SMF 113
Processor Cache Counter Measurements - Overview, Update, and Usage

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

Creators of Pivotor®

•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

Performance Workshops Available

During these workshops you will be analyzing your own data!

- WLM Performance and Re-evaluating of Goals
  - Instructor: Peter Enrico and Scott Chapman
  - September 28 – October 2, 2015 – Columbus, Ohio, USA
- Parallel Sysplex and z/OS Performance Tuning
  (Web / Internet Based!)
  - Instructor: Peter Enrico and Scott Chapman
  - November 17 – 19, 2015
- Essential z/OS Performance Tuning Workshop
  - Instructors: Peter Enrico, Scott Chapman, Tom Beretvas
  - October 19 - 23, 2015 – Dallas, Texas, USA
- z/OS Capacity Planning and Performance Analysis
  - Instructor: Ray Wicks

EPS Sessions at Share

Peter Enrico

Day	Time	Location	Presentation
Wed	11:15	Asia 3	SMF 113 Processor Cache Counter Measurements – Overview, Update, and Usage
Wed	1:45	Asia 3	WLM – Effective Setup and Usage of WLM Report Classes
Thu	11:15	Asia 3	zProcessor Consumption Analysis (including z13), or What is Consuming All the CPU?

Scott Chapman

Day	Time	Location	Presentation
Tue	11:15	Asia 3	Memory Management in the TB Age
Tue	3:15	Southern Hemisphere 4	Lessons Learned from implementing an IDAA
Fri	11:15	Asia 3	WLM in One Page

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:

Abstract and Reports Offer

- **Abstract**

  - The 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 the usage of the processor caches have on the MIPS capacity of a processor. 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. Discussed will be the concept and importance of RNI, L1MP, and several other important performance indicators. Also discussed will be the latest updates and uses of the SMF 113 processor cache counter measurements.

Why do we care about processor cache measurements and usage of the caching hierarchy?

Key Influences of Processor Performance and Capacity

- Question: What are the key influences that result in variations of a particular processor's delivered capacity relative to a customer's environment and workload?

- That is:
  - Why is it when you size a processor using zPCR or IBM's LSPR charts, your results may vary from zPCR or IBMs?
  - Why could the same machine's capacity vary based on a particular workload?
  - Why could the same machine's capacity vary based on a particular customer?
  - Why could the same machine's capacity vary based on a particular configuration setting in z/OS, WLM, PR/SM, etc?

Key Influences of Processor Performance and Capacity

- Question: What are the key influences that result in variations of a particular processor's delivered capacity relative to a customer's environment and workload?

- Answer: As Gary King of IBM would say… there are three key influences:
  - Instruction complexity of one processor family to another
  - Path length of the code executed by customer applications and transactions
  - Usage of the Memory Hierarchy

- A machine's capacity will vary based on each of these three factors

Key Influence - Instruction Complexity

- Instruction complexity of one processor family to another
  - Types of instructions, the sequence in which they are executed, and the way they interact with the processor design
  - Many machine design alternatives affect instruction complexity
    - Each processor family has variations in the chip design
  - As processor evolve the way the chip executes instructions is enhance and geared towards the technology
  - Examples include: Cycle time of CPU, how the instructions are wired to execute (using pipelining, branch prediction, out of order execution, etc.)
  - Influences the workload
    - Once a customers workloads are on a processor, instruction complexity is relatively constant between customers and workloads
    - In other words, relative to the LSPRs and sizing, once a move is made to the new processor family instruction complexity does not vary much from one customer to the next.

Key Influence - Instruction Complexity

- Example of LSPRs for a 704 of different architectures

(System z9 2094-701 = 1.00)

	Processor	#CP	PCI**	MSU***	Low*	Average*	High*
z9	2094-704	4	2,122	298	3.86	3.79	3.60
z10	2097-704	4	3,192	401	6.17	5.70	5.07
z196	2817-704	4	4,320	531	8.06	7.72	7.08
zEC12	2827-704	4	5,409	664	10.36	9.66	8.73
z13	2964-704	4	6,041	592	11.93	10.79	9.40

**PCI = Processor Capacity Index (a.k.a. MIPS)

Key Influence – Path Length

- Path length of the code executed by customer applications and transactions
  - This relates to code executed by applications / jobs / transactions / etc.
    - Instruction count
  - The actual path lengths executed by a workload will vary
    - From customer to customer, and from IBM synthetic workloads versus customer
    - From one customer's application environment versus another application environment of that same customer
      - Example: CICS / DB2 application versus a WAS / DB2 application
  - Is sensitive to the configuration due to MP effects
    - Higher n-ways or difference in configuration may increase path lengths execute (which in turn influences the processor capacity relative to LSPRs)
      - Example: May have more locking in a higher MP environment, or queues may be longer, etc.
  - But when move from one processor to another this generally does not change much for a specific customer
    - Whether the move is from one processor family to another
    - Or from one process in the same family to another

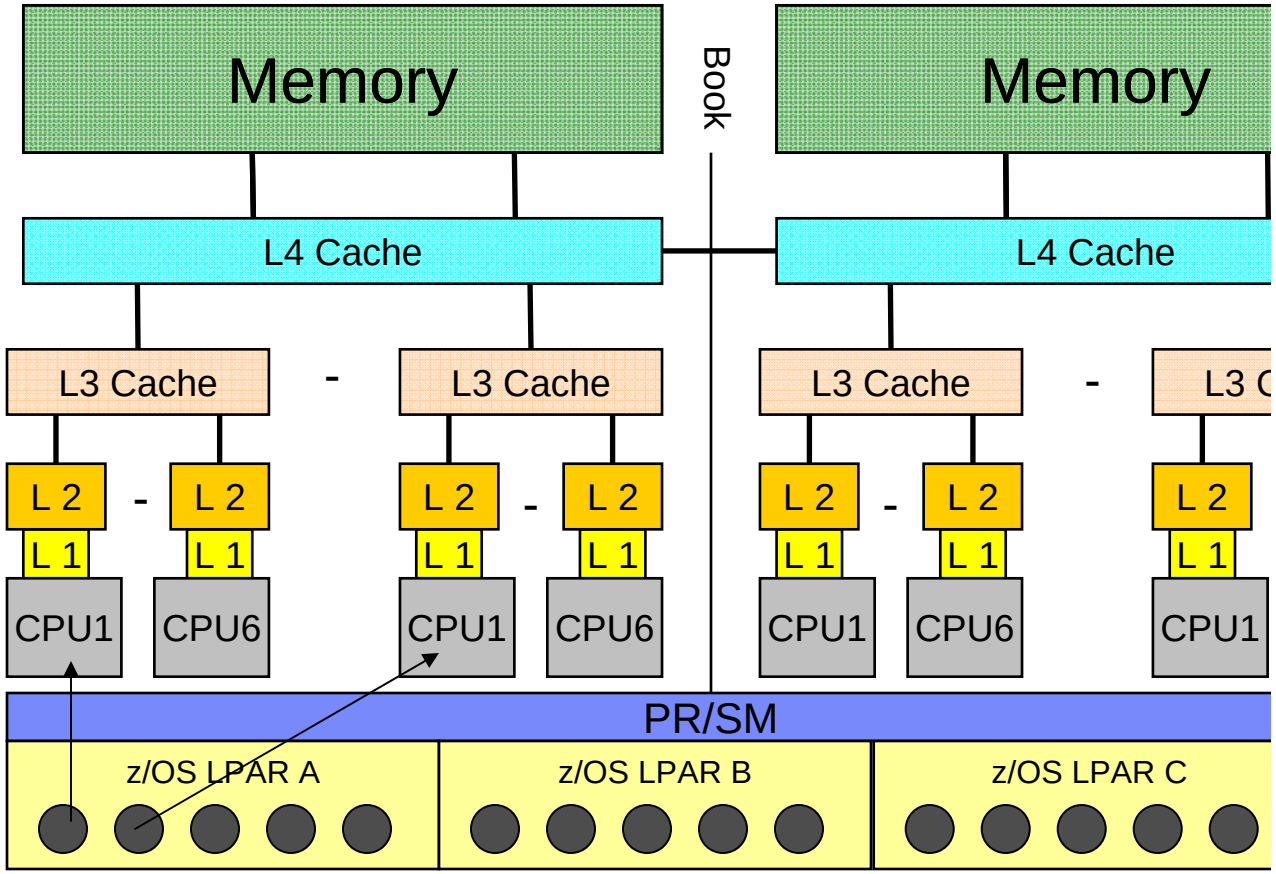
Key Influence – Memory Hierarchy

- Usage of the Memory Hierarchy
  - Heavily influenced by key factors result potentially wide variations in realized capacity
  - From one processor family to another there are many design alternatives
    - Levels of cache, scope of cache, latency, etc.
  - Configuration will influence usage of the memory hierarchy
    - LPAR configuration, competition between LPARs, options such as HiperDispatch, etc.
  - Exploitation by workloads will influence usage of the memory hierarchy
    - Transaction intensity, memory intensity, I/O intensity, application mixtures, competition of resource by applications, etc.
  - z/OS performance management and options
    - WLM management of resources, affinity nodes, IEAOPTxx opts, heap sizes, initiators, etc.
- Final result is that usage of memory hierarchy heavily influences a processor's delivered capacity and performance.
  - Workload performance sensitive to how deep into the memory hierarchy the processor must go to retrieve instructions and data
- So for processor sizing, LSPRs have started focusing on this

