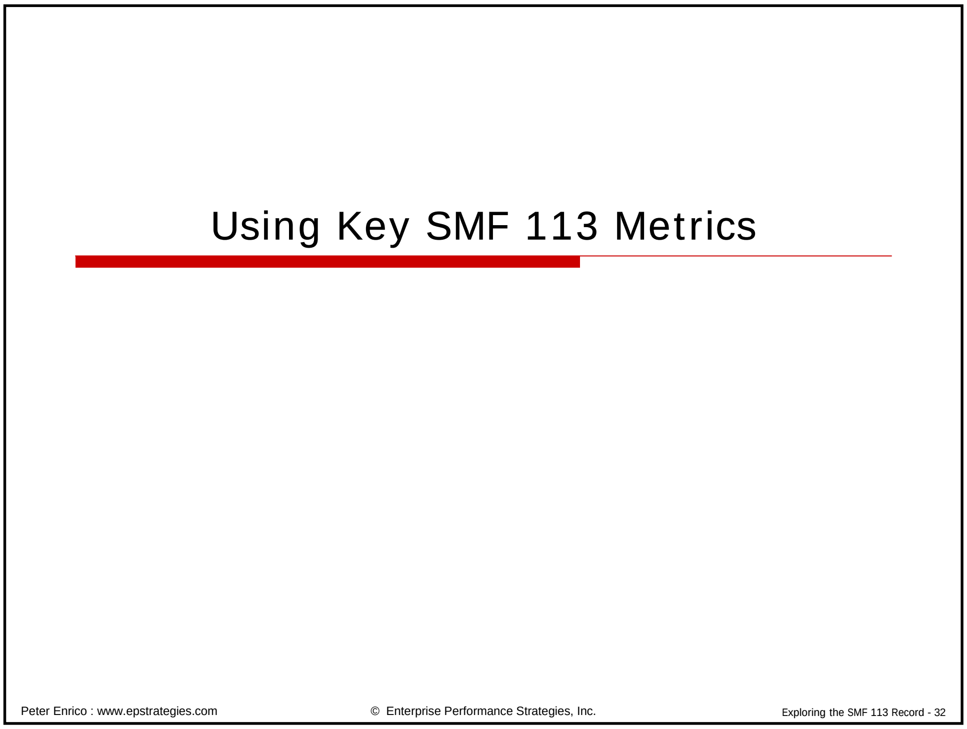


COUNTER SET= EXTENDED

- Note: Previous slide was cache sourced from Local Memory
  - But no count for L1 cached from Remote Memory
  - Need to calculate

- L1 I-cache source from Remote Memory =
  - BC2 : L1 I-Cache Dir-Write
    - (EC128 : write L1 I-cache from L1.5
    +EC130 : write L1 I-cache from L2 local
    +EC132 : write L1 I-cache from L2 remote
    +EC135 : write L1 I-cache from Local Memory
)

- L1 D-cache sourced from Remote Memory =
  - BC4 : L1 D-Cache Dir-Write
    - (EC129 : write L1 D-cache from L1.5
    +EC131 : write L1 D-cache from L2 local
    +EC133 : write L1 D-cache from L2 remote
    +EC134 : write L1 D-cache from Local Memory
)



Using Key SMF 113 Metrics

CPI – Cycles Per Instruction

- □ Key metric to gauge processor contention
  - ■ Useful when doing a before / after comparison
  - ■ Over time, useful to understand instruction mixture consistency
- □ When CPI increases
  - ■ It is taking more cycles to execute the instruction mix
  - ■ Shows an increase in contention
- □ When CPI decreases
  - ■ It is taking less cycles to execute the instruction mix
  - ■ Shows a decrease in contention
- □ Cycles / Instruction
  - ■ Counters needed
    - □ BC0: Cycle Count
    - □ BC1: Instruction Count

CPI = (Cycles / Instructions)

HD Before/After Example – Small MP

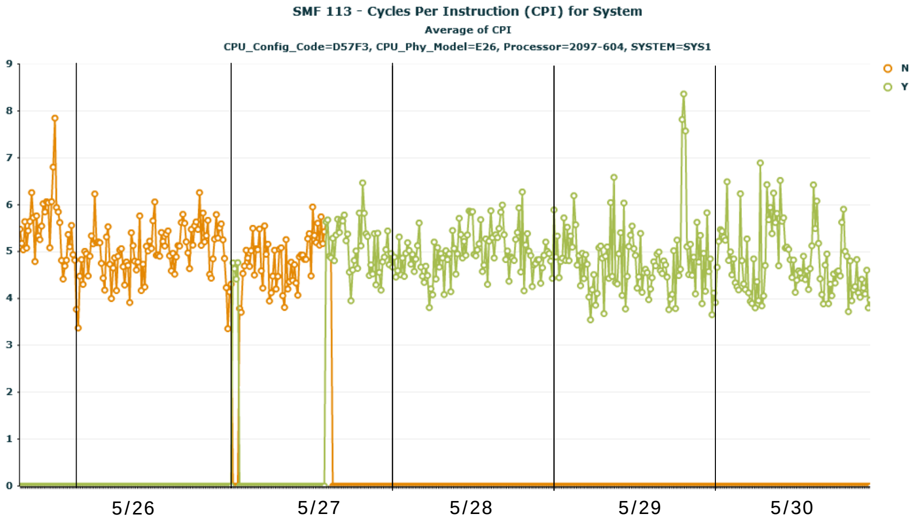
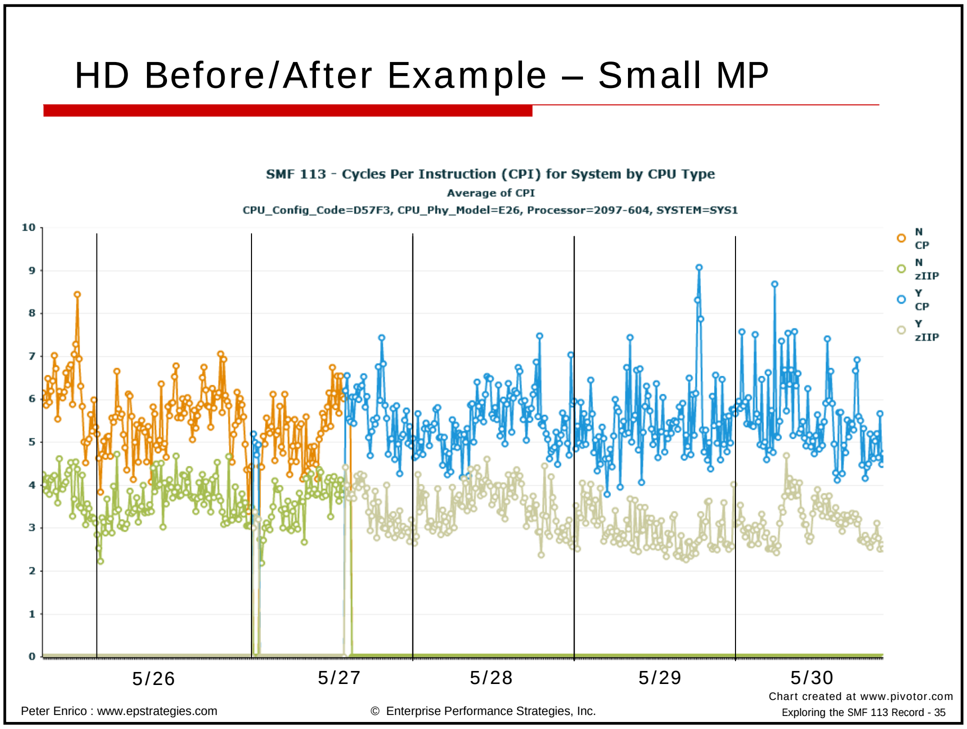


Chart created at www.pivotor.com

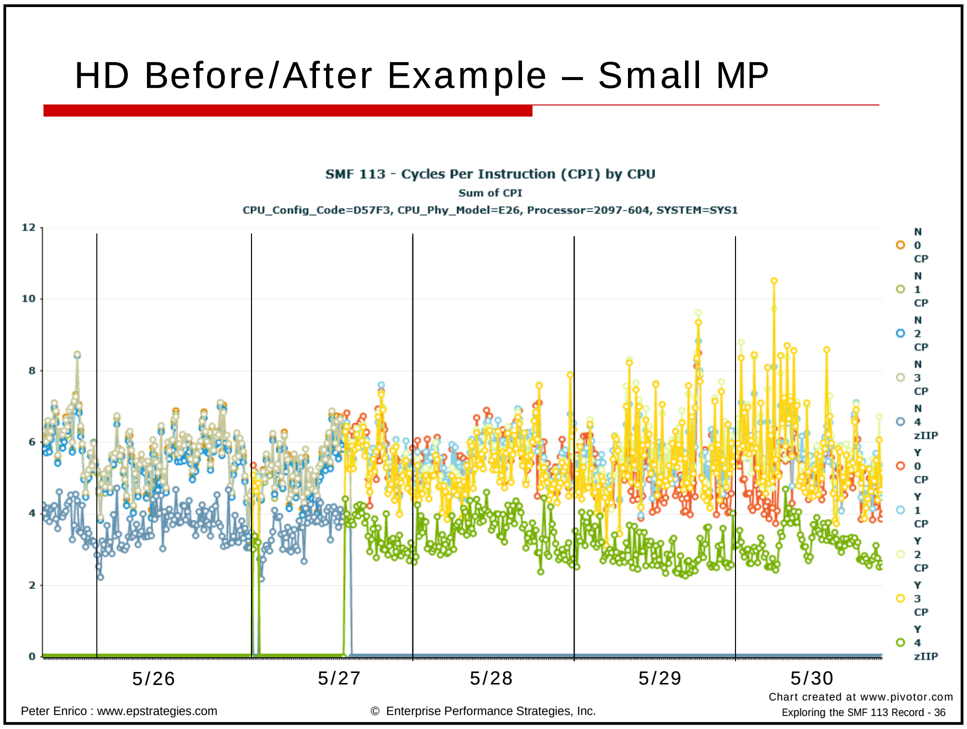


HD Before/After Example – Small MP

SMF 113 - Cycles Per Instruction (CPI) for System by CPU Type
Average of CPI
CPU_Config_Code=D57F3, CPU_Phy_Model=E26, Processor=2097-604, SYSTEM=SYS1

Chart created at www.pivotor.com

Peter Enrico : www.epstrategies.com © Enterprise Performance Strategies, Inc. Exploring the SMF 113 Record - 35



HD Before/After Example – Small MP

SMF 113 - Cycles Per Instruction (CPI) by CPU
Sum of CPI
CPU_Config_Code=D57F3, CPU_Phy_Model=E26, Processor=2097-604, SYSTEM=SYS1

Chart created at www.pivotor.com

Peter Enrico : www.epstrategies.com © Enterprise Performance Strategies, Inc. Exploring the SMF 113 Record - 36

