Exploring the SMF 113 Processor Cache Counters

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

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

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. In addition, the SMF 113 measurements have become the basis for IBM's LSPRs for processor sizing.
  - 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 or http://www.pivotor.com.

Copyright Notice:

Trademarks:

Enterprise Performance Strategies, Inc. presentation materials contain trademarks and registered trademarks of several companies.

The following are trademarks of Enterprise Performance Strategies, Inc.: **Health Check®, Reductions®, Pivotor®**

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries: IBM®, z/OS®, zSeries® WebSphere®, CICS®, DB2®, S390®, WebSphere Application Server®, and many others.

Other trademarks and registered trademarks may exist in this presentation

Reports Offer

- **Report Offer**
  - To help you get the most out of this presentation, please contact Peter Enrico to take advantage of his report offer. If you send Peter your own SMF data, Peter will generate an extensive set of reports directly related to this presentation topic.
  - You can then use these reports to see how your data measures up to the topic of this presentation.

Peter Enrico Presentations - Vienna

- If you like this presentation, please note my other scheduled presentations here in Vienna
- Exploring the SMF 113 Processor Cache Counters and LSPRs
  - Wednesday at 14:30
  - Friday at 10:15 (repeat)
- DASD I/O Analysis from the Workload Point-of-View
  - Thursday at 14:30
  - No repeat

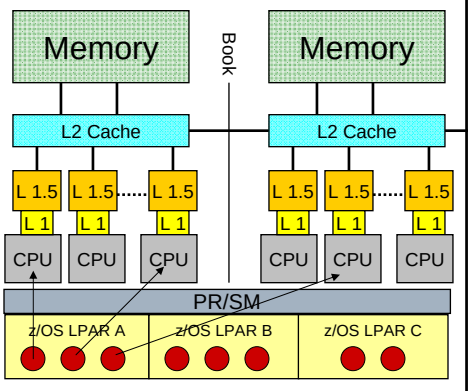
Presentation Overview

- Why the SMF 113
- Overview of contents of SMF 113
- Primary basic formulas for SMF 113 usage
- New 'Nest' related formulas

Introducing the SMF 113

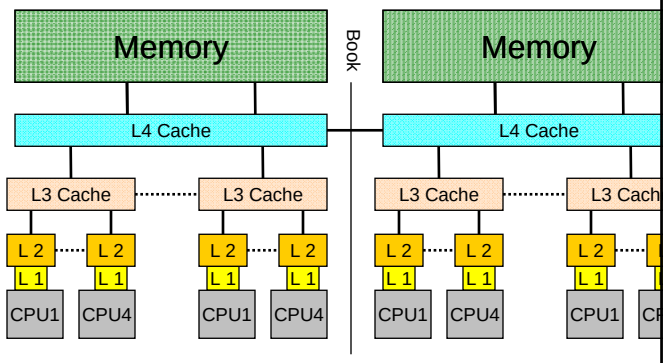
Performance Analyst View of z10 Processor

- 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/book
  - Memory
    - Up to 384GB per book
    - (up to 1.5TB per machine)
    - Option to spread memory into multiple books



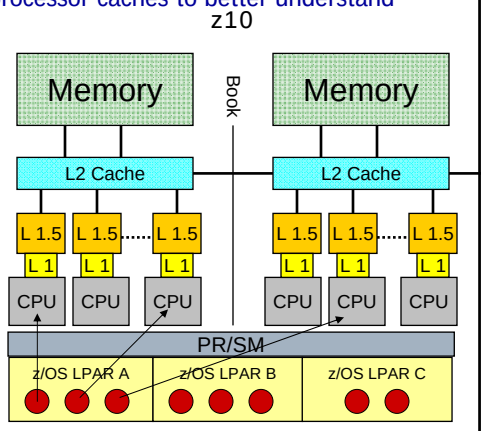
Performance Analyst View of z196 Processor

- It take more cycles to fetch information from further up in cache hierarchy
  - L1 (Private level)
    - Data: 128KB
    - Instruction: 64KB
  - L2 (Private level)
    - 1.5MB
  - L3
    - 24MB / chip
  - L4
    - 192MB / book
  - Memory



Greatest Usage of SMF 113

- IBM's LSPR Workloads
  - See next slide (but fully explained later)
- Used to illustrate the usage of the processor caches to better understand before and after changes
  - Not good for benchmarking
  - But good to assuage concerns or gain insights
- Usage of standard SMF records still required for full processor evaluations
  - SMF 30
  - SMF 70
  - SMF 72.3
  - Etc..



LSPR Table Example – Pre SMF 113

IBM System z10 EC (System z9 2094-701 = 1.00)

Processor	#CP	PCI	MSU	Mixed	LoIO-Mix	TI-Mix	CB-L	ODE-B	WASDB	OLTP-W	OLTP-T
2097-401	1	219	27	0.38	0.38	0.38	0.39	0.4	0.38	0.38	0.39
2097-402	2	414	51	0.73	0.73	0.72	0.76	0.77	0.73	0.69	0.73
2097-403	3	602	75	1.06	1.06	1.04	1.12	1.13	1.08	0.98	1.06
2097-404	4	782	97	1.37	1.39	1.34	1.47	1.48	1.41	1.25	1.37
2097-405	5	957	118	1.68	1.7	1.63	1.82	1.82	1.74	1.52	1.67
2097-406	6	1129	139	1.98	2.01	1.92	2.16	2.16	2.06	1.77	1.97
2097-407	7	1295	160	2.27	2.31	2.2	2.49	2.49	2.38	2.02	2.26
2097-408	8	1458	180	2.56	2.6	2.46	2.82	2.82	2.69	2.26	2.54
2097-409	9	1617	199	2.84	2.89	2.73	3.14	3.14	2.99	2.49	2.81
2097-410	10	1772	218	3.11	3.17	2.98	3.45	3.46	3.29	2.71	3.08
2097-411	11	1923	237	3.37	3.44	3.23	3.76	3.76	3.59	2.93	3.33
2097-412	12	2070	255	3.63	3.71	3.47	4.06	4.07	3.88	3.14	3.58
2097-501	1	473	58	0.83	0.83	0.83	0.85	0.86	0.82	0.81	0.83
2097-502	2	894	110	1.57	1.58	1.55	1.64	1.65	1.58	1.48	1.58
2097-503	3	1296	160	2.27	2.29	2.23	2.42	2.43	2.32	2.1	2.28
2097-504	4	1681	207	2.95	2.98	2.88	3.17	3.19	3.04	2.68	2.95
2097-505	5	2055	252	3.6	3.65	3.5	3.91	3.94	3.74	3.24	3.6
2097-506	6	2418	296	4.24	4.3	4.1	4.63	4.67	4.42	3.78	4.23

LSPR Table Example – Post SMF 113

IBM System z10 EC (System z9 2094-701 = 1.00)

Processor	#CP	PCI**	MSU***	Low*	Average*	High*
2097-401	1	214	27	0.39	0.38	0.35
2097-402	2	404	51	0.76	0.72	0.66
2097-403	3	589	75	1.13	1.05	0.95
2097-404	4	768	97	1.48	1.37	1.23
2097-405	5	943	118	1.84	1.68	1.5
2097-406	6	1115	139	2.18	1.99	1.77
2097-407	7	1284	160	2.52	2.29	2.03
2097-408	8	1449	180	2.85	2.59	2.29
2097-409	9	1610	199	3.18	2.88	2.55
2097-410	10	1769	218	3.5	3.16	2.8
2097-411	11	1924	237	3.81	3.44	3.05
2097-412	12	2077	255	4.12	3.71	3.29
2097-501	1	462	58	0.85	0.83	0.76
2097-502	2	882	110	1.65	1.58	1.42
2097-503	3	1281	160	2.43	2.29	2.05
2097-504	4	1661	207	3.2	2.97	2.64
2097-505	5	2031	252	3.94	3.63	3.22
2097-506	6	2391	296	4.68	4.27	3.78

LSPRs and SMF 113s and RNI Hint

- SMF 113 measurements are now used to provide guidelines / hints for LSPR and zPCR processor sizing
- This RNI Hint table was documented in the Large System Performance Reference (LSPR)
  - Document Number SC28-1187-14
- The next slide shows an example of an LSPR chart used for processor sizing
- Using the SMF 113 records you now need to calculate
  - L1MP - L1 Miss Per 100 Instructions
  - RNI – Relative Nest Intensity
- Note: This table and these guidelines are expected to change as more is learned from the SMF 113 records

L1MP	RNI	Workload Hint
<3%	>= 0.75	AVERAGE
	< 0.75	LOW
3% to 6%	>1.0	HIGH
	0.6 to 1.0	AVERAGE
	< 0.6	LOW
>6%	>=0.75	HIGH
	< 0.75	AVERAGE

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 SMF interval, and can be sync-ed to the SMF interval
      - Function available via APARs
- If Sampling enabled (Topic not presented in 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)

Highlights of SMF 113 Record

Processor Speed Information

- SMF113_2_CPSP
  - CPU speed in cycles per microsecond
  - Recorded for each logical CPU (but is really the physical CPU speed)
  - Example:
    - z10 : 4404 Cycles/Mic (i.e. 4.4 GHz)
    - z196: 5208 Cycles/Mic (i.e. 5.2 GHz)
- For knee capped processors
  - Will reflect the reduced speed
  - But zIIPs and zAAP on the machine will show full speed numbers