Introducing the Processor Caches of IBM's zArchitecture Processors

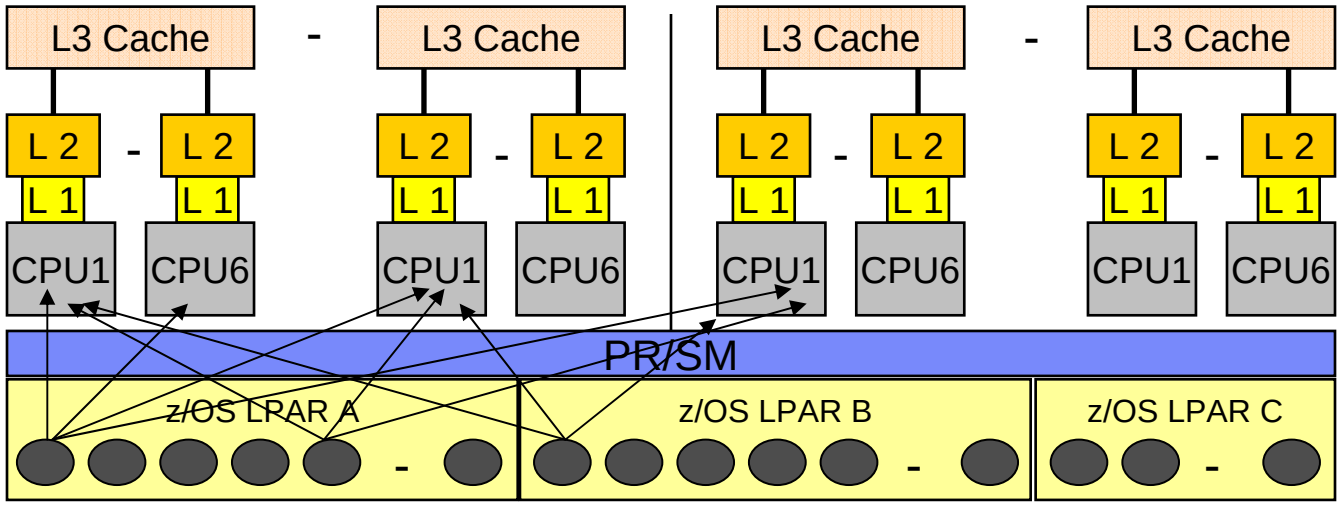
Performance Analyst View of zEC12 Processor (5.5 GHz)

- It take more cycles to fetch information from further up in cache hierarchy
  - L1 (Private level)
    - Data: 96KB
    - Instruction: 64KB
  - L2 (Private level)
    - Unified for Data and Instruction
    - 1MBinstr / 1MBdata
  - L3
    - Unified for Data and Instruction
    - 48MB / chip
  - L4
    - 384MB / book
  - Memory