HD Before/After Example – Small MP

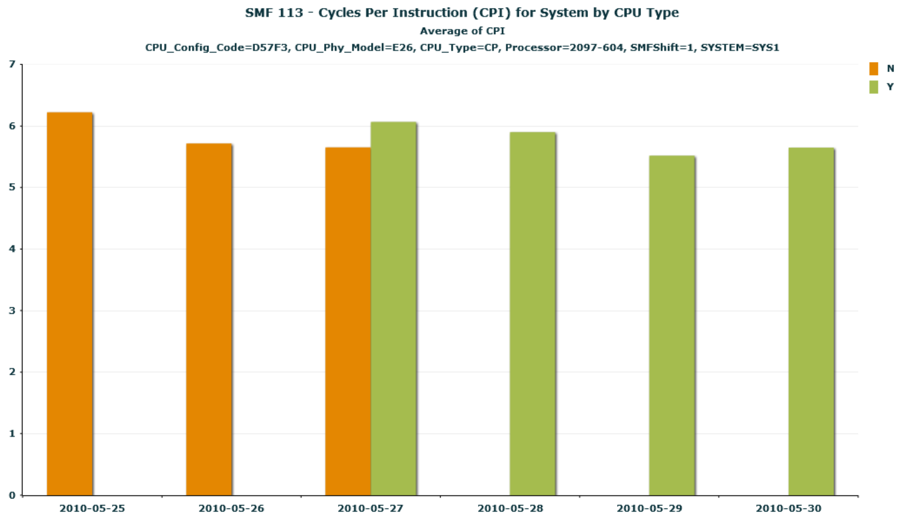


Chart created at www.pivotor.com

LPARCPU – LPAR Physical Busy %

- □ LPAR CPU
  - LPARCPU (Cycle CPU %) (based on Cycle CPU seconds captured and un-captured)
  - Counters needed
    - □ Processor Speed (cycles per microsecond) = SMF113_2_CPSP
    - □ BC0: Cycle Count

*LPARCPU = (((1/CPSP/1,000,000) * B0) / Interval Seconds) * 100*

  - Example: Say for CPU0
    - □ If (Speed = 4404 Cycles/microsecond) &
    - □ (Executed 3,305,217,446,122 cycles executed in 900 seconds)
    - □ Then CPU utilization of CPU0 = 83.4%
  - Add for each CPU to get utilization as a percent of 1 CPU
  - Or Average for all CPUs to get a LPAR Counter Utilization %

HD Before/After Example – Small MP

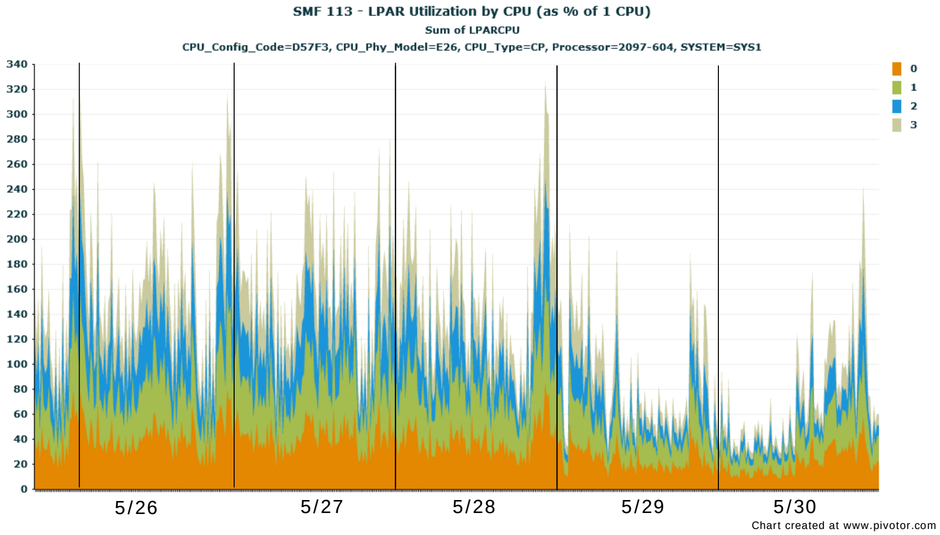


Chart created at www.pivotor.com

HiperDispatch Example

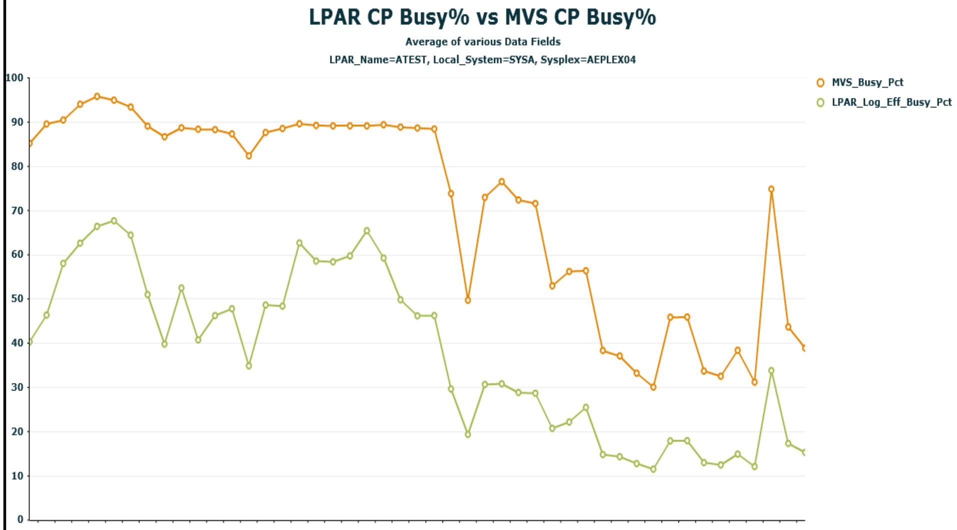


Chart created at www.pivotor.com

PRBSTATE (Problem Instruction to Total Instruction)

□ Problem to Total Instruction Ratio

- Ratio of Problem State instructions to Total instructions

- Counters needed
  - □ BC1 = (Supervisor State Instructions + Problem State Instructions)
  - □ PC33 = (Problem State Instructions)

PRBSTATE = PC33/ BC1

□ This is our stability factor

- Useful to determine if the before / after workload is consistent

HD Before/After Example – Small MP

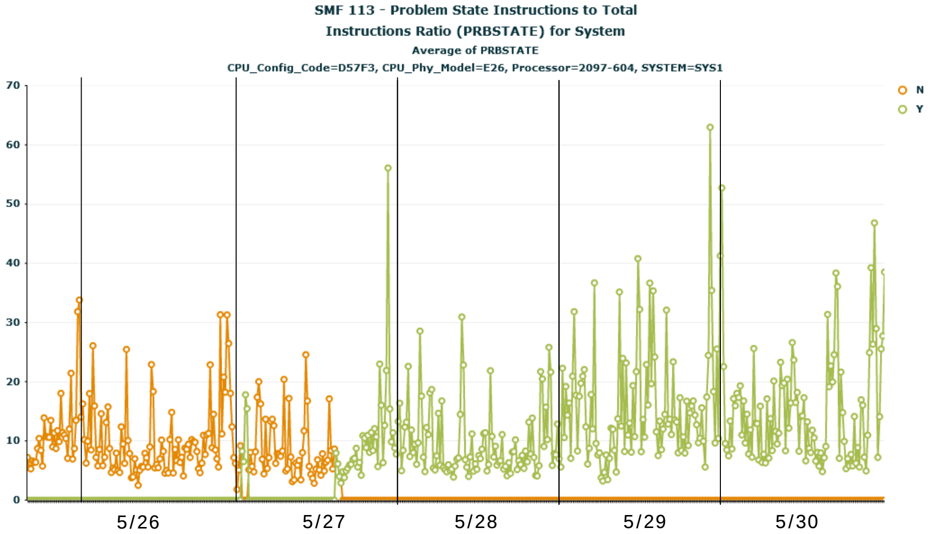
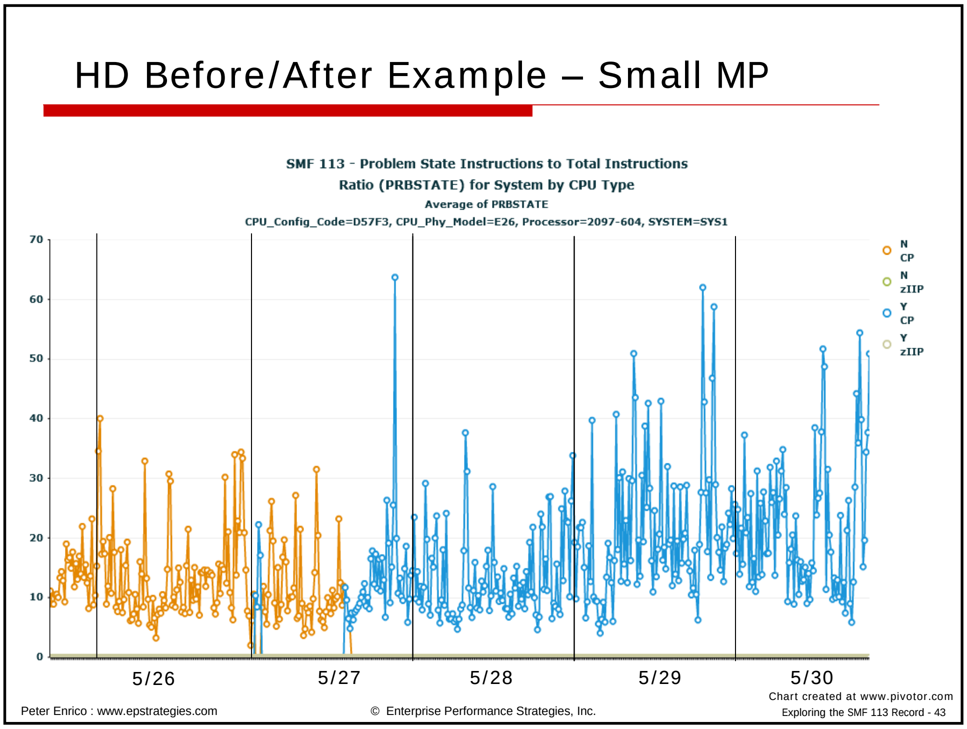


Chart created at www.pivotor.com



HD Before/After Example – Small MP

SMF 113 - Problem State Instructions to Total Instructions Ratio (PRBSTATE) for System by CPU Type