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

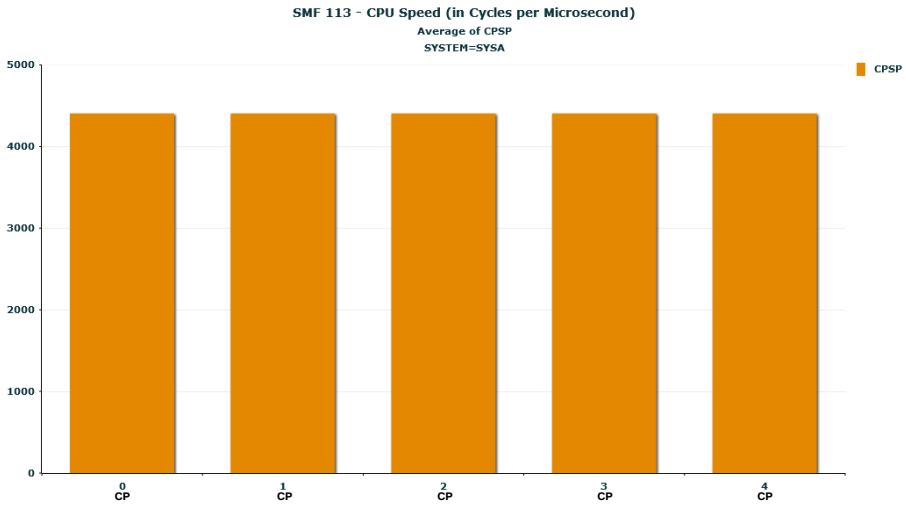
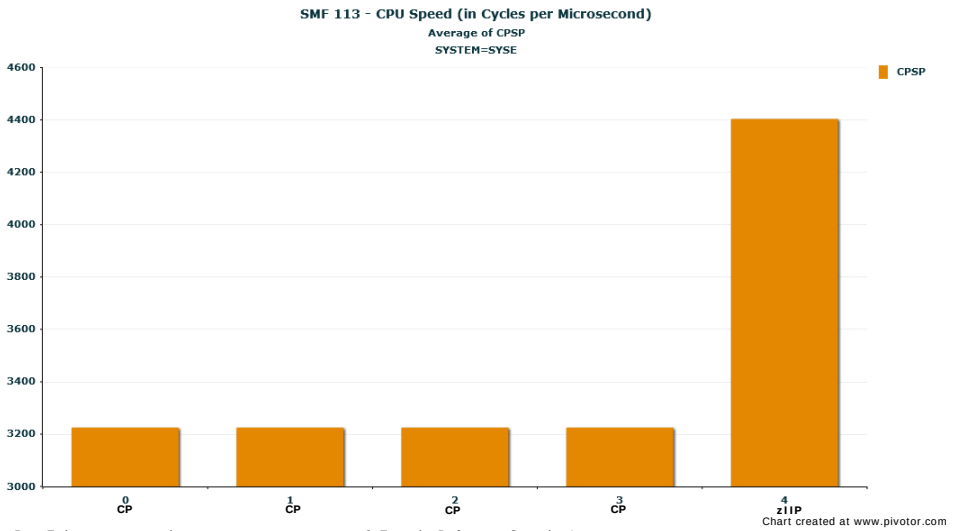


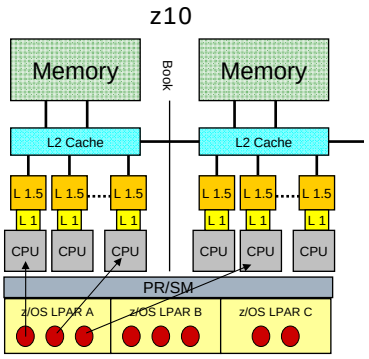
Chart created at www.pivotor.com

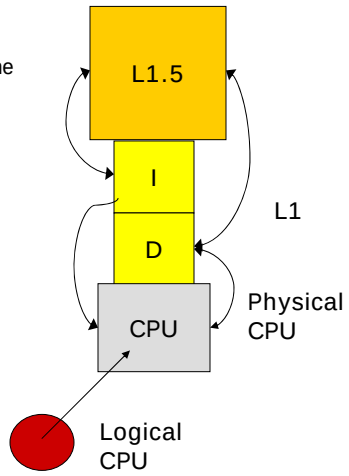
CPU Speed
- 4-CP way LPAR with 1 zIIP



COUNTER SET= BASIC / PROBLEM-STATE

- BASIC and Problem counters contain
  - L1 cache sourcing activity for both Data (D-cache) and Instruction (I-cache)
  - Contain instruction and cycle counters
  - Note 'Penalty' = 'Sourcing' = data/instruction gotten from somewhere and placed into L1 cache





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

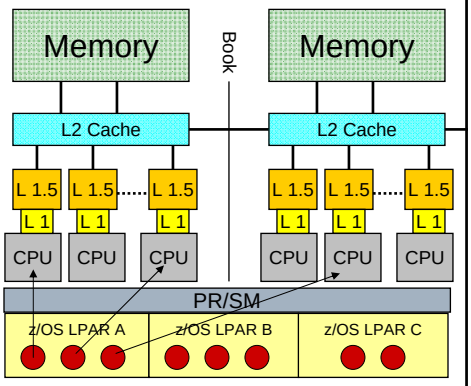
```
COUNTER SET= BASIC
COUNTER IDENTIFIERS:
   0: CYCLE COUNT
   1: INSTRUCTION COUNT
   2: L1 I-CACHE DIRECTORY-WRITE COUNT
   3: L1 I-CACHE PENALTY CYCLE COUNT
   4: L1 D-CACHE DIRECTORY-WRITE COUNT
   5: L1 D-CACHE PENALTY CYCLE COUNT

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 ---------------- ----------------
```

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)

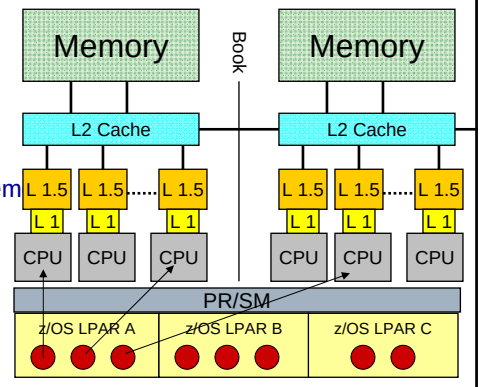
z10 COUNTER SET= EXTENDED

- Source 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)



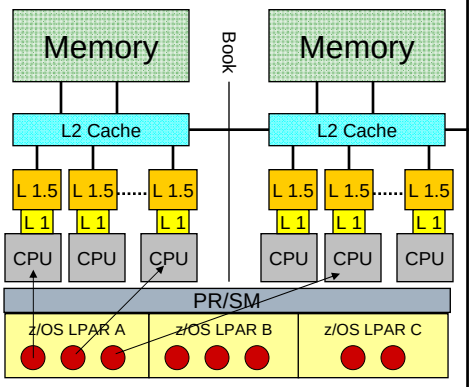
z10 COUNTER SET= EXTENDED

- Source 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)
- Source L1 from L2 Remote cache movem
  - 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)



z10 COUNTER SET= EXTENDED

- Source 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)
- Source L1 from Remote memory cache movement
  - Count does not exist, but see next slide for calculation



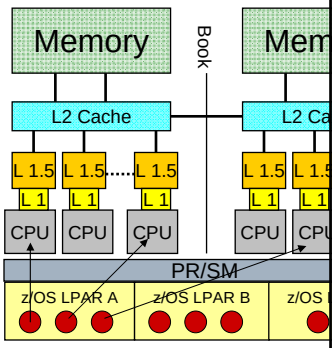
*Footnote: Notice, for some reason, reversal of 134 (D) and 135 (I) whereas all other counters are in (I) then (D) order.

z10 COUNTER SET= EXTENDED

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

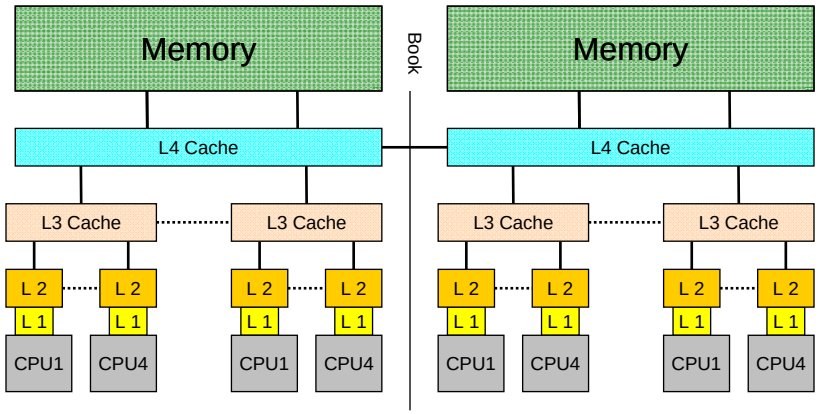
- L1 I-cache source from Remote Memory =
  - B2 : L1 I-Cache Dir-Write
    - (EC128 : write L1 I-cache from L1.5
      +EC130 : write L1 I-cache from L2 local
      +EC132 : write L1 I-cache from L2 remote
      +EC135 : write L1 I-cache from Local Memory
)