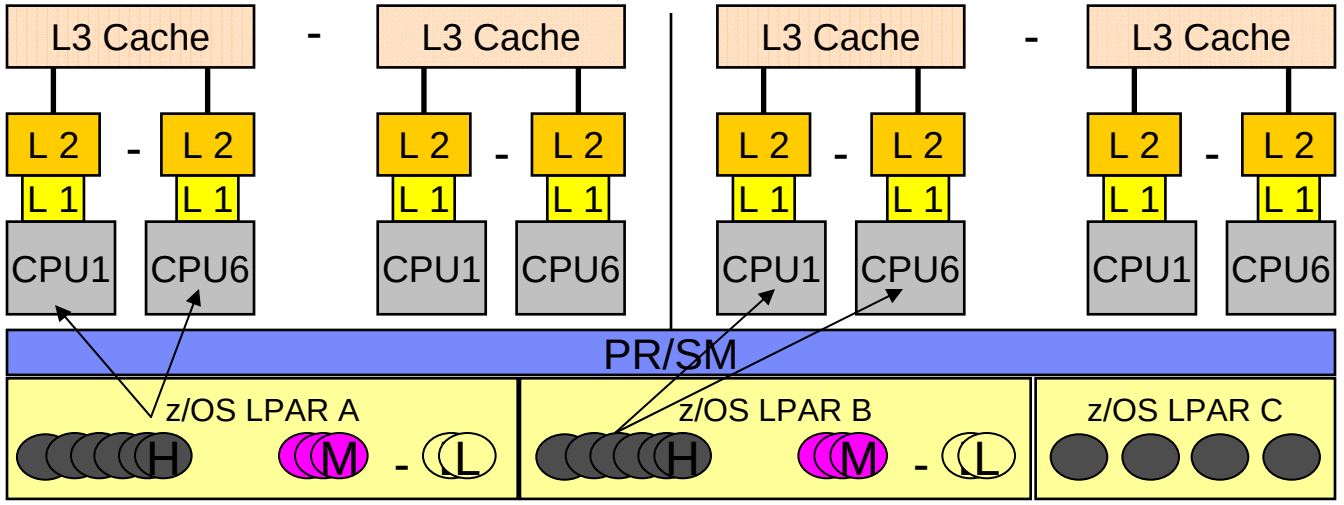


Vertical versus Horizontal CPU Management

- Vertical



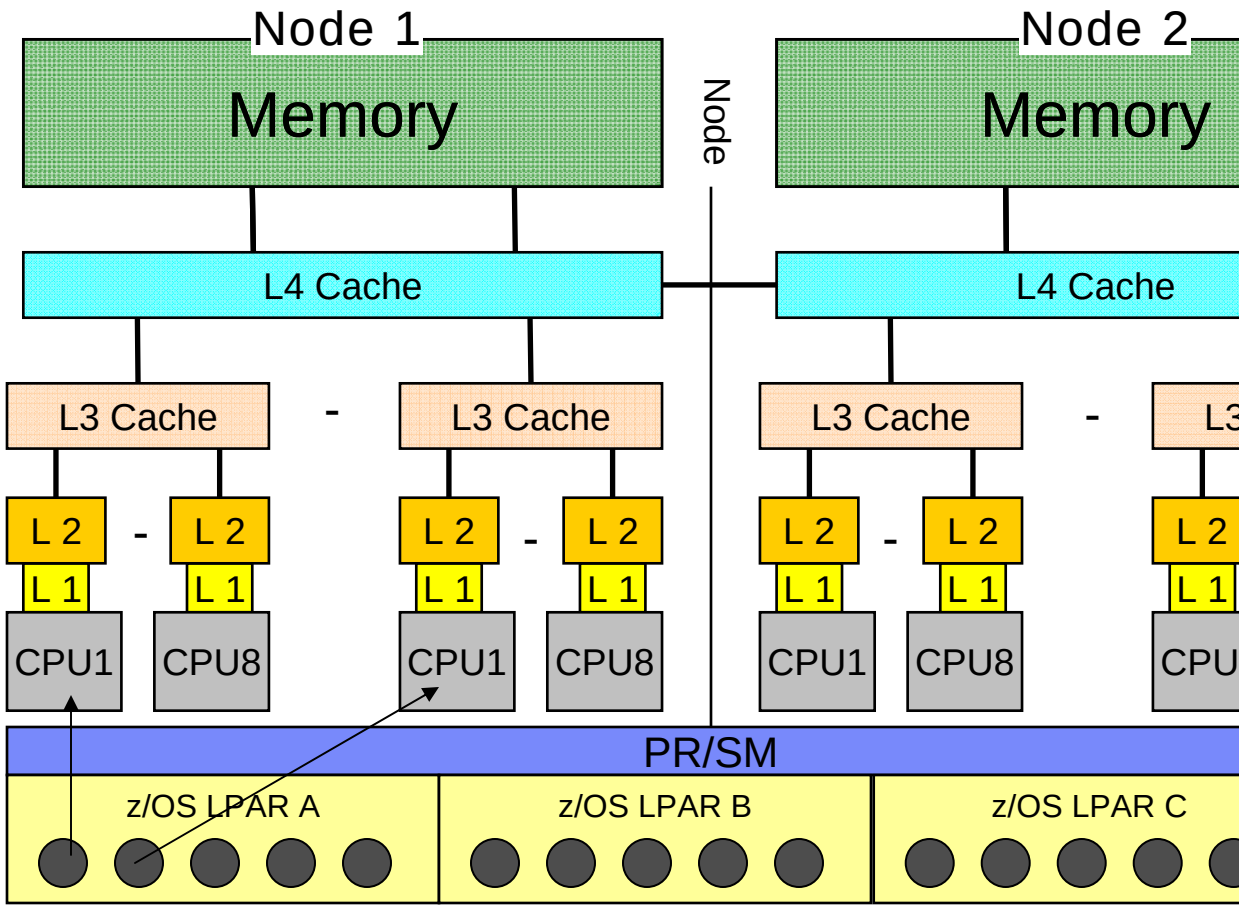
- Horizontal



Arranged into High, med, low Pools by PR/SM, and affinity nodes by z/OS

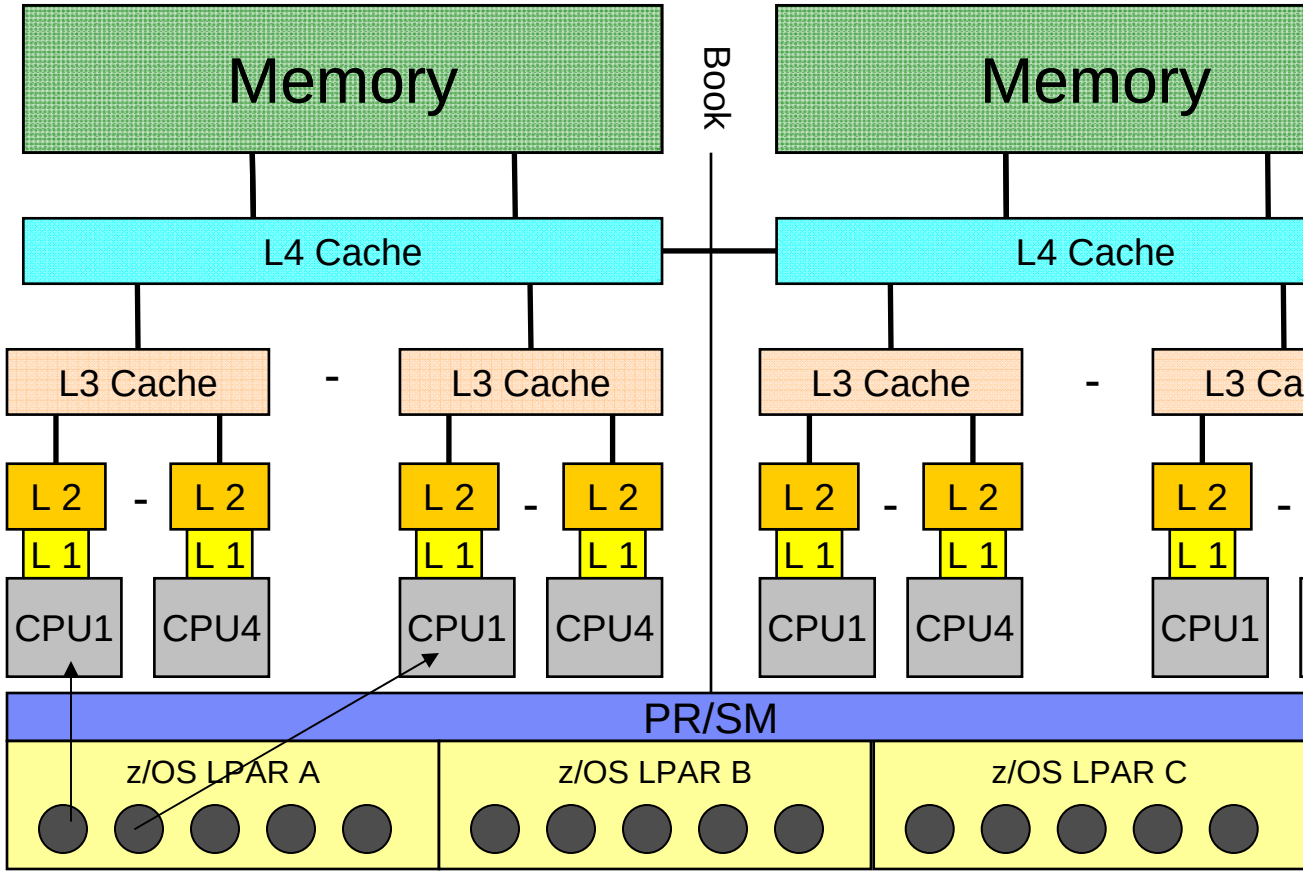
Performance Analyst View of z13 Processor (5.0 GHz)

- It take more cycles to fetch information from further up in cache hierarchy
  - L1 (Private level)
    - Data: 128KB
    - Instruction: 96KB
  - L2 (Private level)
    - Unified for Data and Instruction
    - 2MBinstr / 2MBdata
  - L3
    - Unified for Data and Instruction
    - 64MB / chip
  - L4
    - 420MB / node
      - Plus 224 MB L3 NIC Directory
  - Memory



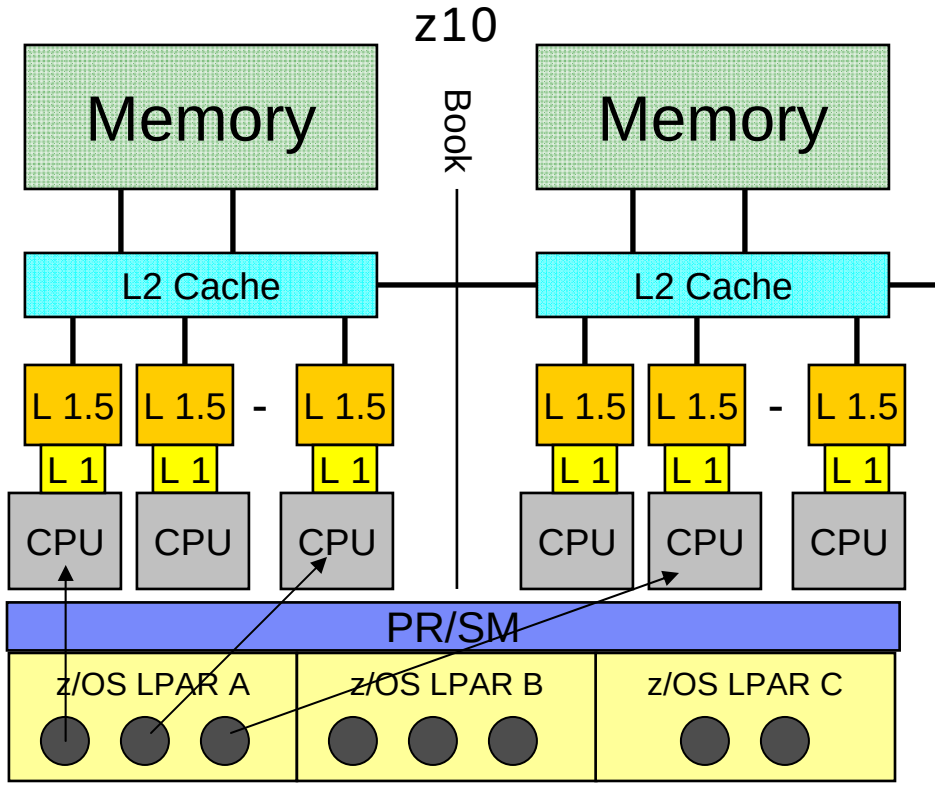
Performance Analyst View of z196 Processor (5.2 GHz)