Average of PRBSTATE

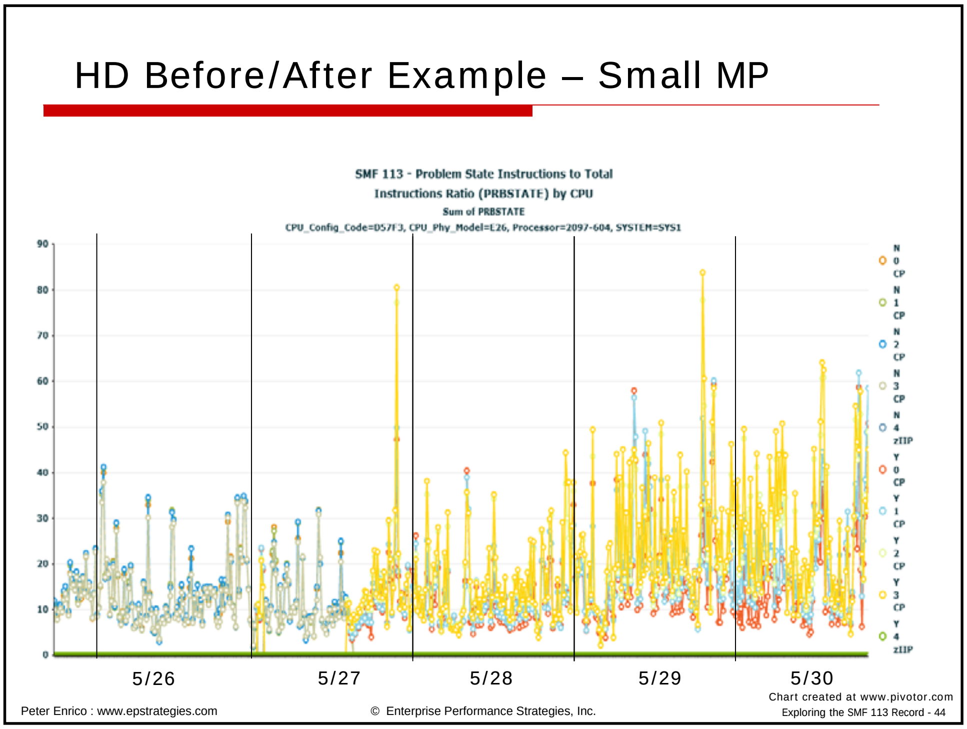
CPU_Config_Code=D57F3, CPU_Phy_Model=E26, Processor=2097-604, SYSTEM=SYS1

Chart created at www.pivotor.com

Peter Enrico : www.epstrategies.com

© Enterprise Performance Strategies, Inc.

Exploring the SMF 113 Record - 43



HD Before/After Example – Small MP

SMF 113 - Problem State Instructions to Total Instructions Ratio (PRBSTATE) by CPU

Sum of PRBSTATE

CPU_Config_Code=D57F3, CPU_Phy_Model=E26, Processor=2097-604, SYSTEM=SYS1

Chart created at www.pivotor.com

Peter Enrico : www.epstrategies.com

© Enterprise Performance Strategies, Inc.

Exploring the SMF 113 Record - 44

HD Before/After Example – Small MP

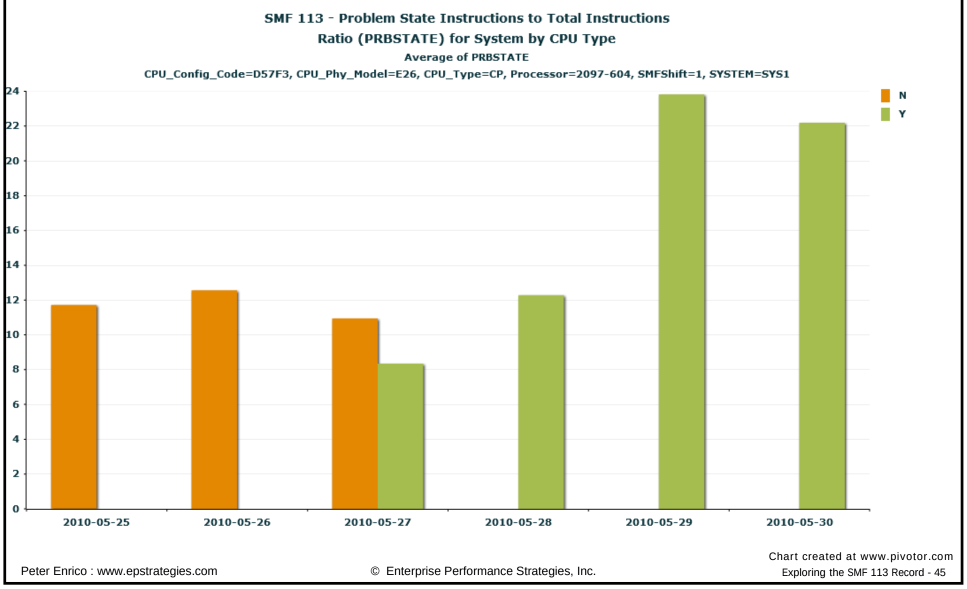


Chart created at www.pivotor.com

Useful Formulas

- Executed Instructions Rate (in Million Instructions per Second)
  - This is really the inverse of the CPI number (cycles per instruction)
  - So recommend using CPI to compare changes rather than this MIPS number
  - Counters needed
    - BC1: Instruction Count
    - Measurement length in seconds

MIPS = (BC1 / Interval Seconds) / 1,000,000

  - This will not, and is not expected to, match any sort of MIPS table value or MIPS number you are utilizing today
  - This MIPS number has absolutely nothing to do with capacity!

Instruction Rate - HiperDispatch Example

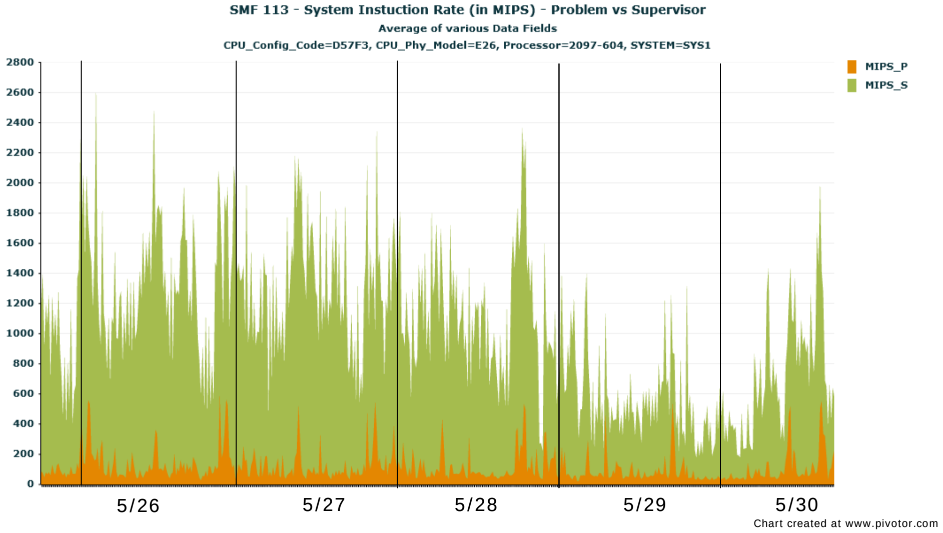
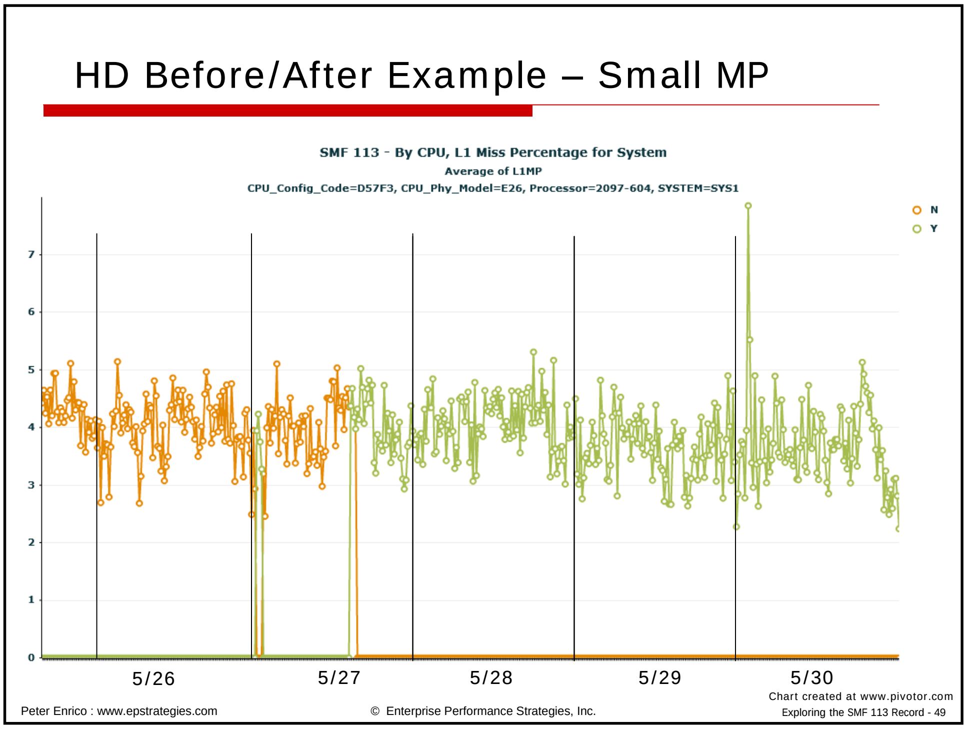


Chart created at www.pivotor.com

L1 Cache Miss %

- L1 Cache Miss %
  - Means percentage of counters when L1 I-cache or L1 D-cache got a cache miss
  - Opposite of the hit percentage
    - Calculate miss rather than hit since the source % numbers (presented in subsequent slides) will be a breakdown of this miss %
  - Based on counters
    - BC2: L1 I-Cache Dir-Write Count
    - BC4: L1 D-Cache Dir-Write Count
    - BC1: INSTRUCTION COUNT

*L1MP= ((BC2 + BC4) / BC1) * 100*



HD Before/After Example – Small MP

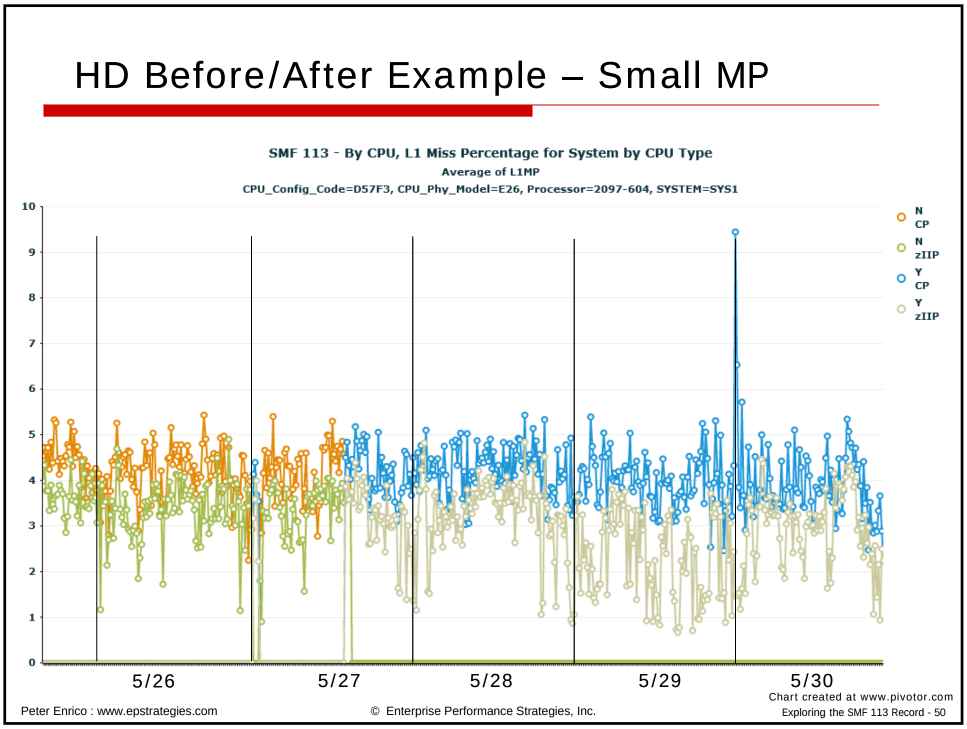
SMF 113 - By CPU, L1 Miss Percentage for System
Average of L1MP
CPU_Config_Code=D57F3, CPU_Phy_Model=E26, Processor=2097-604, SYSTEM=SYS1

Chart created at www.pivotor.com

Peter Enrico : www.epstrategies.com

© Enterprise Performance Strategies, Inc.