- L1 D-cache sourced from Remote Memory =
  - B4 : 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
)



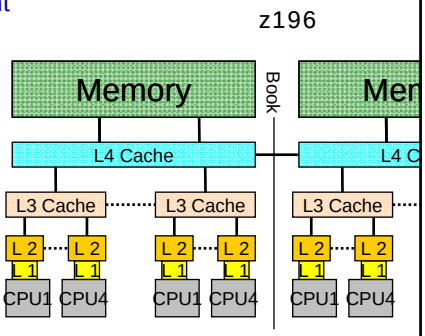
z196 Extended Cache Counters

z196



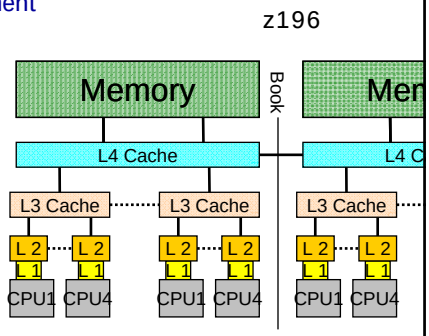
z196 Extended Cache Counters

- Source L1 from L2 cache movement
  - 128: Dir write to L1 D-cache dir from L2 cache (Data)
  - 129: Dir write to L1 I-cache dir from L2 cache (Instruction)
- Source L1 from L3 Off Chip On Book cache movement
  - 152: Dir write to L1 D-cache dir from L3 off chip on book cache (Data)
  - 153: Dir write to L1 I-cache dir from L3 off chip on book cache (Instruction)
- Source L1 from L3 Off Book cache movement
  - 134: Dir write to L1 D-cache dir from L3 off book cache (Data)
  - 143: Dir write to L1 I-cache dir from L3 off book cache (Instruction)



z196 Extended Cache Counters

- Source L1 from L4 Local cache movement
  - 135: Dir write to L1 D-cache dir from L4 local cache (Data)
  - 136: Dir write to L1 I-cache dir from L4 local cache (Instruction)
- Source L1 from L4 Remote cache movement
  - 138: Dir write to L1 D-cache dir from L4 remote cache (Data)
  - 139: Dir write to L1 I-cache dir from L4 remote cache (Instruction)
- Source L1 from Local Memory cache movement
  - 141: Dir write to L1 D-cache dir from local memory (Data)
  - 142: Dir write to L1 I-cache dir from local memory (Instruction)
- Source L1 from Remote Memory cache movement
  - Data movement - derived
  - Instruction movement - derived



Using Key SMF 113 Metrics

Using Summarized Data in Formulas

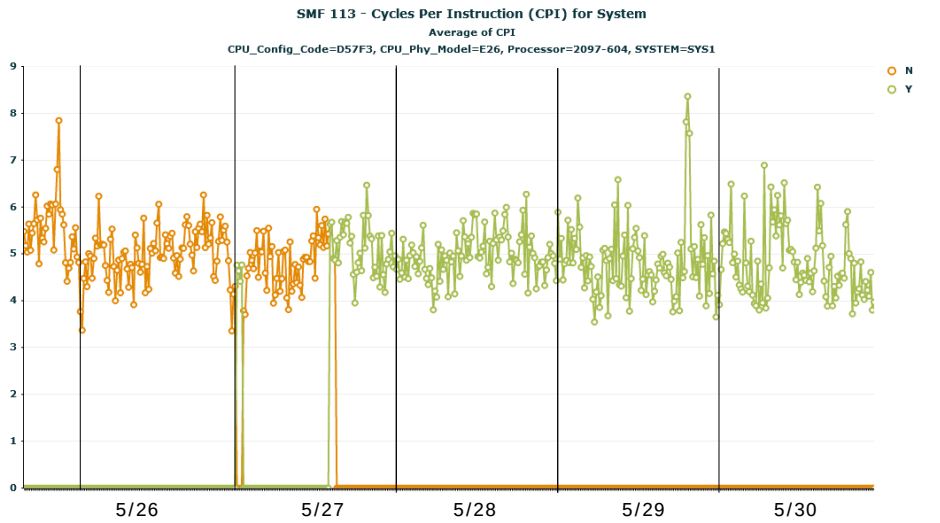
- When using the SMF 113 records, insights could be gained by summarizing the counters and formulas based on the following:
  - By system
  - By system, by CPU type (i.e. for all CPUs of a given type combined)
    - Example: For SYSA, for all CPs combined, all zIIPs, all zAAPs
  - By system, by CPU type, by CPU
    - Example: For SYSA, for CP or zIIP or zAAPs, by CPU number
  - By machine (but remember that counters only collected for z/OS images
    - Example: For CEC1 for all Systems

CPI – Cycles Per Instruction

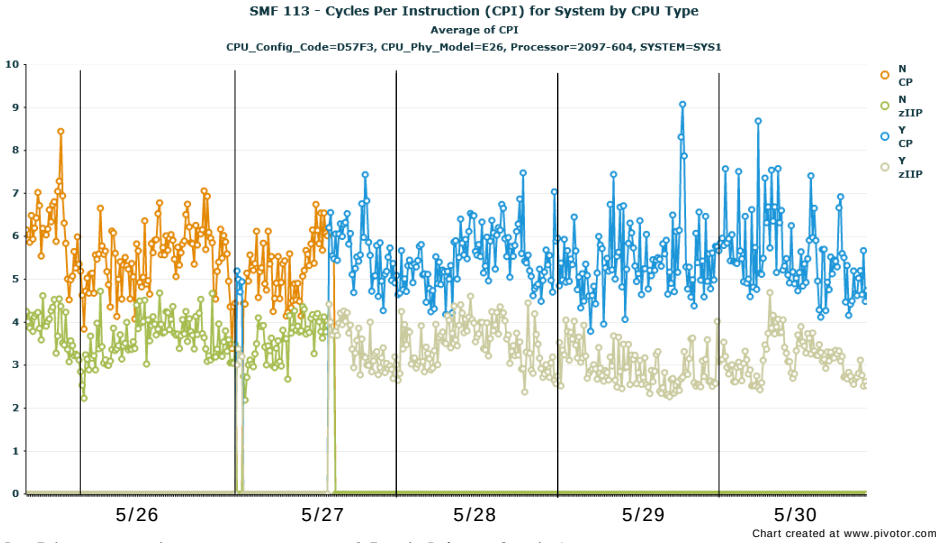
- Key metric to gauge processor contention
  - Useful when doing a before / after comparison
  - Over time, useful to understand instruction mixture consistency
- When CPI increases
  - It is taking more cycles to execute the instruction mix
  - Shows an increase in contention
- When CPI decreases
  - It is taking less cycles to execute the instruction mix
  - Shows a decrease in contention
- Cycles / Instruction
  - Counters needed
    - B0: Cycle Count
    - B1: Instruction Count

CPI = (Total Cycles / Total Instructions)
= (B0/B1)