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



Performance Analyst View of z10 Processor (4.4 GHz)

- 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

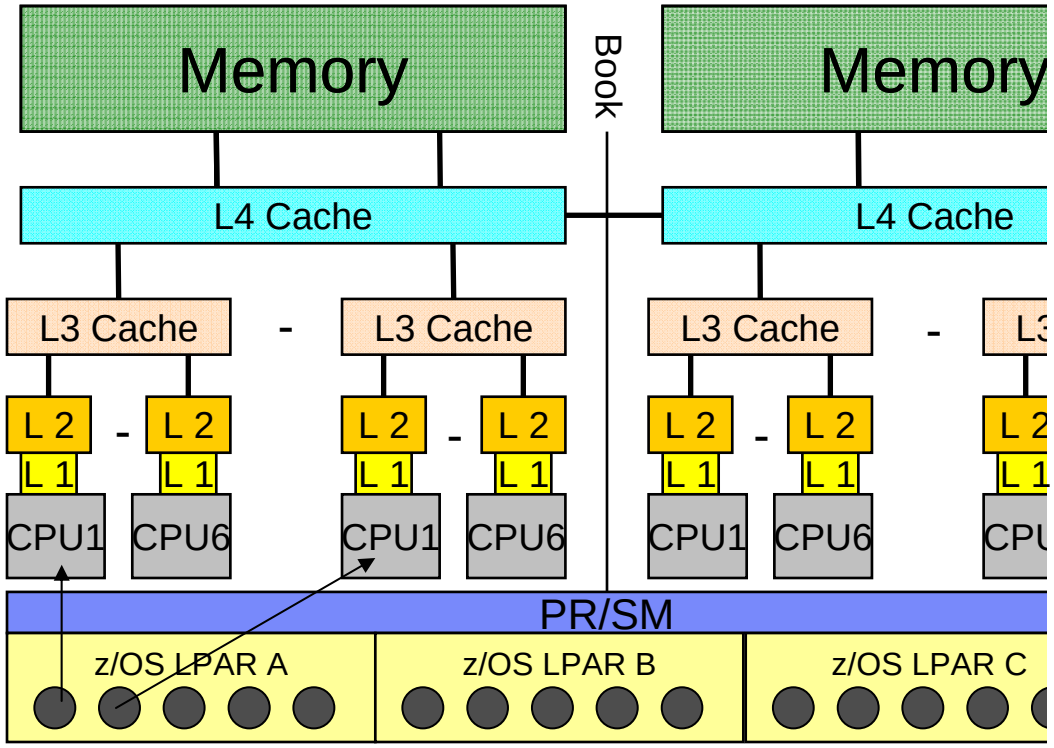


Greatest Usage of SMF 113

- To understand the variability of a machine's capacity based on the usage of the processor cache memory hierarchy
  - IBM's LSPR Workloads

- 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 (old way)

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

LSPRs and SMF 113s and RNI Hint (a measure of usage of memory hierarchy)

- 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 – Post SMF 113

zEnterprise 196
(System z9 2094-701 = 1.00)

Processor	#CP	PCI**	MSU***	Low*	Average*	High*
2817-701	1	1202	150	2.14	2.15	2.06
2817-702	2	2272	281	4.15	4.06	3.78
2817-703	3	3311	408	6.13	5.92	5.46
2817-704	4	4320	531	8.06	7.72	7.08
2817-705	5	5300	650	9.96	9.47	8.66
2817-706	6	6251	766	11.82	11.17	10.19
2817-707	7	7175	879	13.65	12.82	11.68
2817-708	8	8072	988	15.44	14.42	13.12
2817-709	9	8943	1091	17.19	15.97	14.52
2817-710	10	9788	1191	18.92	17.49	15.88
2817-711	11	10609	1286	20.61	18.95	17.21
2817-712	12	11407	1381	22.27	20.38	18.49
2817-713	13	12181	1473	23.89	21.76	19.74
2817-714	14	12932	1562	25.49	23.1	20.95
2817-715	15	13662	1648	27.06	24.41	22.12
2817-716	16	14371	1731	28.59	25.67	23.26

Introducing the SMF 113

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)
    - zEC12: 5500 Cycles/Mic (i.e. 5.5 GHz)
    - z13: 5000 Cycles/Mic (i.e. 5.0 GHz)

- For knee capped processors
  - Will reflect the reduced speed
  - But zIIPs and zAAP on the machine will show full speed numbers