Exploring the SMF 113 Record - 49



HD Before/After Example – Small MP

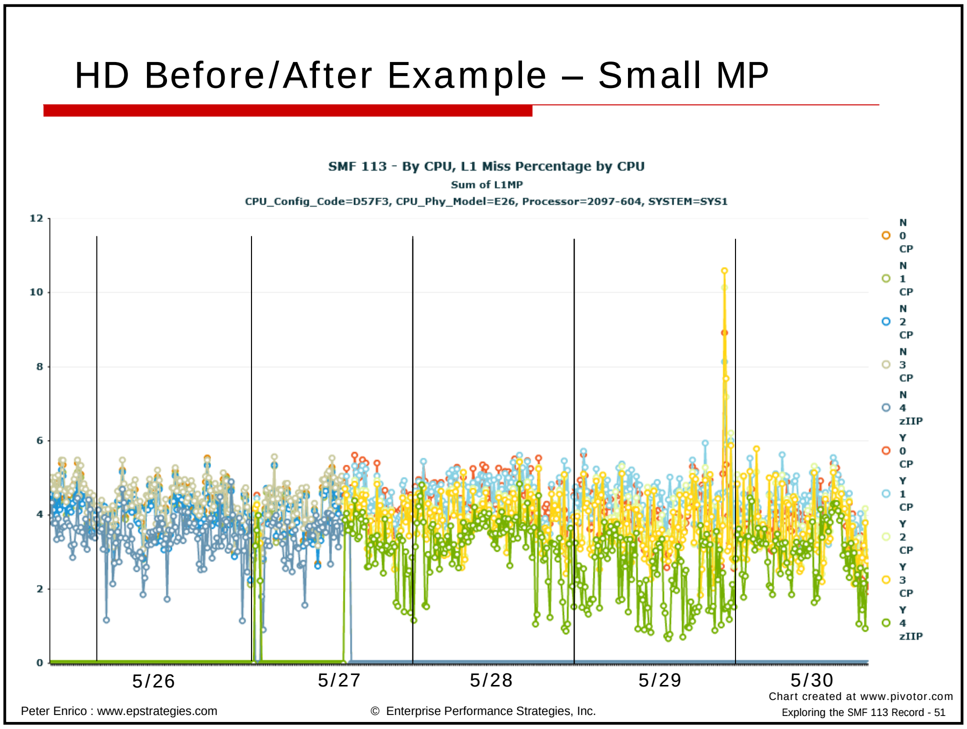
SMF 113 - By CPU, L1 Miss Percentage for System by CPU Type
Average of L1MP
CPU_Config_Code=D57F3, CPU_Phy_Model=E26, Processor=2097-604, SYSTEM=SYS1

Chart created at www.pivotor.com

Peter Enrico : www.epstrategies.com

© Enterprise Performance Strategies, Inc.

Exploring the SMF 113 Record - 50



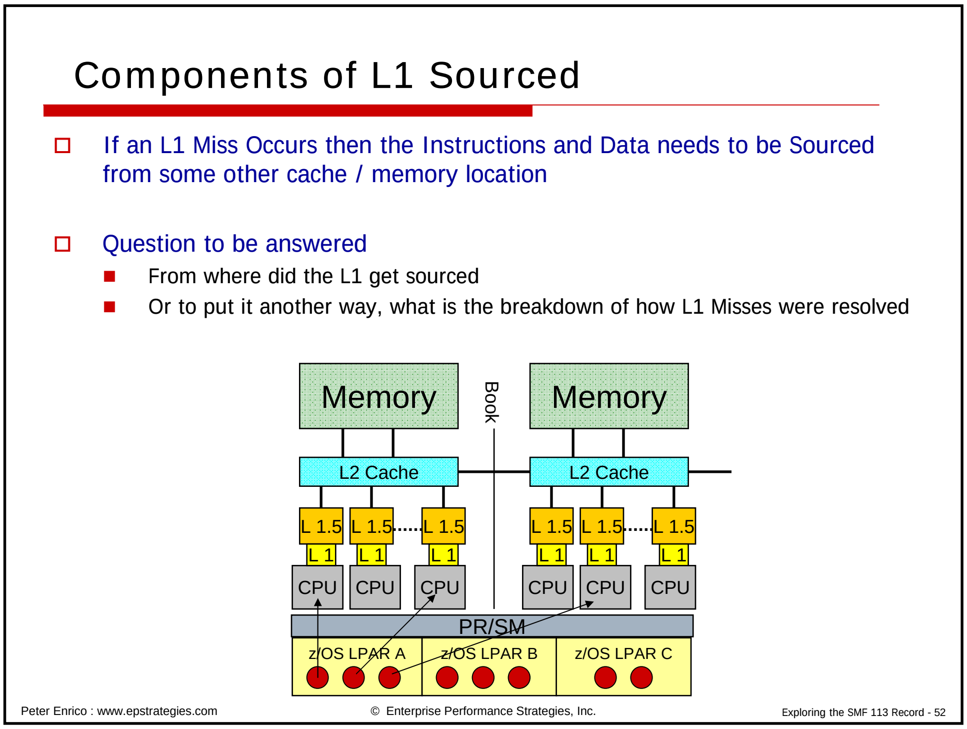
HD Before/After Example – Small MP

SMF 113 - By CPU, L1 Miss Percentage by CPU

Sum of L1MP

CPU_Config_Code=D57F3, CPU_Phy_Model=E26, Processor=2097-604, SYSTEM=SYS1

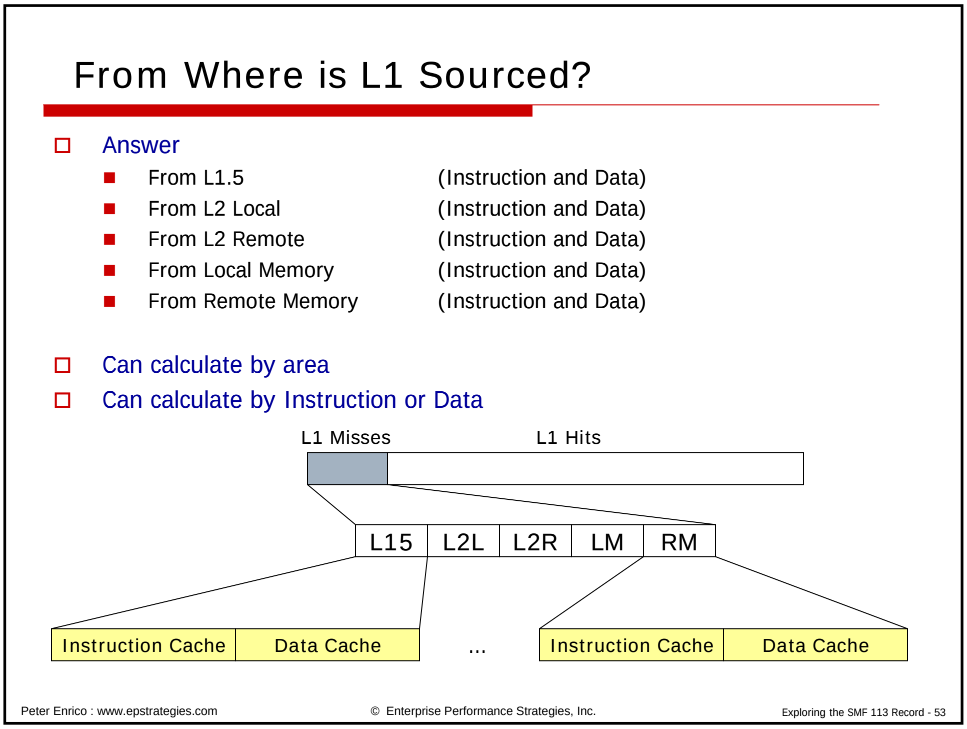
Chart created at www.pivotor.com



Components of L1 Sourced

- □ If an L1 Miss Occurs then the Instructions and Data needs to be Sourced from some other cache / memory location

- □ Question to be answered
  - ■ From where did the L1 get sourced
  - ■ Or to put it another way, what is the breakdown of how L1 Misses were resolved



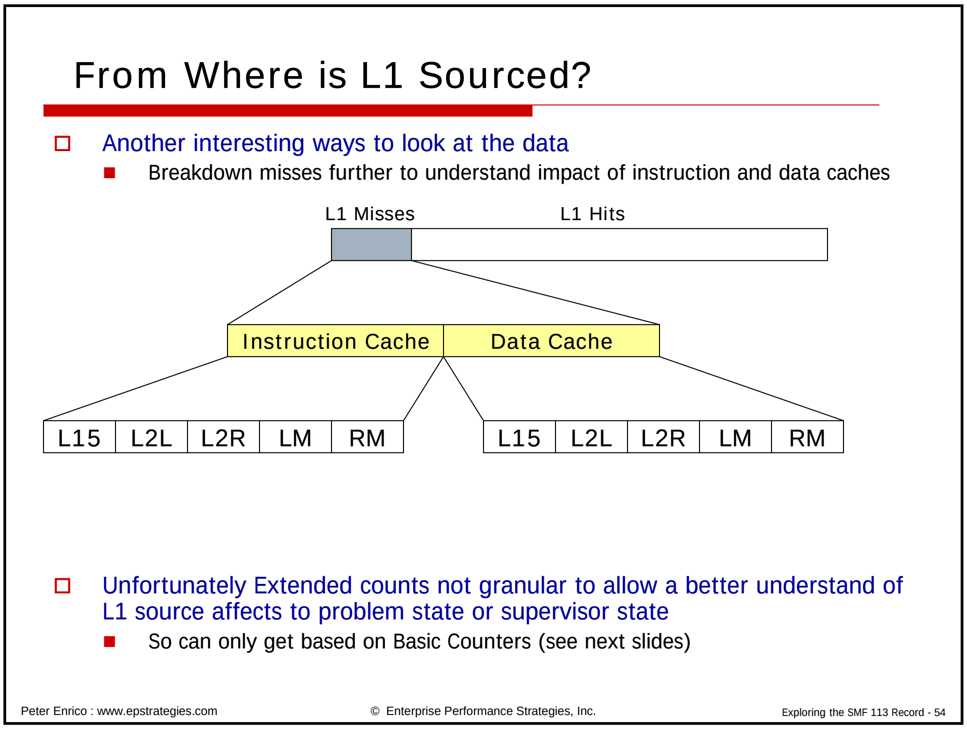
From Where is L1 Sourced?