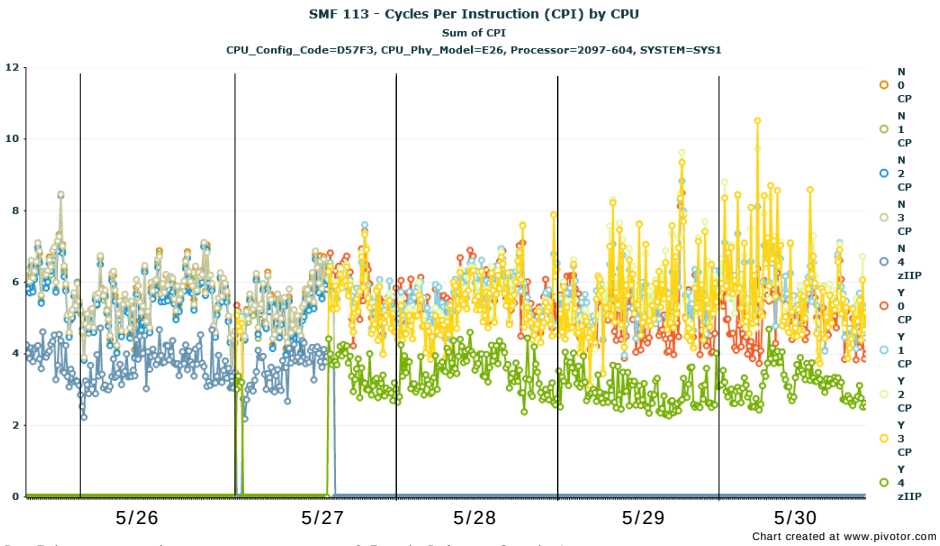
HD Before/After Example – Small MP



HD Before/After Example – Small MP



HD Before/After Example – Small MP



HD Before/After Example – Small MP

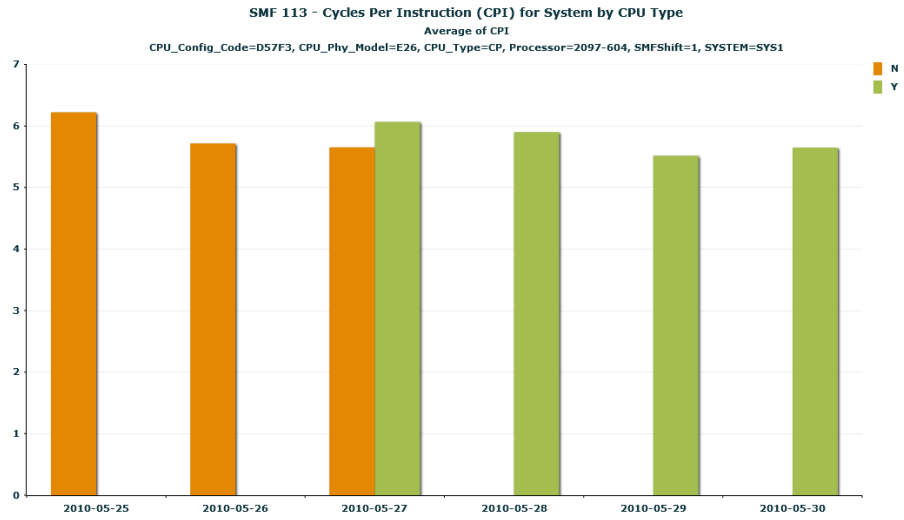


Chart created at www.pivotor.com

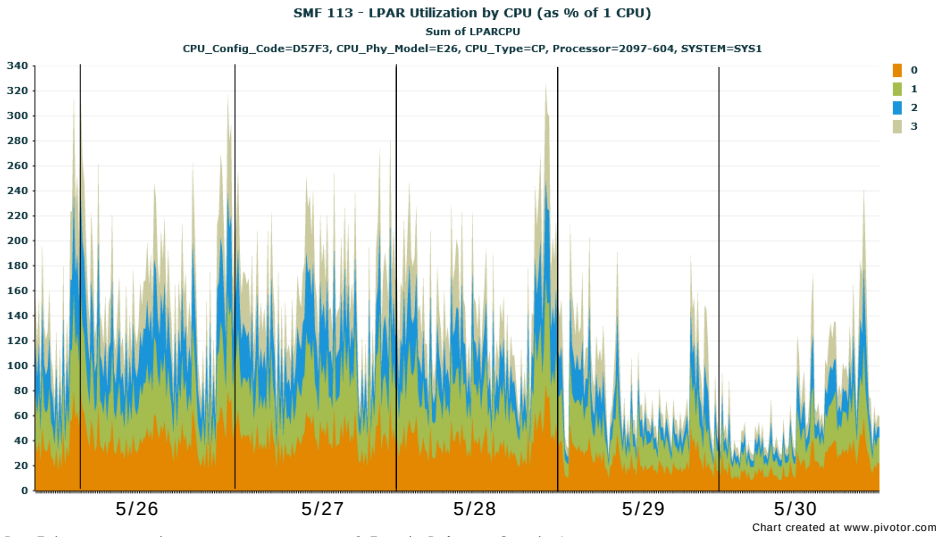
LPARCPU – LPAR Physical Busy %

- LPAR CPU
  - LPARCPU (Cycle CPU %) (based on Cycle CPU seconds captured and un-captured)
  - Counters needed
    - Processor Speed (cycles per microsecond) = SMF113_2_CPSP
    - B0: 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



PRBSTATE (Problem Instruction to Total Instruction)

- Problem to Total Instruction Ratio
  - Ratio of Problem State instructions to Total instructions
  - Counters needed
    - B1 = (Supervisor State Instructions + Problem State Instructions)
    - P33 = (Problem State Instructions)

  PRBSTATE = P33/ B1

- This is our stability factor
  - Useful to help indicate if the before / after workload is consistent