Example of CPU Speed zED12
- 16 full speed CPs with 3 full speed zIIPs

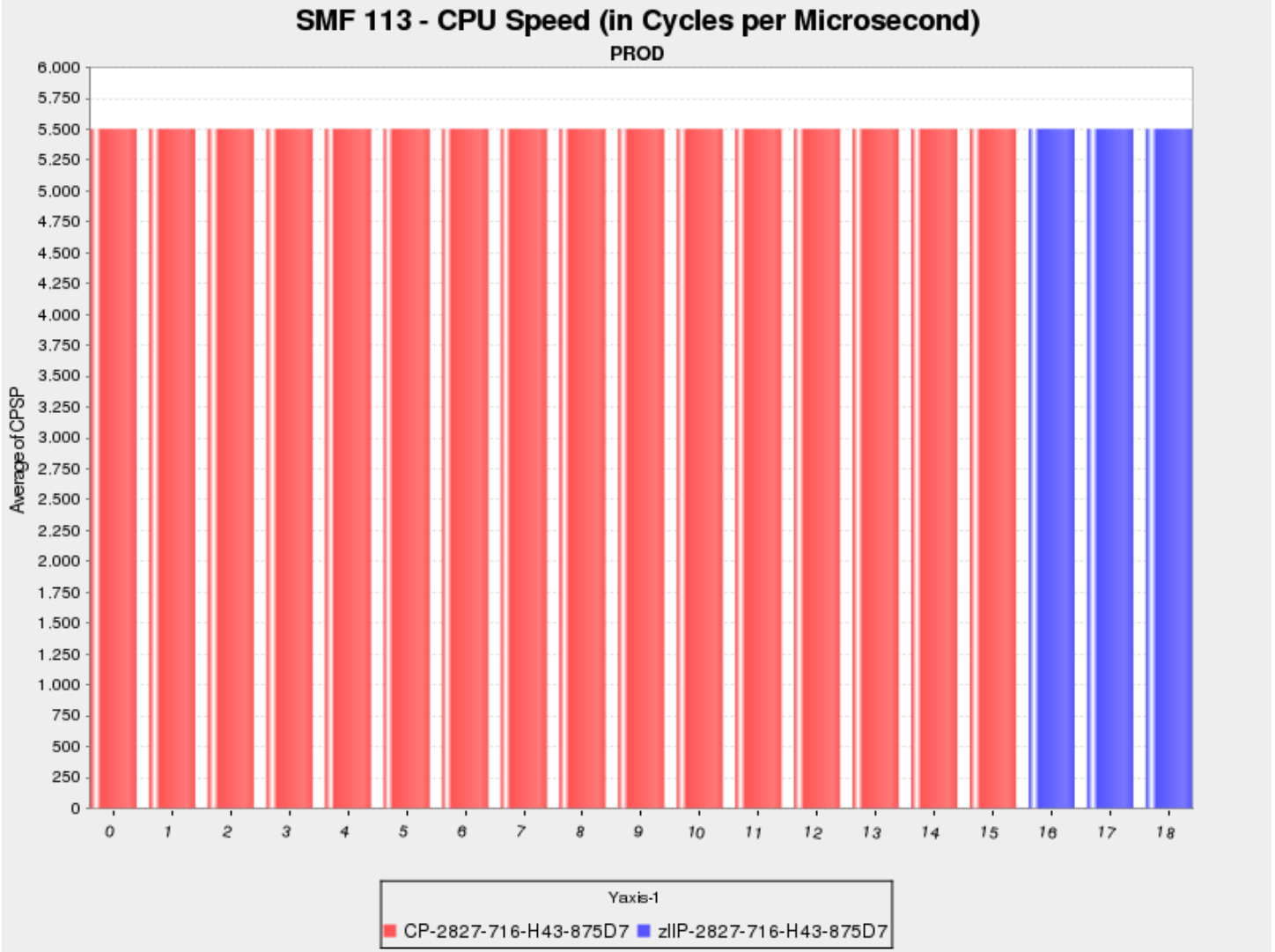


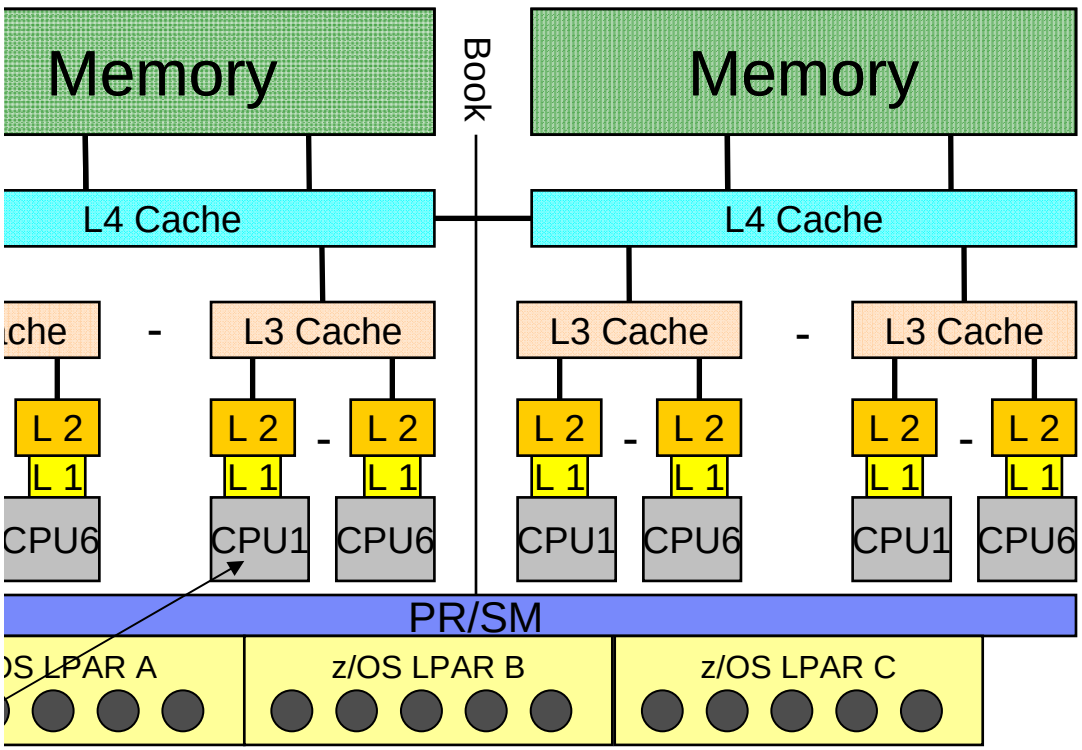
Chart created at www.pivotor.com

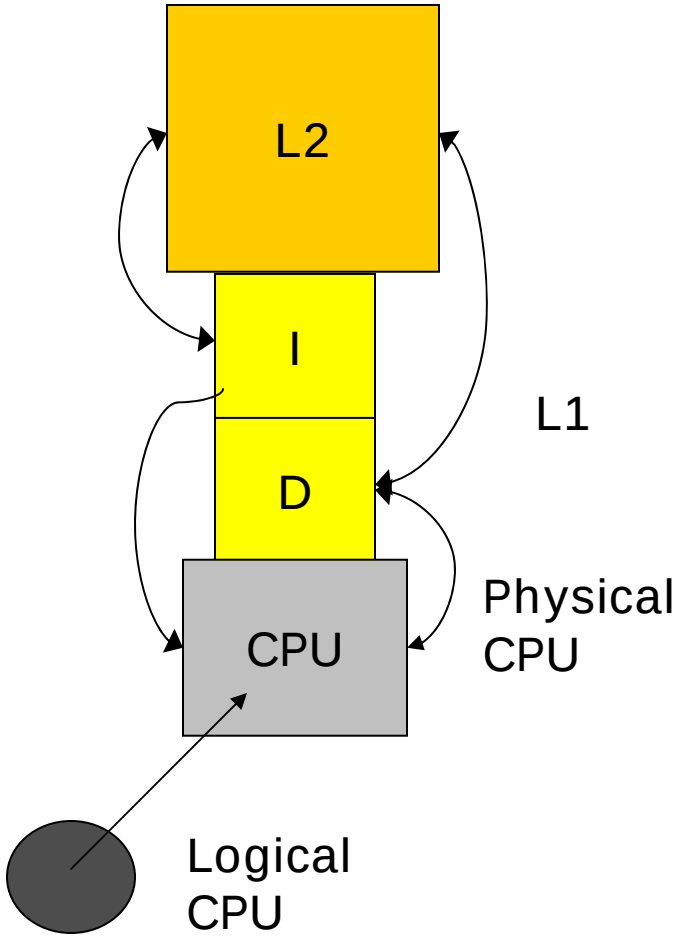
Counter Sets for z Machines Stored in SMF 113

- 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)

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





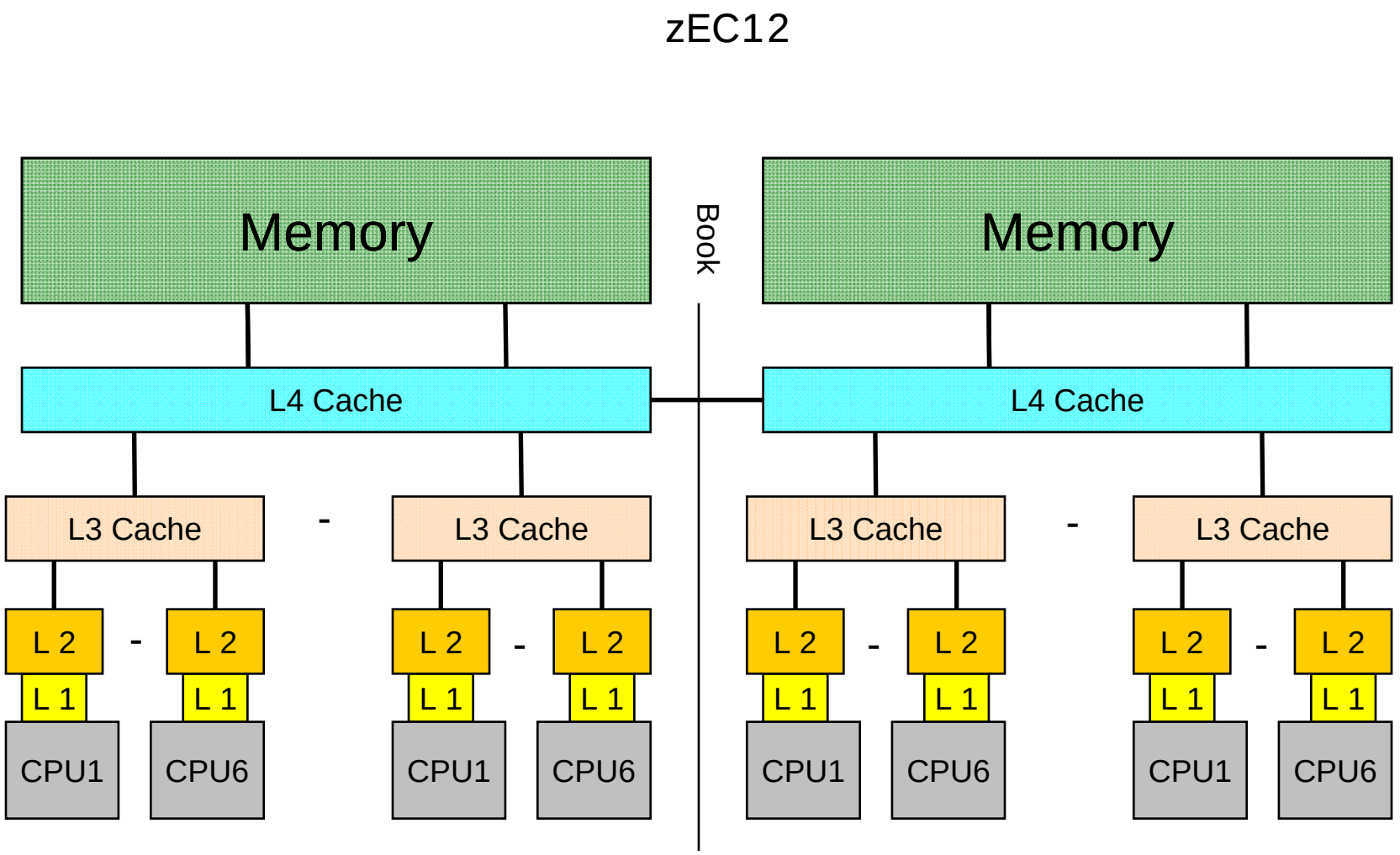
COUNTER SET= BASIC

- 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
  - Counter in which the SMF 30 Instruction counts are based
- 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= PROBLEM-STATE