- □ Answer

■ From L1.5	(Instruction and Data)
■ From L2 Local	(Instruction and Data)
■ From L2 Remote	(Instruction and Data)
■ From Local Memory	(Instruction and Data)
■ From Remote Memory	(Instruction and Data)

- □ Can calculate by area
- □ Can calculate by Instruction or Data

L1 Misses
L1 Hits
L15 | L2L | L2R | LM | RM
Instruction Cache | Data Cache
…
Instruction Cache | Data Cache



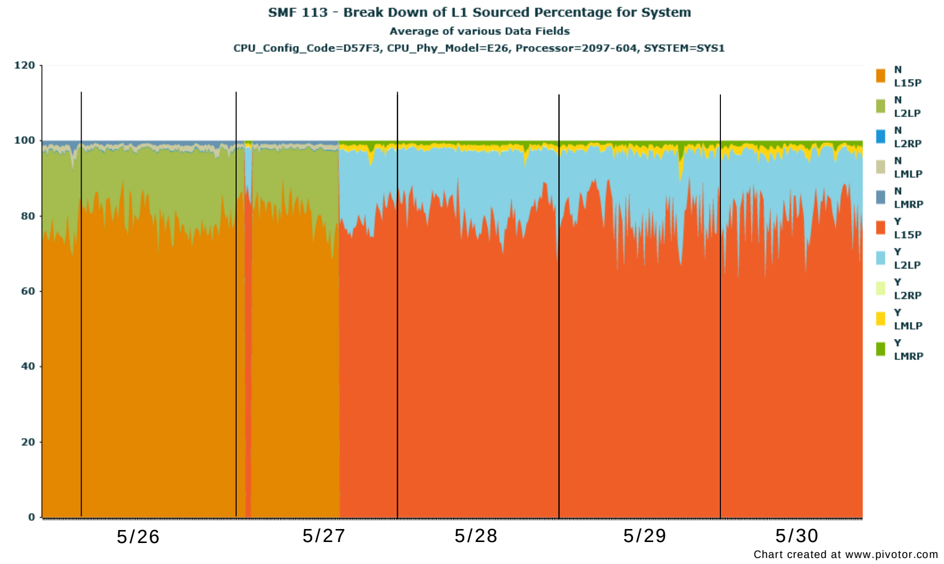
From Where is L1 Sourced?

- □ Another interesting ways to look at the data
  - ■ Breakdown misses further to understand impact of instruction and data caches

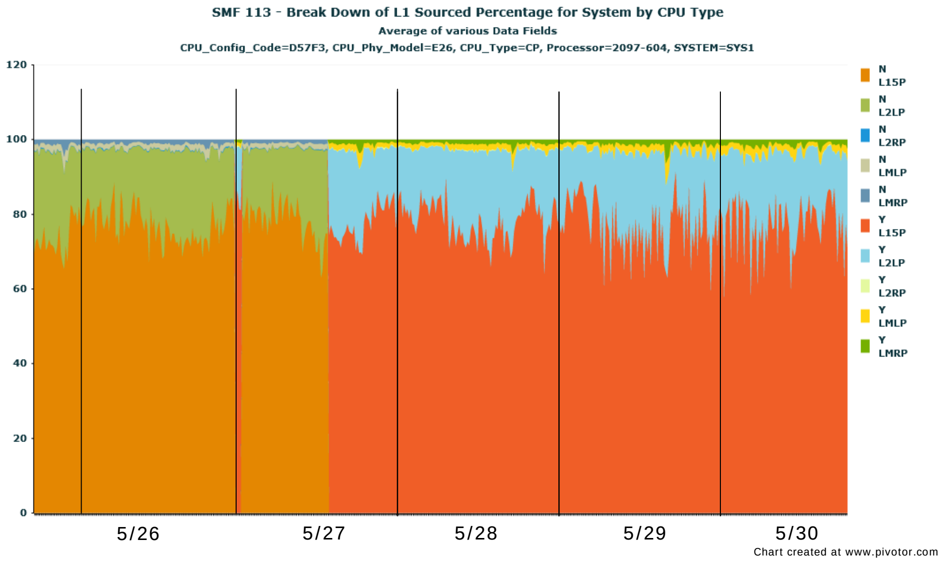
L1 Misses
L1 Hits
Instruction Cache | Data Cache
L15 | L2L | L2R | LM | RM
L15 | L2L | L2R | LM | RM

- □ Unfortunately Extended counts not granular to allow a better understand of L1 source affects to problem state or supervisor state
  - ■ So can only get based on Basic Counters (see next slides)

HD Before/After Example – Small MP



HD Before/After Example – Small MP



HD Before/After Example – Small MP

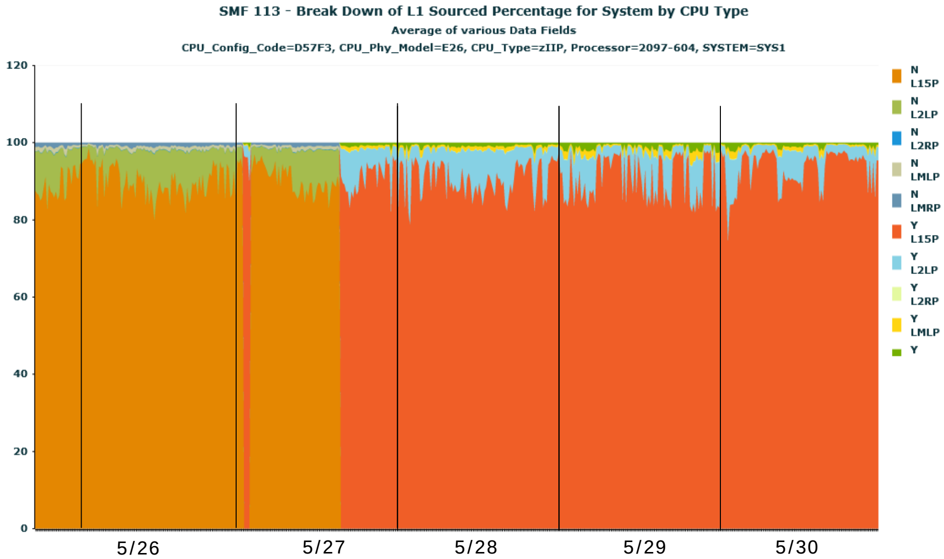


Chart created at www.pivotor.com

CPU 4 (Med Pool) L1 Sourced Breakdown

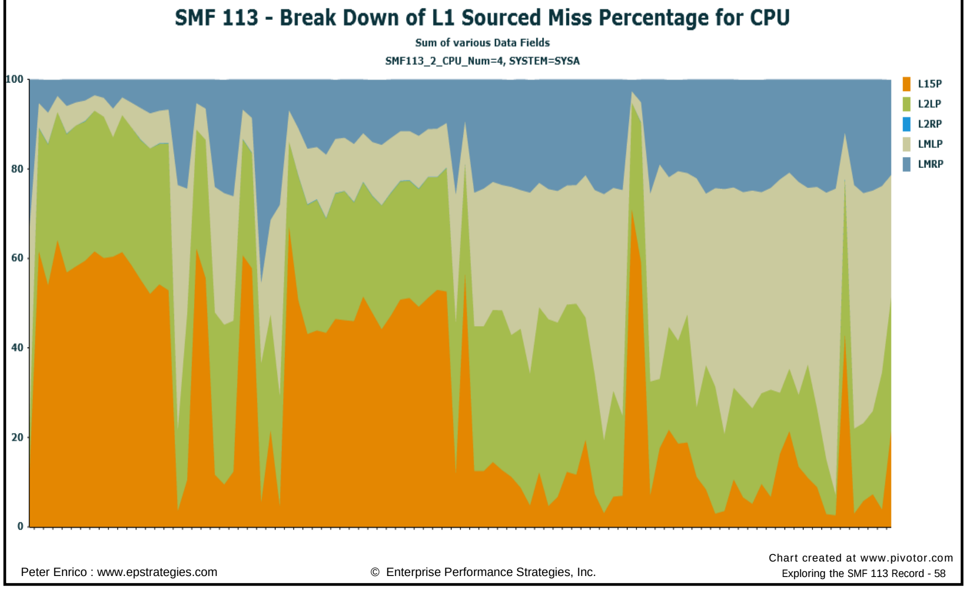
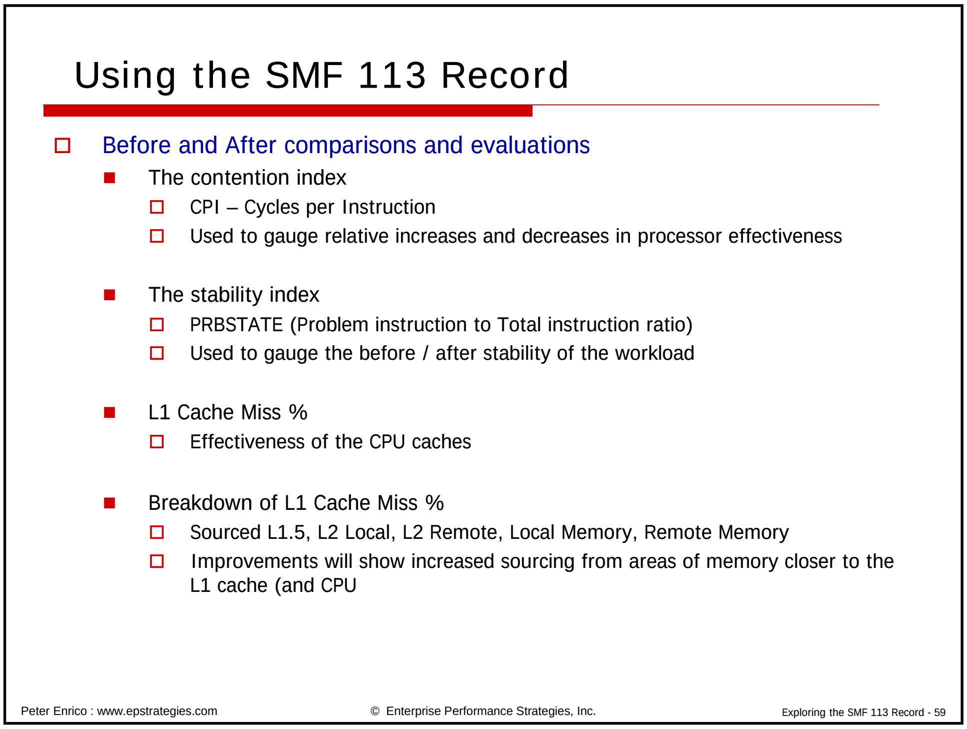
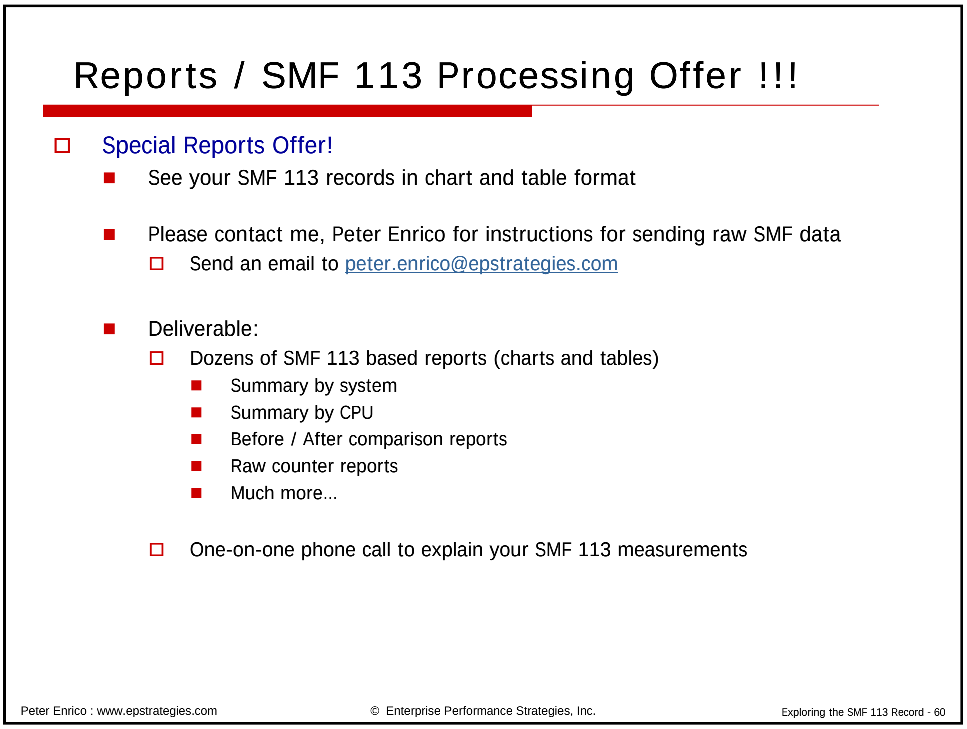