HD Before/After Example – Small MP

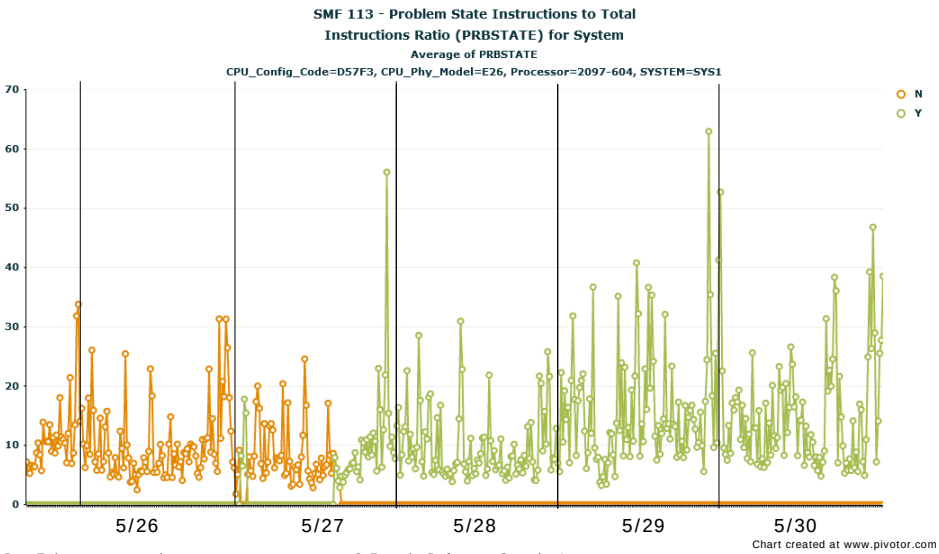


Chart created at www.pivotor.com

HD Before/After Example – Small MP

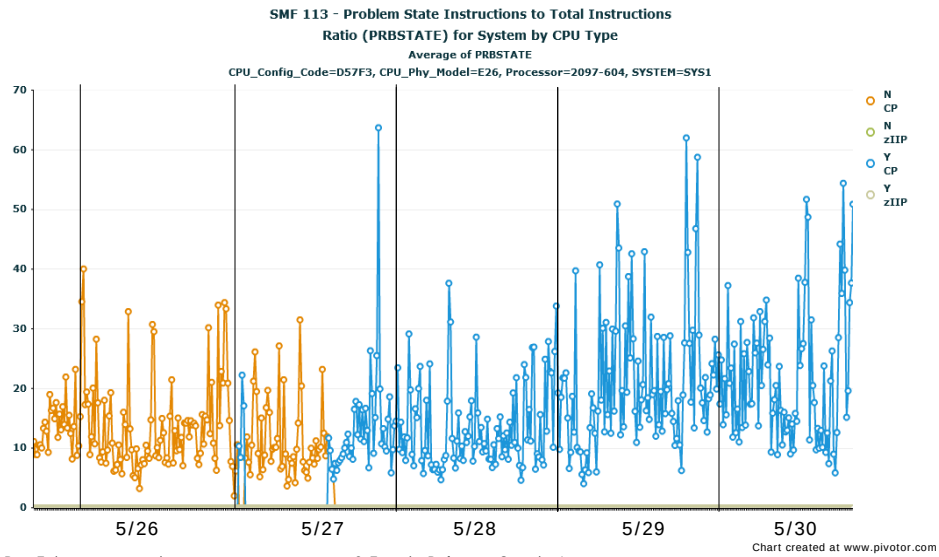


Chart created at www.pivotor.com

HD Before/After Example – Small MP

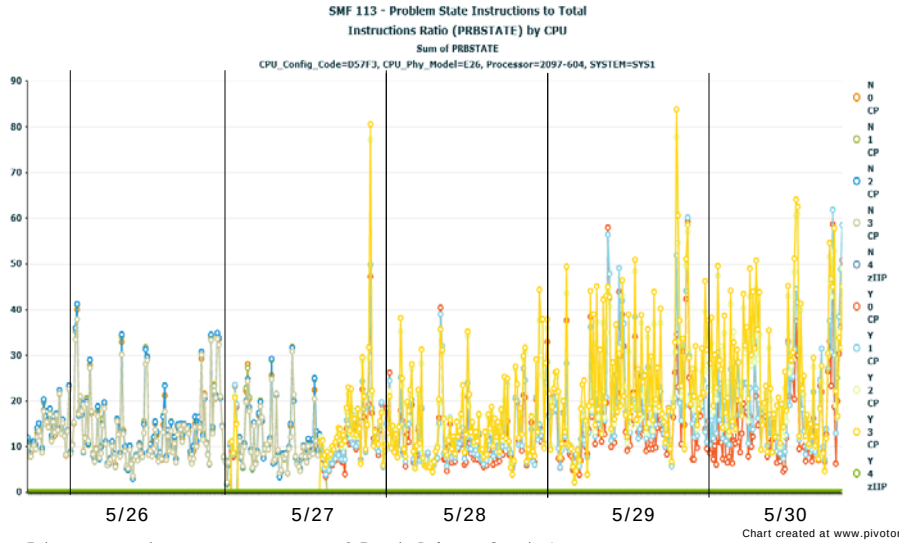


Chart created at www.pivotor.com

HD Before/After Example – Small MP

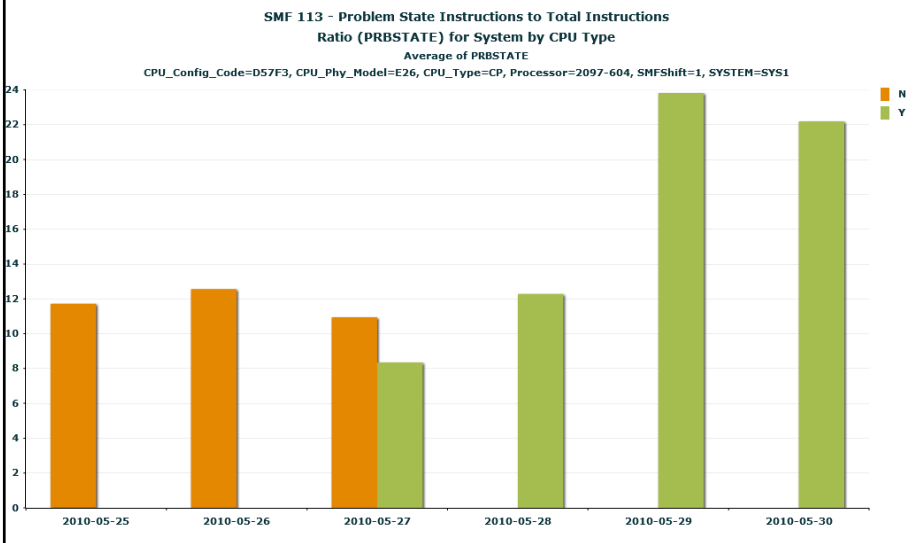


Chart created at www.pivotor.com

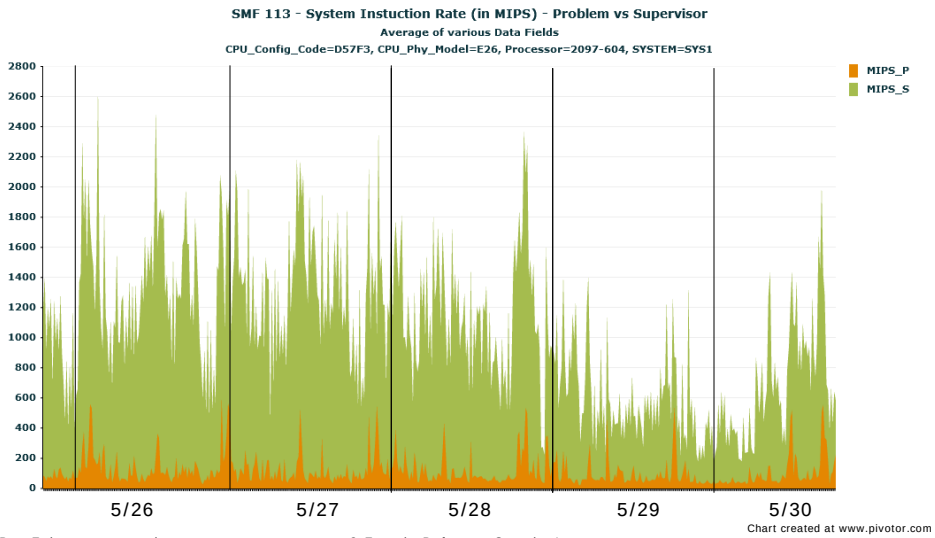
Useful Formulas

- 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
    - B1: Instruction Count
    - Measurement length in seconds

MIPS = (B1 / 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



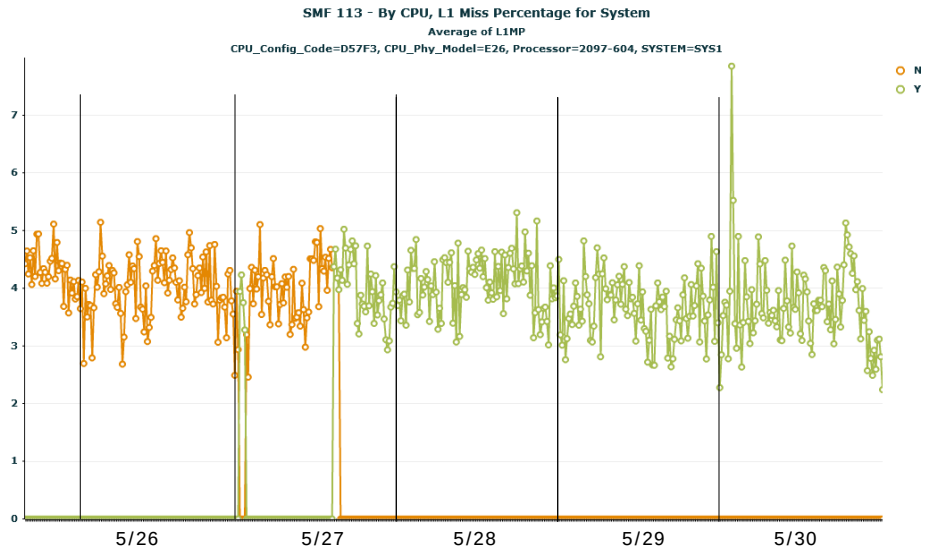
Level 1 misses per 100 instructions
(Renamed from L1 Cache Miss %)

- Level 1 misses per 100 instructions
  - Renamed from L1 Cache Miss %
    - To account for overlap
  - Measure of counters when L1 I-cache or L1 D-cache got a cache miss
  - (sort of like) Opposite of the hit percentage
    - Calculate miss rather than hit since the source % numbers (presented in subsequent slides) will be a breakdown of this cache miss value
  - Based on counters
    - B2: L1 I-Cache Dir-Write Count
    - B4: L1 D-Cache Dir-Write Count
    - B1: INSTRUCTION COUNT

*L1MP= ((B2 + B4) / B1) * 100*

  - Why is it not a miss percentage?
    - Instructions like LR (Load Register) don't need to access cache at all
    - Instructions like MVCL may access cache multiple times
    - So while same formula as 'Miss Percentage' this name is more accurate

HD Before/After Example – Small MP



HD Before/After Example – Small MP

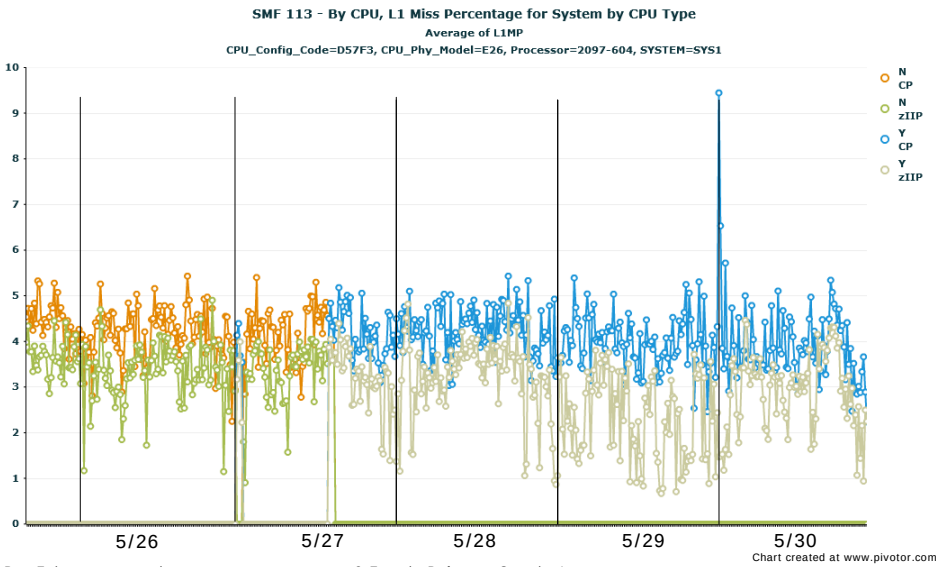


Chart created at www.pivotor.com

HD Before/After Example – Small MP

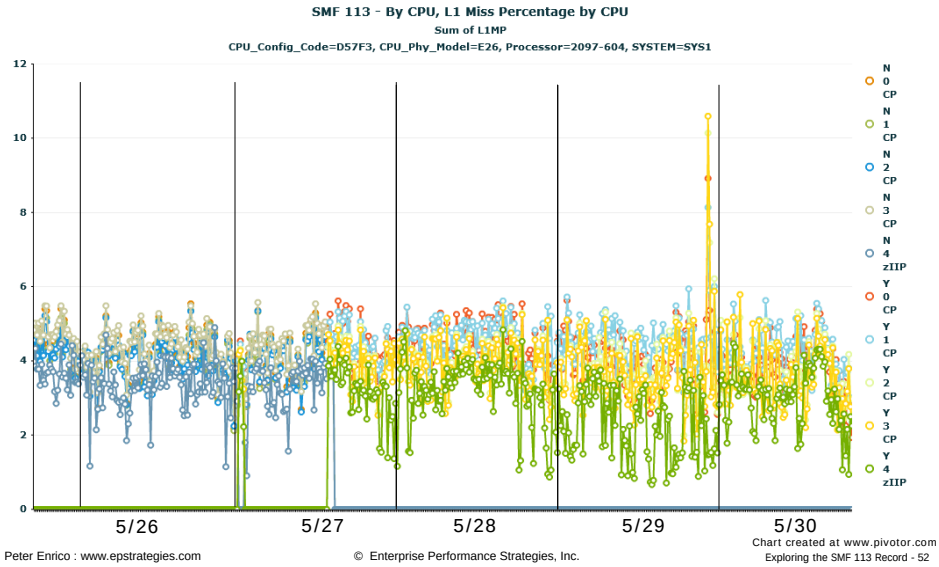
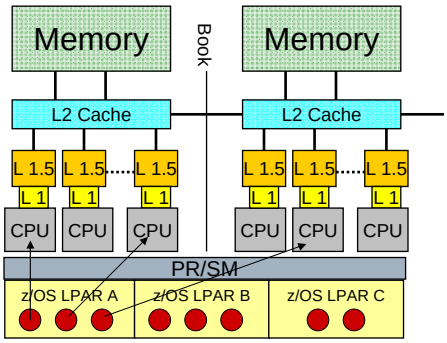