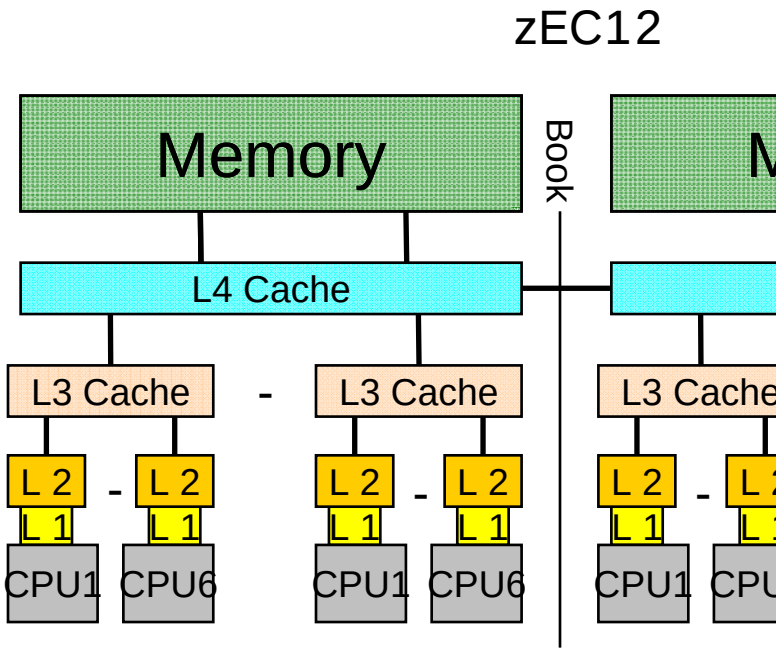
- Activity count for CPU when in both problem state
  - Counters for general purpose processors, zIIPs, and zAAPs
- 32: PROBLEM-STATE CYCLE COUNT
  - Number of CPU cycles, excluding the number of cycles CPU is in wait state
- 33: PROBLEM-STATE INSTRUCTION COUNT
  - Number of problem state instructions 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)

zEC12 and z196 Extended Cache Counters



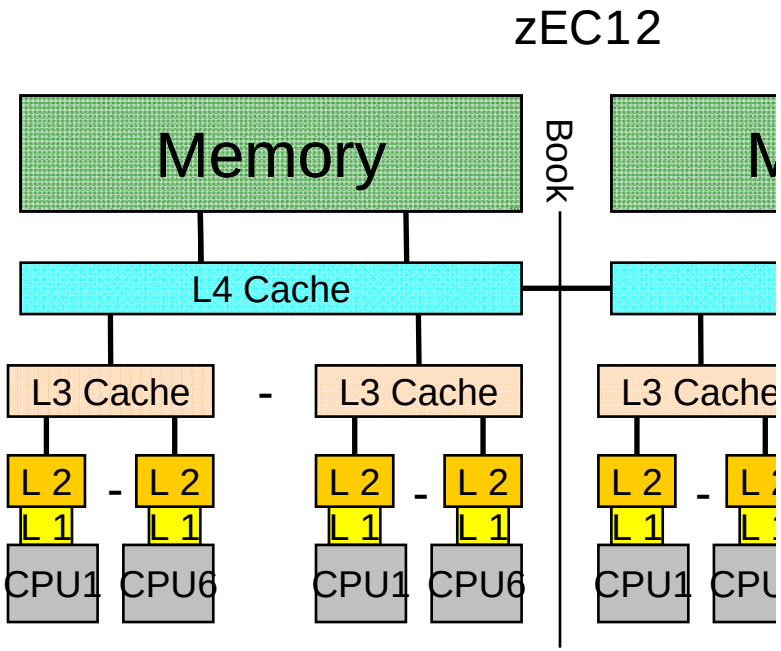
z196 and zEC12 Extended Cache Counters

- 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 On Chip On Book cache movement
  - 150: Dir write to L1 D-cache dir from L3 on chip on book cache (Data)
  - 153: Dir write to L1 I-cache dir from L3 on chip on book 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)
  - 155: Dir write to L1 I-cache dir from L3 off chip on book cache (Instruction)
  - Used for L4LP calculation
- 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)
  - Used for L4RP calculation