Chart created at www.pivotor.com



Using the SMF 113 Record

- Before and After comparisons and evaluations
  - The contention index
    - CPI – Cycles per Instruction
    - Used to gauge relative increases and decreases in processor effectiveness
  - The stability index
    - PRBSTATE (Problem instruction to Total instruction ratio)
    - Used to gauge the before / after stability of the workload
  - L1 Cache Miss %
    - Effectiveness of the CPU caches
  - Breakdown of L1 Cache Miss %
    - Sourced L1.5, L2 Local, L2 Remote, Local Memory, Remote Memory
    - Improvements will show increased sourcing from areas of memory closer to the L1 cache (and CPU



Reports / SMF 113 Processing Offer !!!

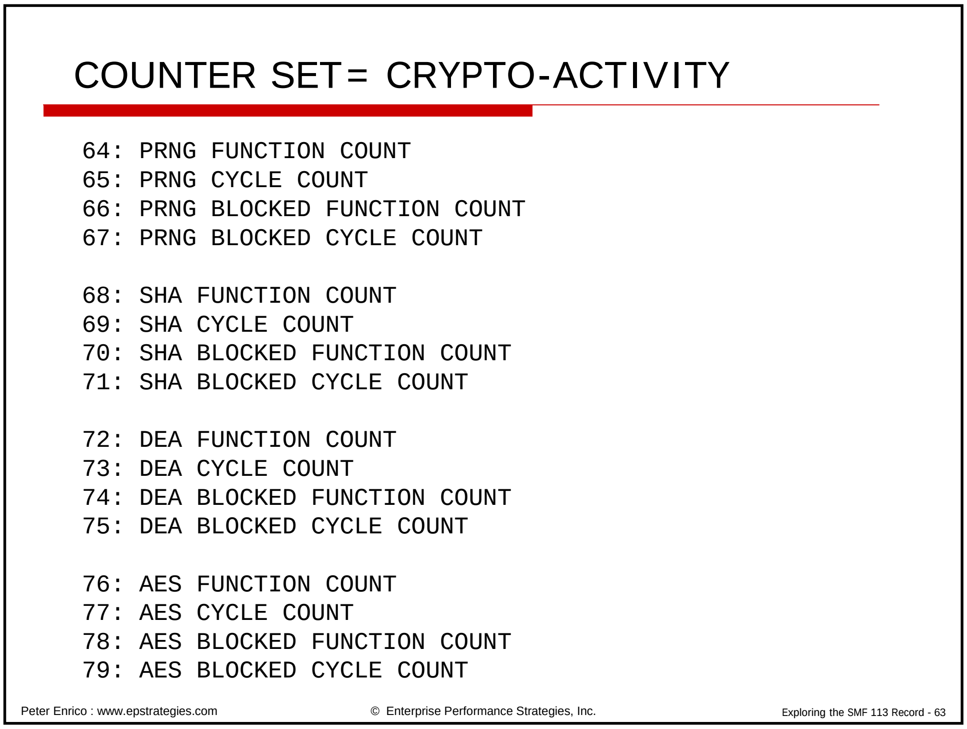
- Special Reports Offer!
  - See your SMF 113 records in chart and table format
  - Please contact me, Peter Enrico for instructions for sending raw SMF data
    - Send an email to peter.enrico@epstrategies.com
  - Deliverable:
    - Dozens of SMF 113 based reports (charts and tables)
      - Summary by system
      - Summary by CPU
      - Before / After comparison reports
      - Raw counter reports
      - Much more…
    - One-on-one phone call to explain your SMF 113 measurements

Crypto Counters

(If time permits)

COUNTER SET= CRYPTO-ACTIVITY

- zArchitecture include a something called message-security assist that supports cryptographic operations
- On z10 machines, 2 CPs can share a single crypto processor
  - So there is the concept of interference
  - When 2 sharing CPs attempt to use same coprocessor, one will be blocked until a predetermined time slice has passed
- Counter Set = Crypto-Activity
  - Contains counters of the crypto processor activity
  - Interference activity
- A cryptographic coprocessor group
  - Contains of a cipher coprocessor for DEA, AES
  - Contains a hash coprocessor for SHA
- Crypto processors with 2 CPs so blocking possible



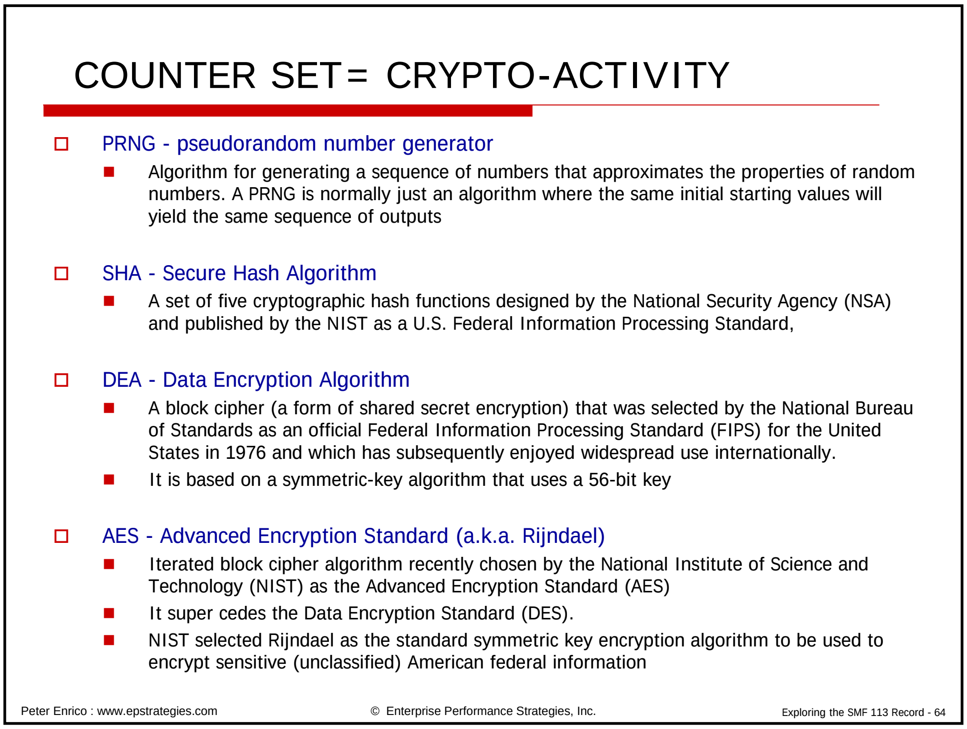
COUNTER SET= CRYPTO-ACTIVITY

64: PRNG FUNCTION COUNT
65: PRNG CYCLE COUNT
66: PRNG BLOCKED FUNCTION COUNT
67: PRNG BLOCKED CYCLE COUNT

68: SHA FUNCTION COUNT
69: SHA CYCLE COUNT
70: SHA BLOCKED FUNCTION COUNT
71: SHA BLOCKED CYCLE COUNT

72: DEA FUNCTION COUNT
73: DEA CYCLE COUNT
74: DEA BLOCKED FUNCTION COUNT
75: DEA BLOCKED CYCLE COUNT

76: AES FUNCTION COUNT
77: AES CYCLE COUNT
78: AES BLOCKED FUNCTION COUNT
79: AES BLOCKED CYCLE COUNT



COUNTER SET= CRYPTO-ACTIVITY

- □ PRNG - pseudorandom number generator
  - Algorithm for generating a sequence of numbers that approximates the properties of random numbers. A PRNG is normally just an algorithm where the same initial starting values will yield the same sequence of outputs
- □ SHA - Secure Hash Algorithm
  - A set of five cryptographic hash functions designed by the National Security Agency (NSA) and published by the NIST as a U.S. Federal Information Processing Standard,
- □ DEA - Data Encryption Algorithm
  - A block cipher (a form of shared secret encryption) that was selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the United States in 1976 and which has subsequently enjoyed widespread use internationally.
  - It is based on a symmetric-key algorithm that uses a 56-bit key
- □ AES - Advanced Encryption Standard (a.k.a. Rijndael)
  - Iterated block cipher algorithm recently chosen by the National Institute of Science and Technology (NIST) as the Advanced Encryption Standard (AES)
  - It super cedes the Data Encryption Standard (DES).
  - NIST selected Rijndael as the standard symmetric key encryption algorithm to be used to encrypt sensitive (unclassified) American federal information