Chart created at www.pivotor.com

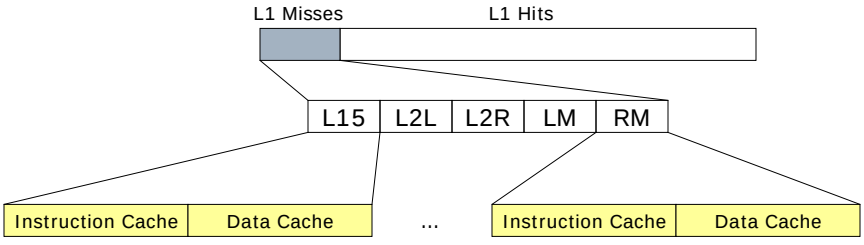
z10 Components of L1 Sourced

- If an L1 Miss Per 100 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



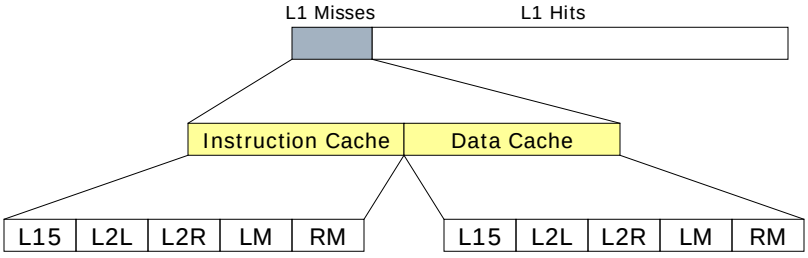
z10 From Where is L1 Sourced?

- 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



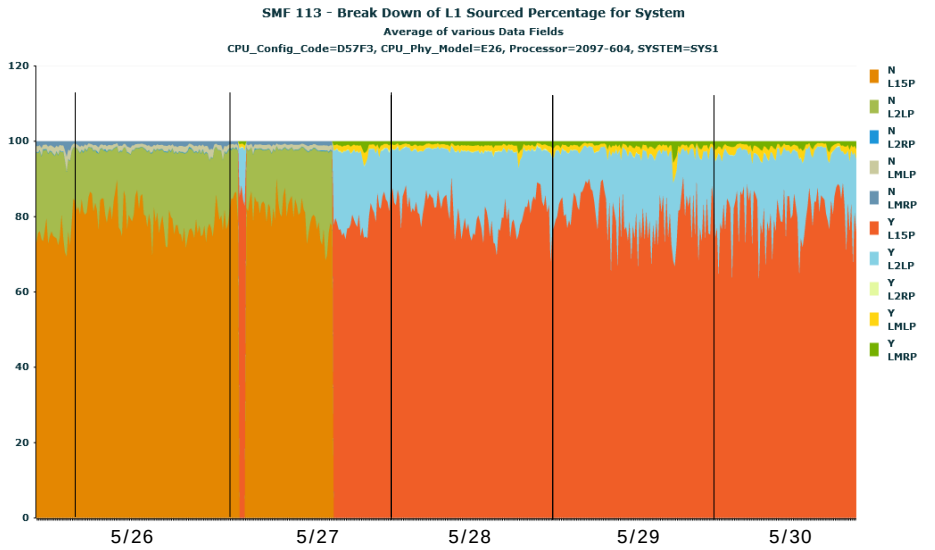
z10 From Where is L1 Sourced?

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



- 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)

z10 HD Before/After Example – Small MP



z10 HD Before/After Example – Small MP

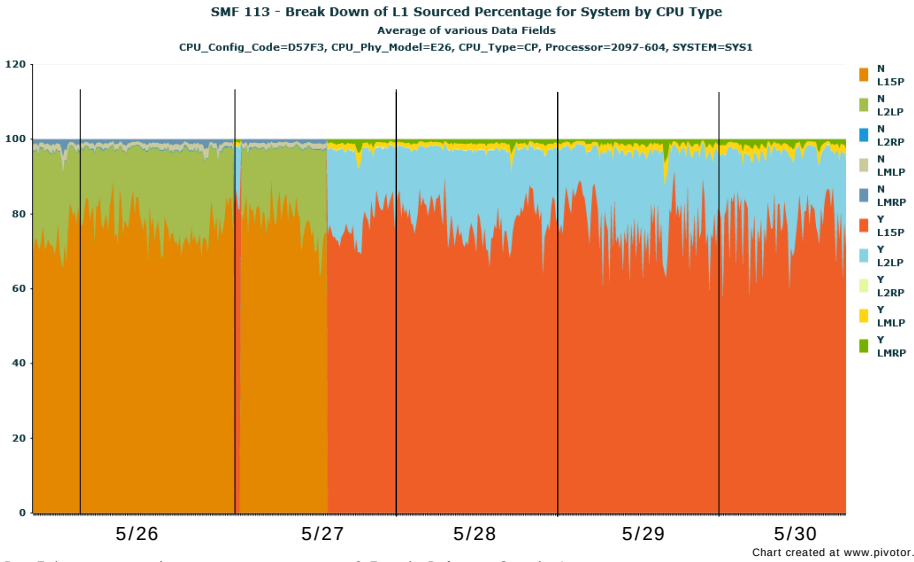


Chart created at www.pivotor.com

z10 HD Before/After Example – Small MP

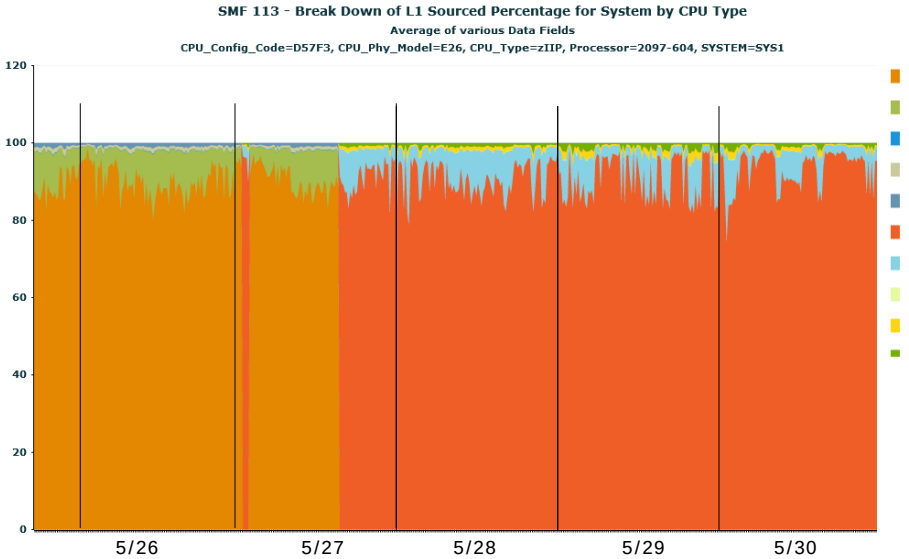


Chart created at www.pivotor.com

z10 CPU 4 (Med Pool) L1 Sourced Breakdown

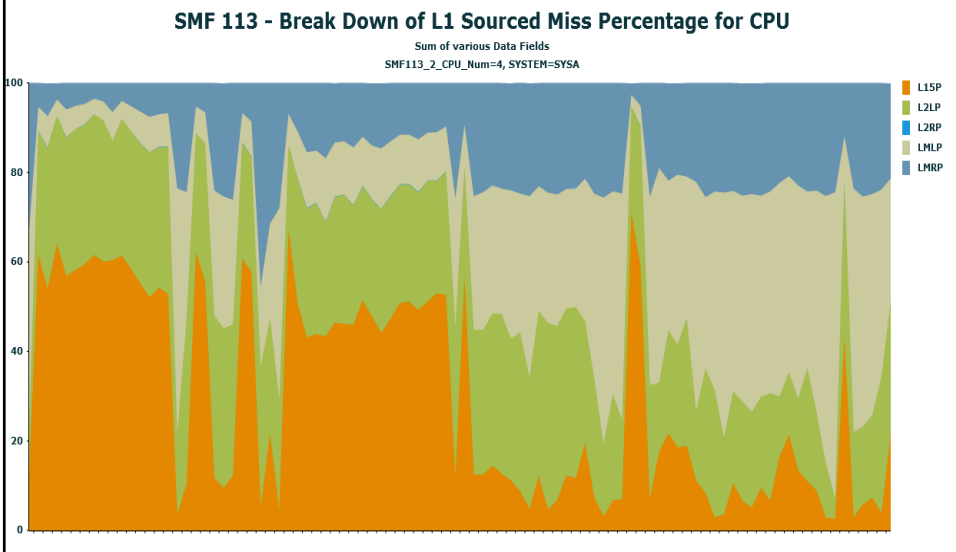


Chart created at www.pivotor.com

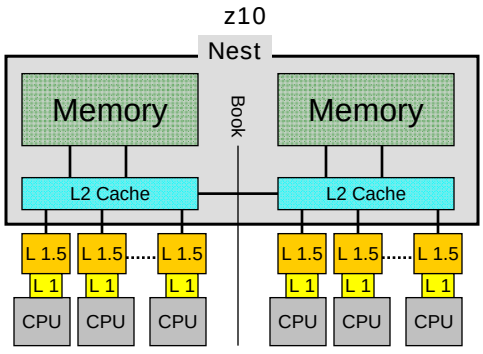
Using the SMF 113 Record

- 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 per 100 Instructions
    - Effectiveness of the CPU caches
  - Breakdown of L1 Cache Miss per 100 Instructions
    - 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