z196 and zEC12 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

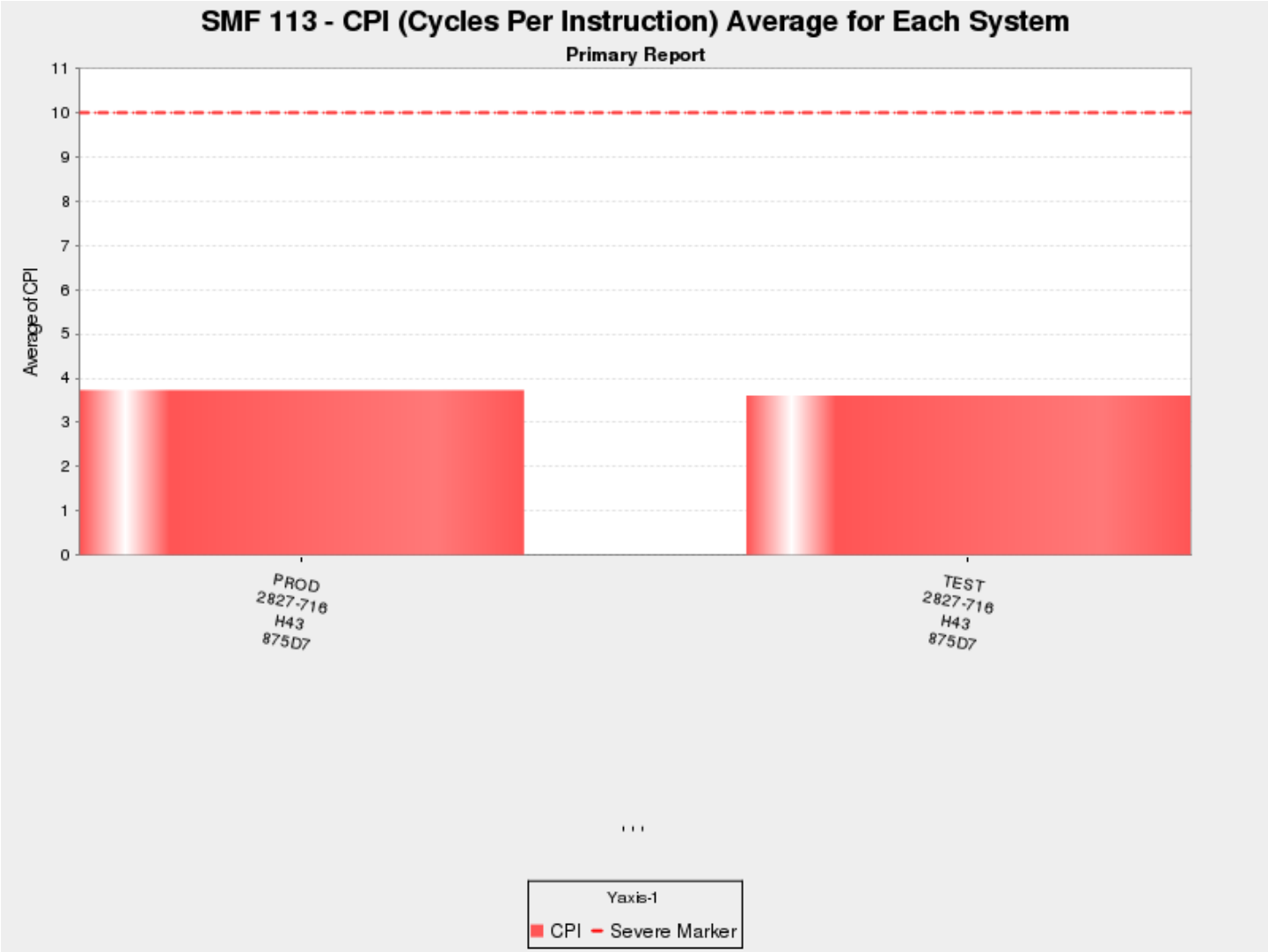
CPI – Cycles Per Instruction

- 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

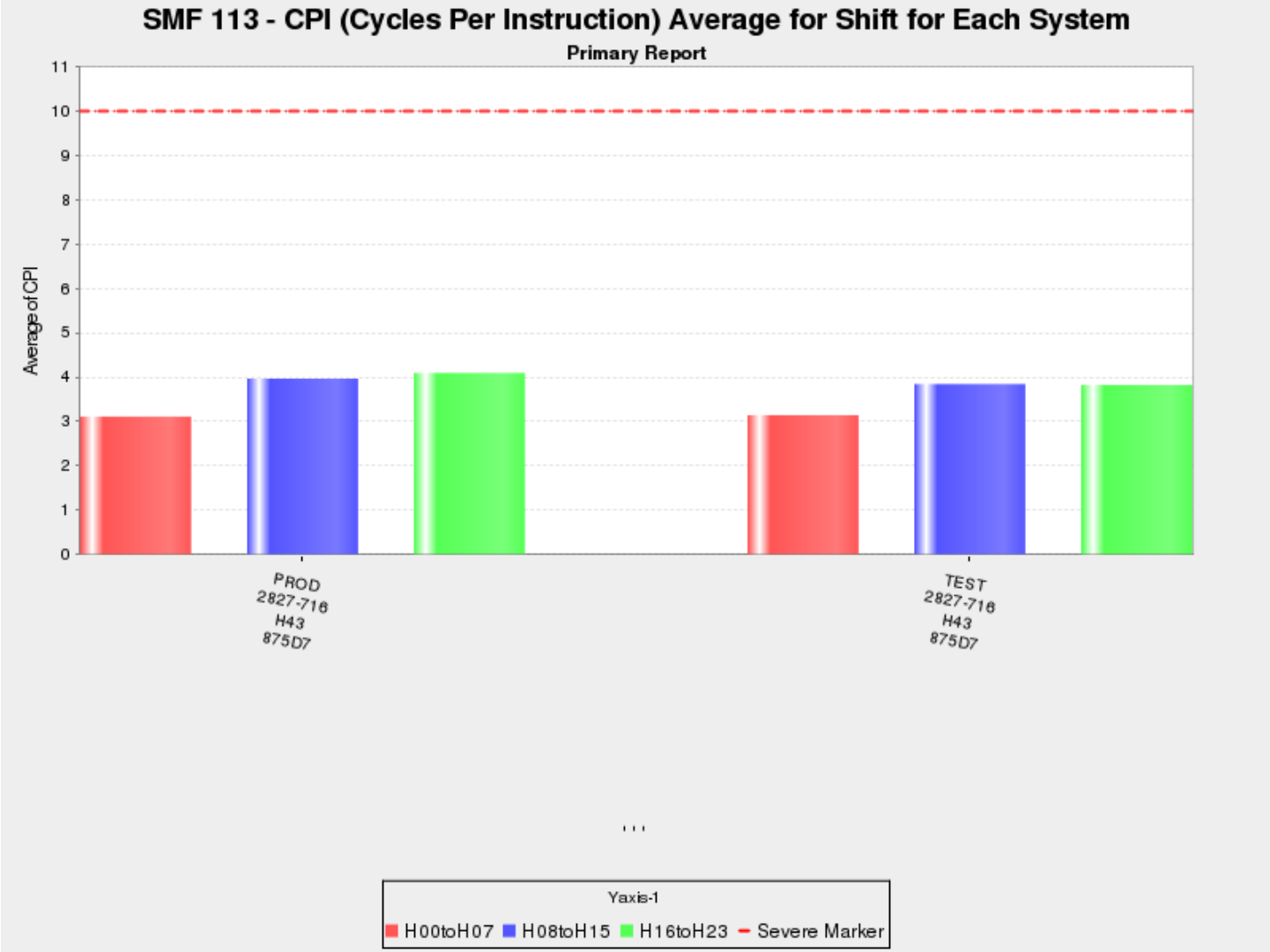
CPI = (Total Cycles / Total Instructions)

= (B0/B1)

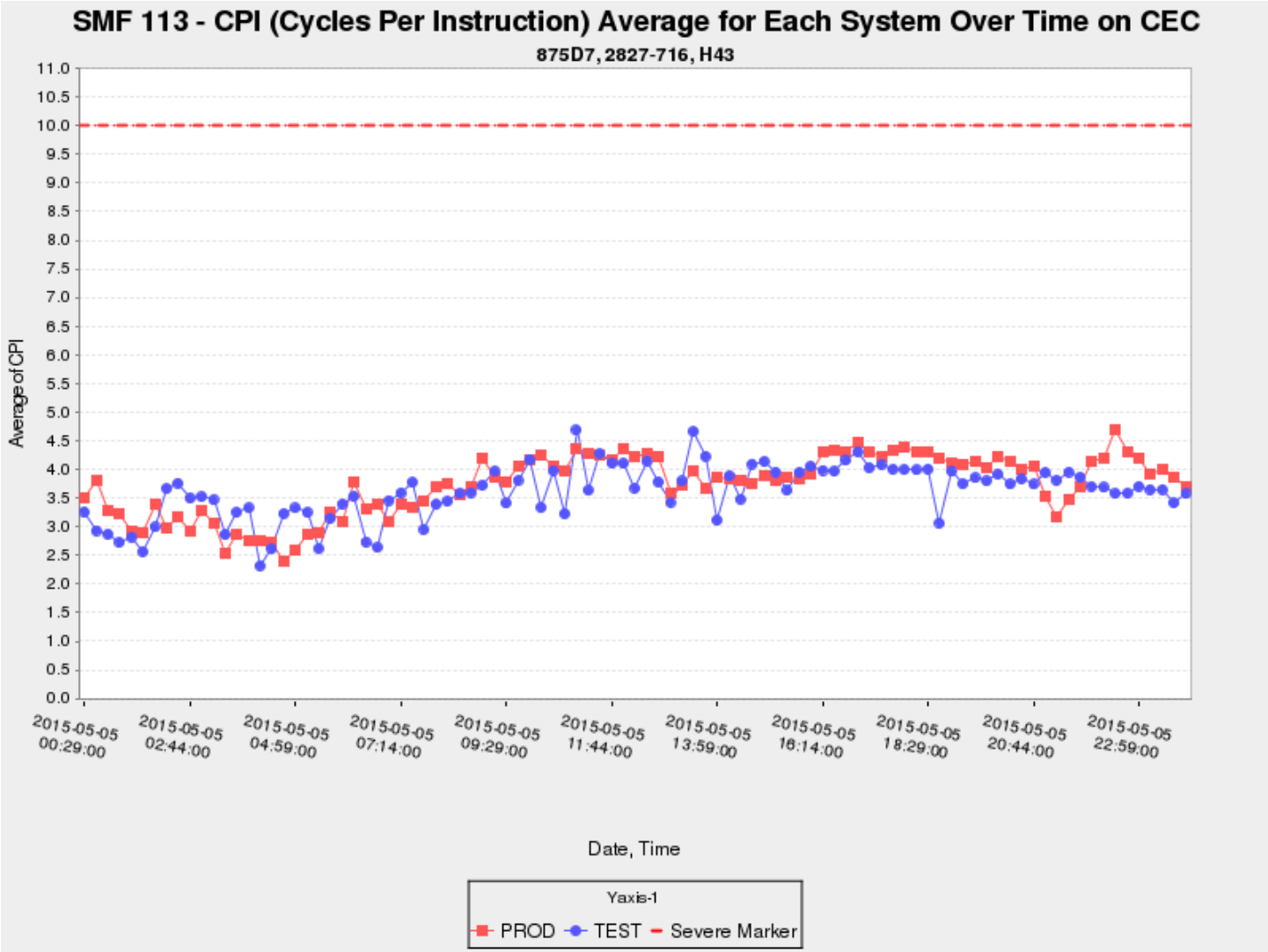
For each System, Average CPI Over 24 Hours