COUNTER SET= CRYPTO *.CNT file excerpt example

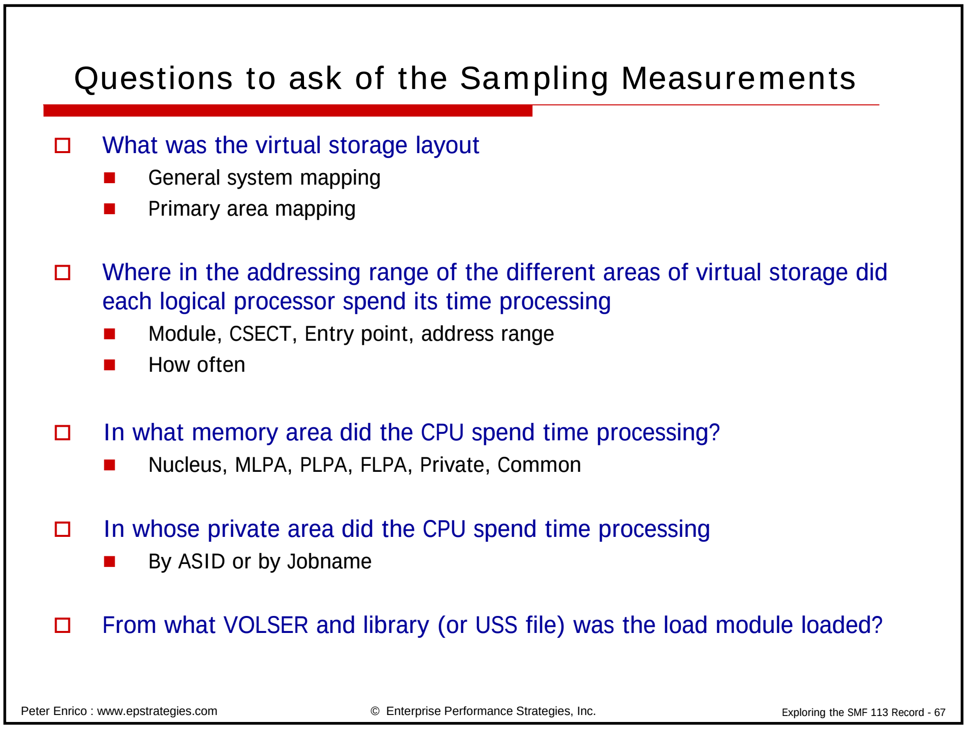
```
COUNTER SET= CRYPTO-ACTIVITY
COUNTER IDENTIFIERS:
  64: PRNG FUNCTION COUNT
  65: PRNG CYCLE COUNT
  66: PRNG BLOCKED FUNCTION COUNT
  67: PRNG BLOCKED CYCLE COUNT
  68: SHA FUNCTION COUNT
  69: SHA CYCLE COUNT
  70: SHA BLOCKED FUNCTION COUNT
  71: SHA BLOCKED CYCLE COUNT
  72: DEA FUNCTION COUNT
  73: DEA CYCLE COUNT
  74: DEA BLOCKED FUNCTION COUNT
  75: DEA BLOCKED CYCLE COUNT
  76: AES FUNCTION COUNT
  77: AES CYCLE COUNT
  78: AES BLOCKED FUNCTION COUNT
  79: AES BLOCKED CYCLE COUNT

START TIME: 2010/03/16 11:25:21  START TOD: C5AFCE3D7E54909C
END TIME:   2010/03/16 14:44:15  END TOD:   C5AFFAB2674B2F8C
COUNTER VALUES (HEXADECIMAL) FOR CPU 00 (CPU SPEED = 4404 CYCLES/MIC):
64- 67 0000000000000000 0000000000000000 0000000000000000 0000000000000000
68- 71 000000000000009D 0000000000043B46 0000000000000000 0000000000000000
72- 75 0000000000000000 0000000000000000 0000000000000000 0000000000000000
76- 79 0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

CPU Measurement Facility Sampling

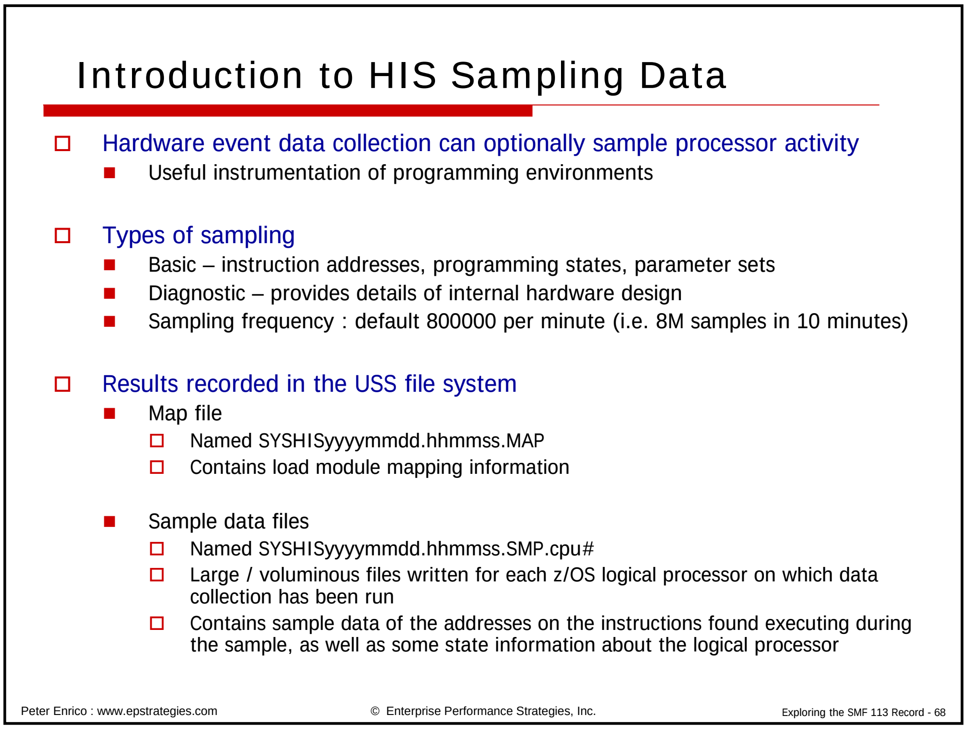
(If time permits)

(Note: Real quickly for completeness before discussing SMF 113)



Questions to ask of the Sampling Measurements

- □ What was the virtual storage layout
  - General system mapping
  - Primary area mapping
- □ Where in the addressing range of the different areas of virtual storage did each logical processor spend its time processing
  - Module, CSECT, Entry point, address range
  - How often
- □ In what memory area did the CPU spend time processing?
  - Nucleus, MLPA, PLPA, FLPA, Private, Common
- □ In whose private area did the CPU spend time processing
  - By ASID or by Jobname
- □ From what VOLSER and library (or USS file) was the load module loaded?



Introduction to HIS Sampling Data

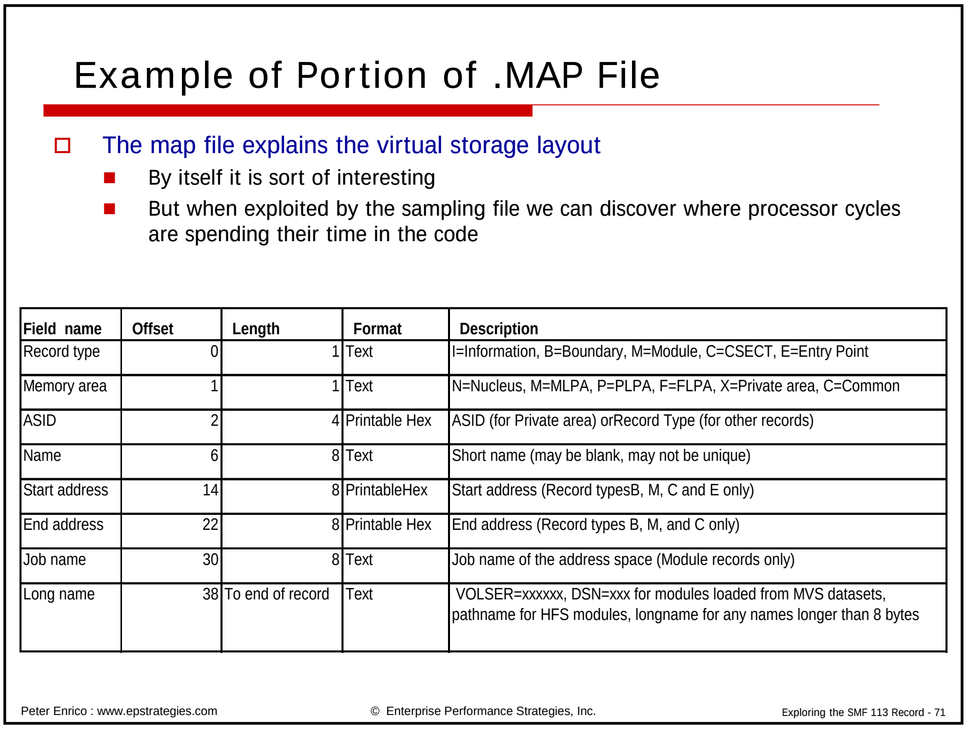
- □ Hardware event data collection can optionally sample processor activity
  - Useful instrumentation of programming environments
- □ Types of sampling
  - Basic – instruction addresses, programming states, parameter sets
  - Diagnostic – provides details of internal hardware design
  - Sampling frequency : default 800000 per minute (i.e. 8M samples in 10 minutes)
- □ Results recorded in the USS file system
  - Map file
    - □ Named SYSHISyyyymmdd.hhmmss.MAP
    - □ Contains load module mapping information
  - Sample data files
    - □ Named SYSHISyyyymmdd.hhmmss.SMP.cpu#
    - □ Large / voluminous files written for each z/OS logical processor on which data collection has been run
    - □ Contains sample data of the addresses on the instructions found executing during the sample, as well as some state information about the logical processor

Introduction to HIS Sampling Data

- Contents of load module mapping output file
  - MAPONLY option
    - Used to collect only load module mapping information
  - MAPASID
    - Used to map a specific set of ASIDs (address spaces) to produce private area load module map
      - Example MAPASID=ALL, or MAPASID=(A,1E,30)
  - MAPJOB
    - Used to map a specific set of job names to produce private area load module maps
      - Example: MAPJOB=(WLMSPAS1,MDD*,TSOA*)

Example of Portion of .MAP File

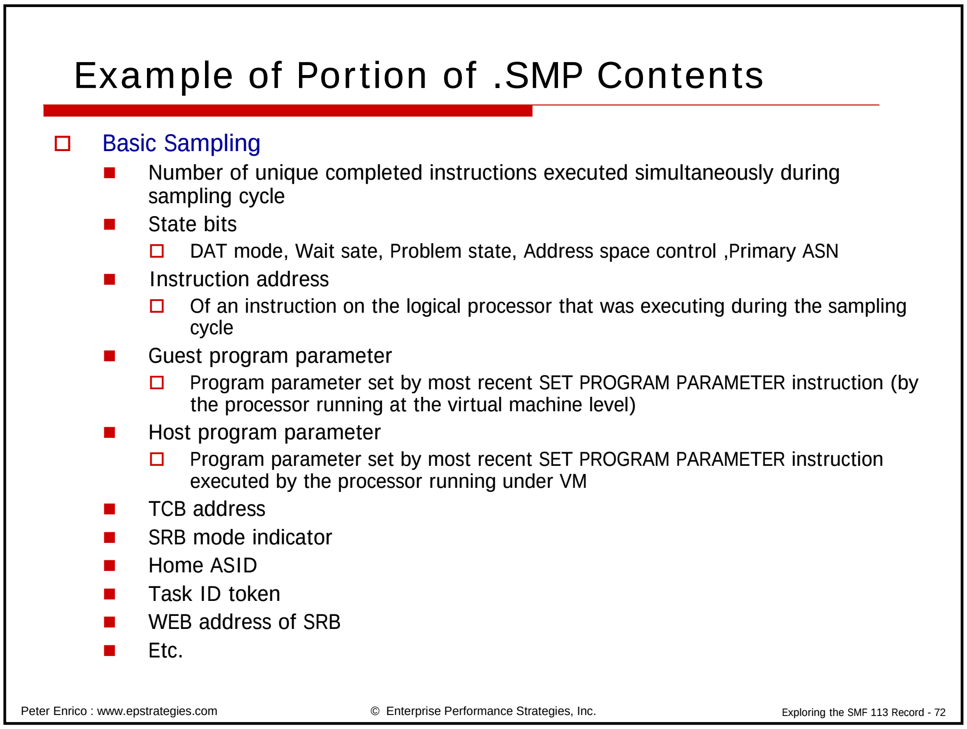
```
I SYS SY1
I SMFIIBM2
I OS  z/OS
I FMIDHBB7750
I DATE08091
I TIME16311902
I MAP V1R1
I LPID00000000
I MACH00002097
B BDY PRIVATE 00000000008FFFFF
B BDY CSA     0090000000BB2FFF
.  .  .
CNNUC IECVPRNT00FD700000FD74F7
ENNUC PRTDSE  00FD7006
ENNUC PRTSIO  00FD700C
.  .  .
MPPLPAIGG019T800BDE46000BDE53F
MPPLPAIGG019TX00BDE54000BDE5DF
.  .  .
MMMLPAIEFACTRT06663CD006663EBF        VOLSER=CTTPAKDSN=ARTMVS.EXITS.LOAD