New 'Nest' Related Formulas

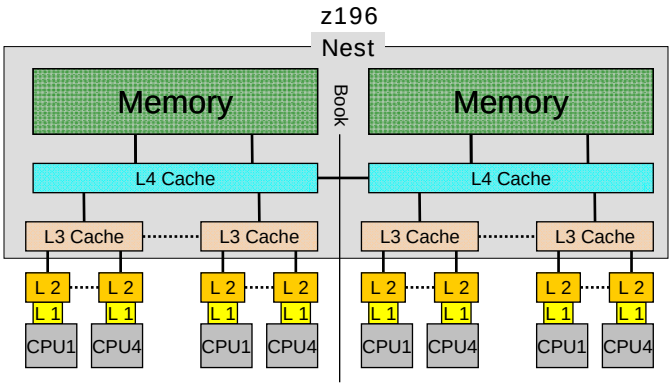
Nest View of z10 Processor

- Caches can be thought as divided into
  - Private Area Caches which are part of the processor design
    - Heavily influenced by instruction complexity and processor design
  - Shared Area Caches which are part of the memory hierarchy
    - This is called 'The Nest'
    - Heavily influence by workload mixtures and configuration
- During processor evaluations the Nest has many design alternatives, is very workload variable, and is influenced by many workload and configuration factors
  - Whereas Private Area Cache is more stable for a workload (since not influenced as much by configuration)
- For this reason, the new LSPR workloads focus on the Nest



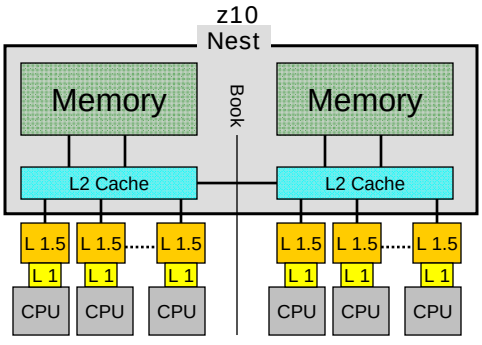
Nest View of z196 Processor

- Caches can be thought as divided into
  - Private Area Caches which are part of the processor design
    - Heavily influenced by instruction complexity and processor design
  - Shared Area Caches which are part of the memory hierarchy
    - This is called 'The Nest'
    - Heavily influence by workload mixtures and configuration



Evaluating Processor / Nest / Workload Relationship

- There is a desire to understand the variability of processor capacity relative to the workload 'usage' of the Nest
- Good News: The SMF 113 provides insight into the number of times L1 was sourced from the Nest
  - L2 local, L2 remote
  - Local Memory, Remote Memory
- Less Than Good News:
  The SMF 113 does not provide the penalty cycles for the individual levels of cache
  - Only total penalty cycles for all L1 sourcing (for I and D cache)



COUNTER SET= BASIC (Reminder)

- 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)

z10 New CPI Formulas (for Contention Index)

- Total Machine Cycles per Instruction (Actual CPI)

  CPI = (Total Cycles / Total Instructions)

  = (B0/B1)

- Estimate Finite CPI (Est Finite CPI) – as if finite cache/memory

  *EF_CPI = (Penalty Cycles / Total Instructions) * .84*

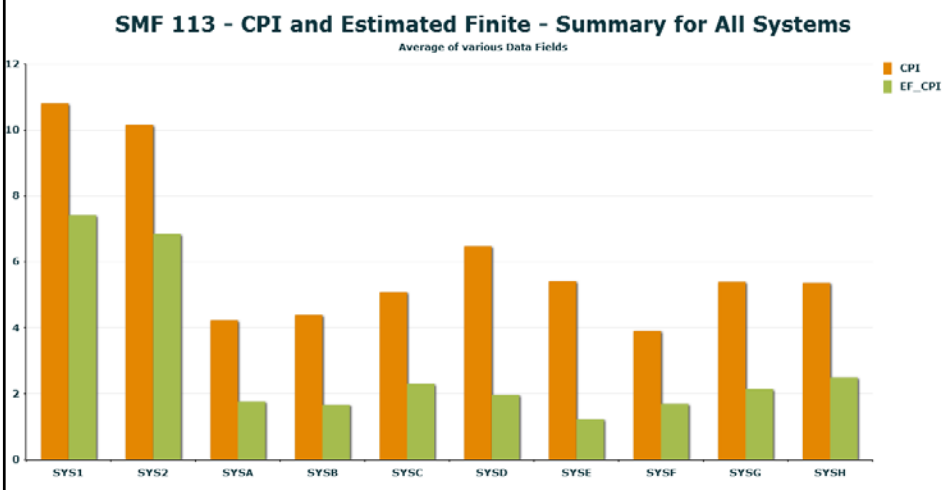
  *= ((B3+B5)/B1) * .84* *(where .84 is Gary King z10 Constant)*

  - Think of this as Penalty Cycles per instruction, but since there is an 'overlap' of sourcing cycles from the different levels, we need scale value downward to exclude these 'overlap' cycles
    - Thus the multiplication by King constant .84 for the z10 and (.57+(.1*RNI)) for z196
    - Note a lower value for z196 to show improved overlapping
- Estimated Instruction Complexity CPI (Est Instr Cmplx CPI)

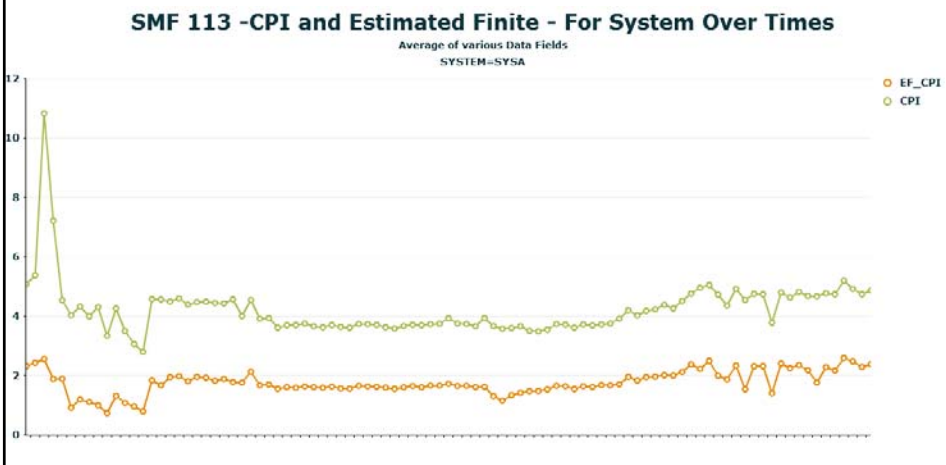
  EIC_CPI = ((CPI) – (EF_CPI))

  - Think of this as CPI if there was an infinitely large L1 cache (i.e. no penalty cycles)

CPI vs EF_CPI by System (z10)



CPI vs EF_CPI For System Over Time (z10)



z196 New CPI Formulas (for Contention Index)

- Total Machine Cycles per Instruction (Actual CPI)

  CPI = (Total Cycles / Total Instructions)

  = (B0/B1)

- Estimate Finite CPI (Est Finite CPI)

  EF_CPI = (Penalty Cycles / Total Instructions)

  *= ((B3+B5)/B1) ** (.57+(.1*RNI)) *(where scale is Gary King z196 estimate)*

  - Think of this as Penalty Cycles per instruction, but since there is an 'overlap' of sourcing cycles from the different levels, we need scale value downward to exclude these 'overlap' cycles
    - Thus the multiplication by King constant .84 for the z10 and .63 for z196
    - Note a lower value for z196 to show improved overlapping

- Estimated Instruction Complexity CPI (Est Instr Cmplx CPI)

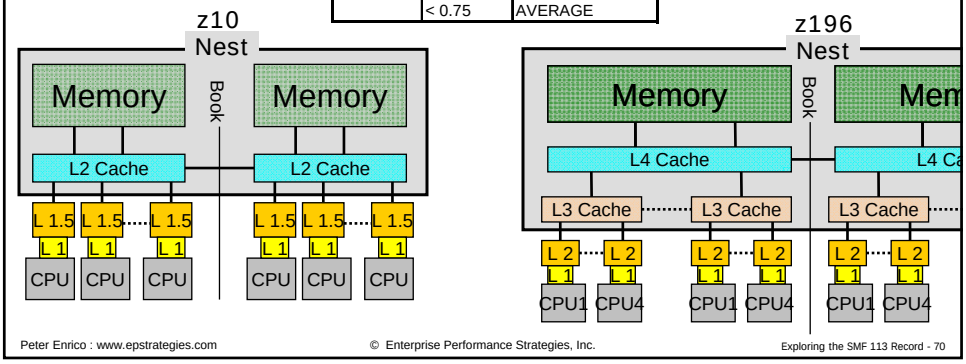
  EIC_CPI = ((CPI) – (EF_CPI))

  - Think of this as CPI if there was an infinitely large L1 cache (i.e. no penalty cycles)

Relative Nest Intensity

- Think of RNI as a 'stress factor' on the memory hierarchy
  - New LSPR workloads will be based on RNI

L1MP	RNI	Workload Hint
<3%	>= 0.75	AVERAGE
	< 0.75	LOW
3% to 6%	>1.0	HIGH
	0.6 to 1.0	AVERAGE
	< 0.6	LOW
>6%	>=0.75	HIGH
	< 0.75	AVERAGE



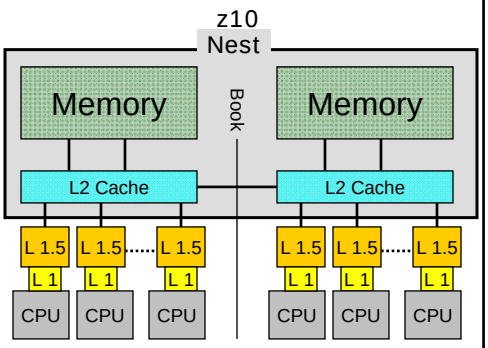
Relative Nest Intensity (for z10)

- *"Relative Nest Intensity reflects the distribution and latency of sourcing from shared caches and memory"* (J Burg, IBM)

  *RNI = ((1.0 * L2LP) + (2.4 * L2RP) + (7.5 * MEMP)) /100*

  (where weights are Gary King Constants)

- L2 Local Sourcing %
  - Weighted by 1.0
- L2 Remote Sourcing %
  - Weighted by 2.4
- Memory Sourcing % (Local + Remote)
  - Weighted by 7.5
- Note: L1.5 not considered since not part of Nest



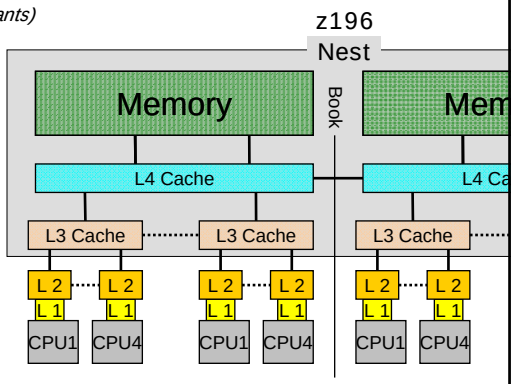
Relative Nest Intensity (for z196)

- *"Relative Nest Intensity reflects the distribution and latency of sourcing from shared caches and memory"* (J Burg, IBM)

  *RNI = 1.6 * ((0.4 * L3P) +*
  *(1.0 * L4LP) +*
  *(2.4 * L4RP) +*
  *(7.5 * MEMP)) /100*

  (where weights are Gary King Constants)

- Note: L2 not part of nest so not factored in
- Note benefit L3P relative to other caches



HD (Y/N) Before/After Example – RNI Small MP z10

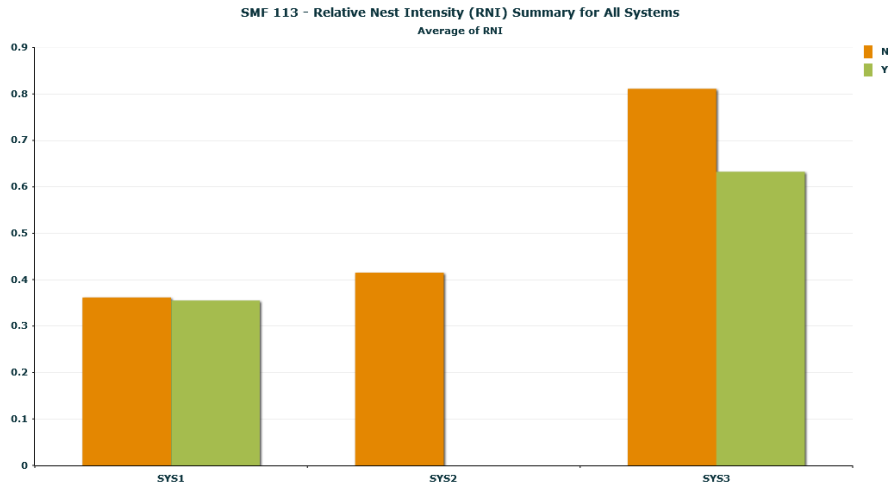


Chart created at www.pivotor.com

HD Before/After Example – RNI Small MP z10

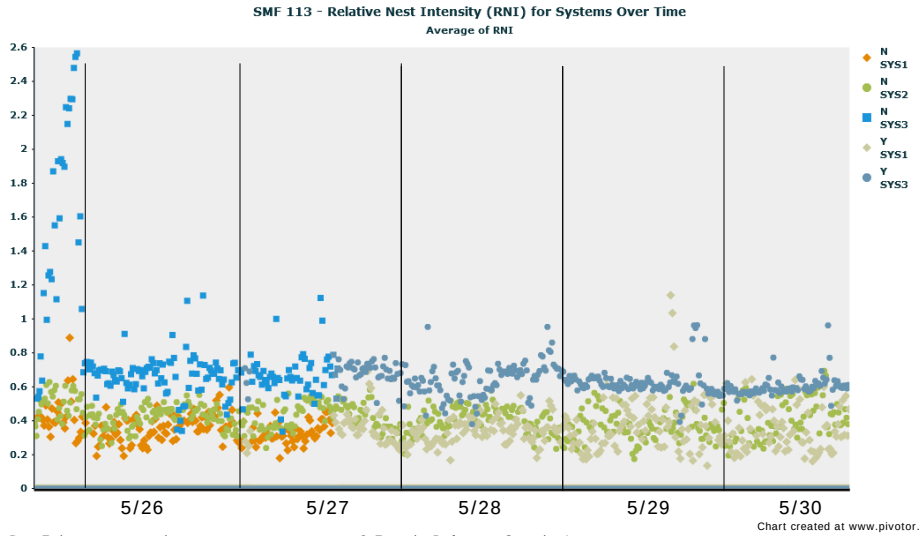


Chart created at www.pivotor.com

LSPRs and SMF 113s and RNI Hint

- SMF 113 measurements are now used to provide guidelines / hints for LSPR and zPCR processor sizing
- This RNI Hint table was documented in the Large System Performance Reference (LSPR)
  - Document Number SC28-1187-14
- The next slide shows an example of an LSPR chart used for processor sizing
- Using the SMF 113 records you now need to calculate
  - L1MP - L1 Miss Per 100 Instructions
  - RNI – Relative Nest Intensity
- Note: This table and these guidelines are expected to change as more is learned from the SMF 113 records

L1MP	RNI	Workload Hint
<3%	>= 0.75	AVERAGE
	< 0.75	LOW
3% to 6%	>1.0	HIGH
	0.6 to 1.0	AVERAGE
	< 0.6	LOW
>6%	>=0.75	HIGH
	< 0.75	AVERAGE

LSPR Table Example

IBM System z9 EC
(System z9 2094-701 = 1.00)

Processor	#CP	PCI**	MSU***	Low*	Average*	High*
2094-601	1	454	65	0.81	0.81	0.81
2094-602	2	880	127	1.6	1.57	1.53
2094-603	3	1303	184	2.38	2.33	2.23
2094-604	4	1720	240	3.13	3.07	2.92
2094-605	5	2109	292	3.87	3.77	3.58
2094-606	6	2482	339	4.59	4.43	4.21
2094-607	7	2842	385	5.3	5.08	4.81
2094-608	8	3188	428	5.99	5.69	5.37
2094-701	1	560	81	1	1	1
2094-702	2	1086	158	1.98	1.94	1.89
2094-703	3	1607	229	2.93	2.87	2.75
2094-704	4	2122	298	3.86	3.79	3.6
2094-705	5	2601	363	4.78	4.65	4.42
2094-706	6	3062	422	5.67	5.47	5.19
2094-707	7	3505	479	6.54	6.26	5.93
2094-708	8	3932	532	7.38	7.02	6.62

Estimated Sourcing Cycles per L1 Miss Per 100 Instr

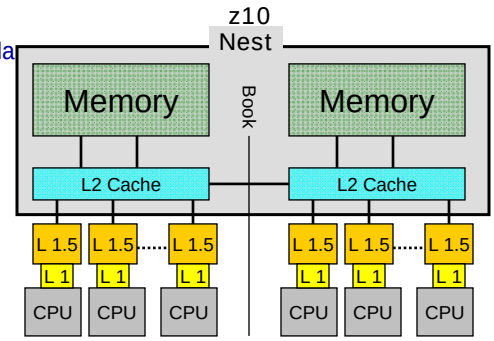
- Estimated Sourcing Cycles per L1 Miss (from the Nest)

  Est_SCPL1M = (Penalty Cycles / Penalty Writes)

  *= ((B3+B5)/(B2+B4)) * X* *(where .84 is Gary King z10 Constant)*

  - Since penalty cycles and writes include sourcing from L1.5, and since we are only interested in the Nest, we need to multiple by .84 to compensate 'Sourcing Cycles per L1 Miss' downward

- X = Scaling factor for sourcing overla
  - z10: X = .84
  - z196: X = (.57+(0.1 * RNI))
  - Note: Scaling factors still being refined by IBM



Presentation Summary

- SMF 113 processor cache counters will become more crucial over time
  - Currently mostly be used as input to zPCR for processor sizing exercises
- It is recommended that you enable SMF 113 data collection
  - Very low overhead
  - Collect regularly
- Watch out for APARs that will announce enhancements
- Please do not hesitate to ask me questions about this important subject!