CMMLPAIEFACTRT06663CD006663EBF
MMMLPAEZBREARR24CA000024D16FFF
.  .  .
MX0002IEAVXMAS25B0000025B01FC7PCAUTH  VOLSER=ZD110 DSN=SYS1.NUCLEUS
CX0002IEAVXMAS25B0000025B01167
.  .  .
MX0003IAXDINIT25B0000025B00B77RASP    VOLSER=ZD110 DSN=SYS1.NUCLEUS
CX0003IAXDI   25B0000025B00B77
```



Example of Portion of .MAP File

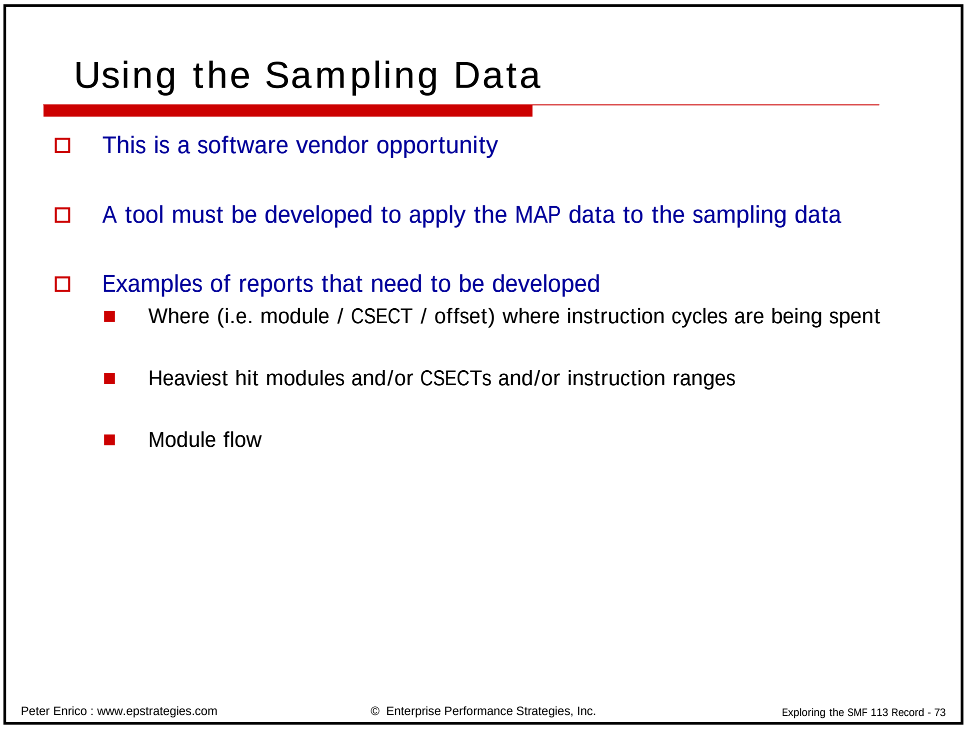
- The map file explains the virtual storage layout
  - By itself it is sort of interesting
  - But when exploited by the sampling file we can discover where processor cycles are spending their time in the code

Field name	Offset	Length	Format	Description
Record type	0	1	Text	I=Information, B=Boundary, M=Module, C=CSECT, E=Entry Point
Memory area	1	1	Text	N=Nucleus, M=MLPA, P=PLPA, F=FLPA, X=Private area, C=Common
ASID	2	4	Printable Hex	ASID (for Private area) orRecord Type (for other records)
Name	6	8	Text	Short name (may be blank, may not be unique)
Start address	14	8	PrintableHex	Start address (Record typesB, M, C and E only)
End address	22	8	Printable Hex	End address (Record types B, M, and C only)
Job name	30	8	Text	Job name of the address space (Module records only)
Long name	38	To end of record	Text	VOLSER=xxxxxx, DSN=xxx for modules loaded from MVS datasets, pathname for HFS modules, longname for any names longer than 8 bytes



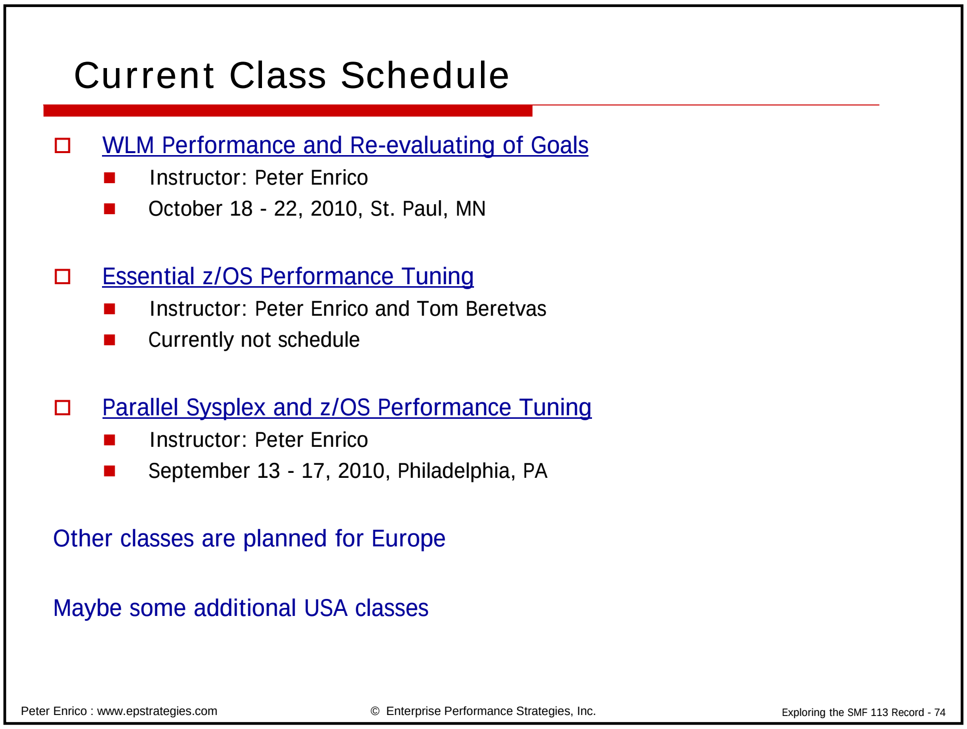
Example of Portion of .SMP Contents

- Basic Sampling
  - Number of unique completed instructions executed simultaneously during sampling cycle
  - State bits
    - DAT mode, Wait sate, Problem state, Address space control ,Primary ASN
  - Instruction address
    - Of an instruction on the logical processor that was executing during the sampling cycle
  - Guest program parameter
    - Program parameter set by most recent SET PROGRAM PARAMETER instruction (by the processor running at the virtual machine level)
  - Host program parameter
    - Program parameter set by most recent SET PROGRAM PARAMETER instruction executed by the processor running under VM
  - TCB address
  - SRB mode indicator
  - Home ASID
  - Task ID token
  - WEB address of SRB
  - Etc.



Using the Sampling Data

- □ This is a software vendor opportunity
- □ A tool must be developed to apply the MAP data to the sampling data
- □ Examples of reports that need to be developed
  - ■ Where (i.e. module / CSECT / offset) where instruction cycles are being spent
  - ■ Heaviest hit modules and/or CSECTs and/or instruction ranges
  - ■ Module flow



Current Class Schedule

- □ WLM Performance and Re-evaluating of Goals
  - ■ Instructor: Peter Enrico
  - ■ October 18 - 22, 2010, St. Paul, MN
- □ Essential z/OS Performance Tuning
  - ■ Instructor: Peter Enrico and Tom Beretvas
  - ■ Currently not schedule
- □ Parallel Sysplex and z/OS Performance Tuning
  - ■ Instructor: Peter Enrico
  - ■ September 13 - 17, 2010, Philadelphia, PA

Other classes are planned for Europe

Maybe some additional USA classes

Thank You!

- □ We thank you for attending this Webinar, for your business, and for your support

- □ z/OS Performance Education classes
  - We hope to see you in class sometime
  - Remember, it is a week of learning and doing an actual analysis!

- □ z/OS Performance Consulting services
  - We would like to help you with the performance measurement, analysis, and tuning of your z/OS environment

- □ Pivotor – Our data mining and data reporting software
  - Including z/OS Performance Data (as well as other platforms)
  - This is an extremely powerful data mining, data reporting, and data analysis tool
  - It is also a great front end to your SAS databases (such as MXG and MICS), your Tivoli Decision Support Databases, or to your raw SMF data
  - Great for UNIX / Linux / other data as well