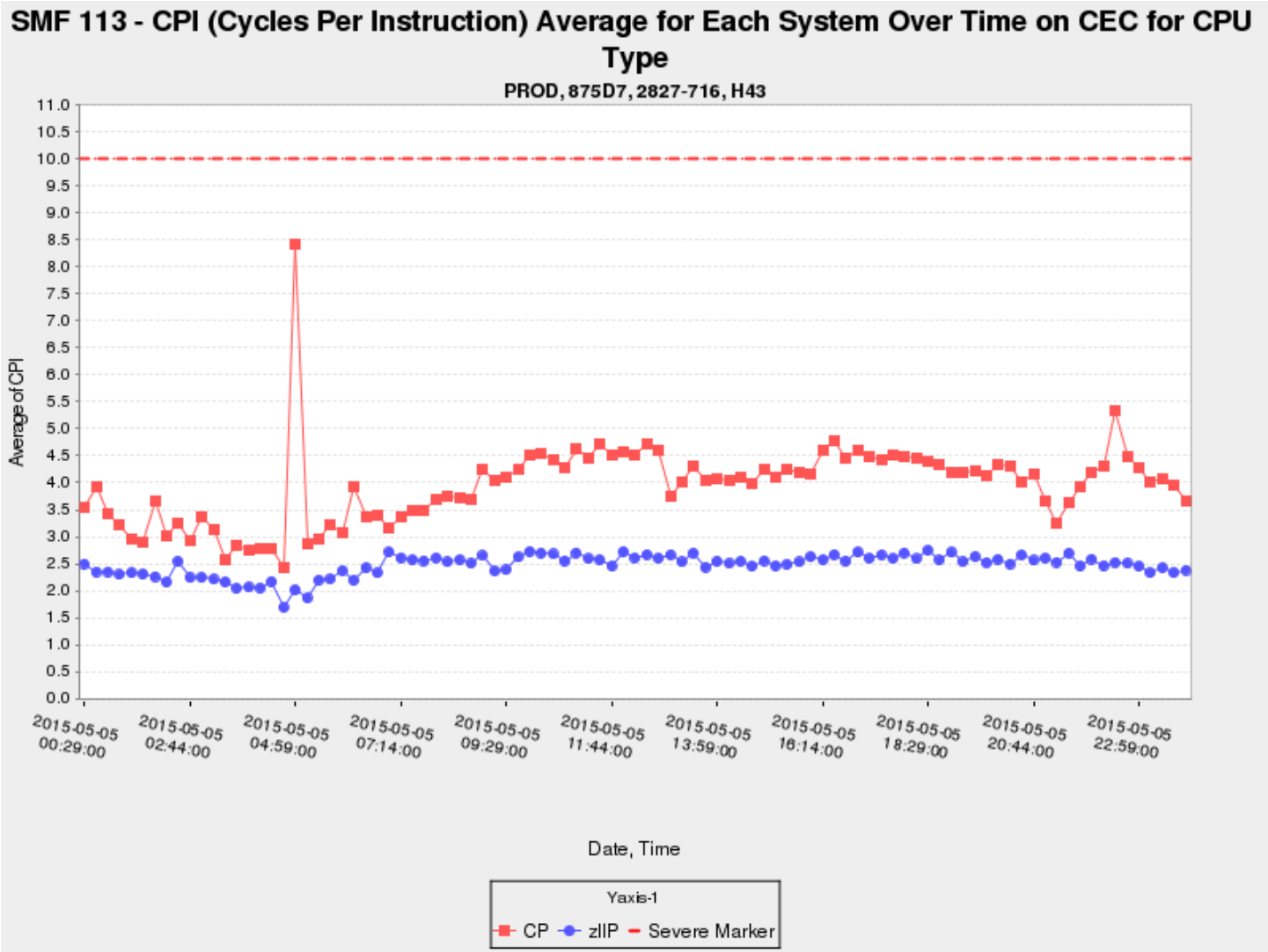
For each System, Average CPI for Each Shift



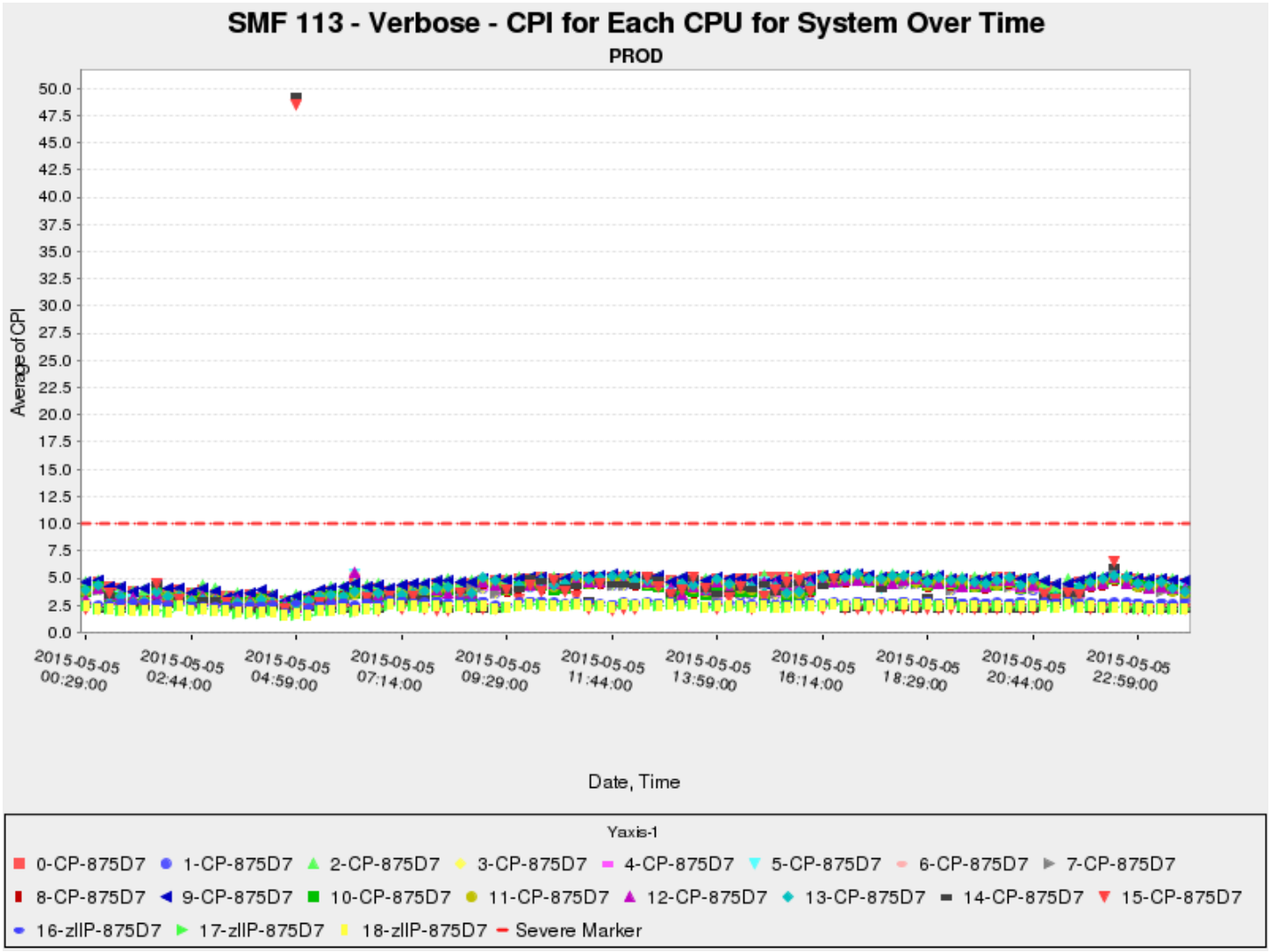
For each System, CPI Over Time



For System PROD, CPI by CPU Type



For System PROD, CPI by Engine Over Time



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

  Executed 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!

Executed MIPS For Each System Over Time

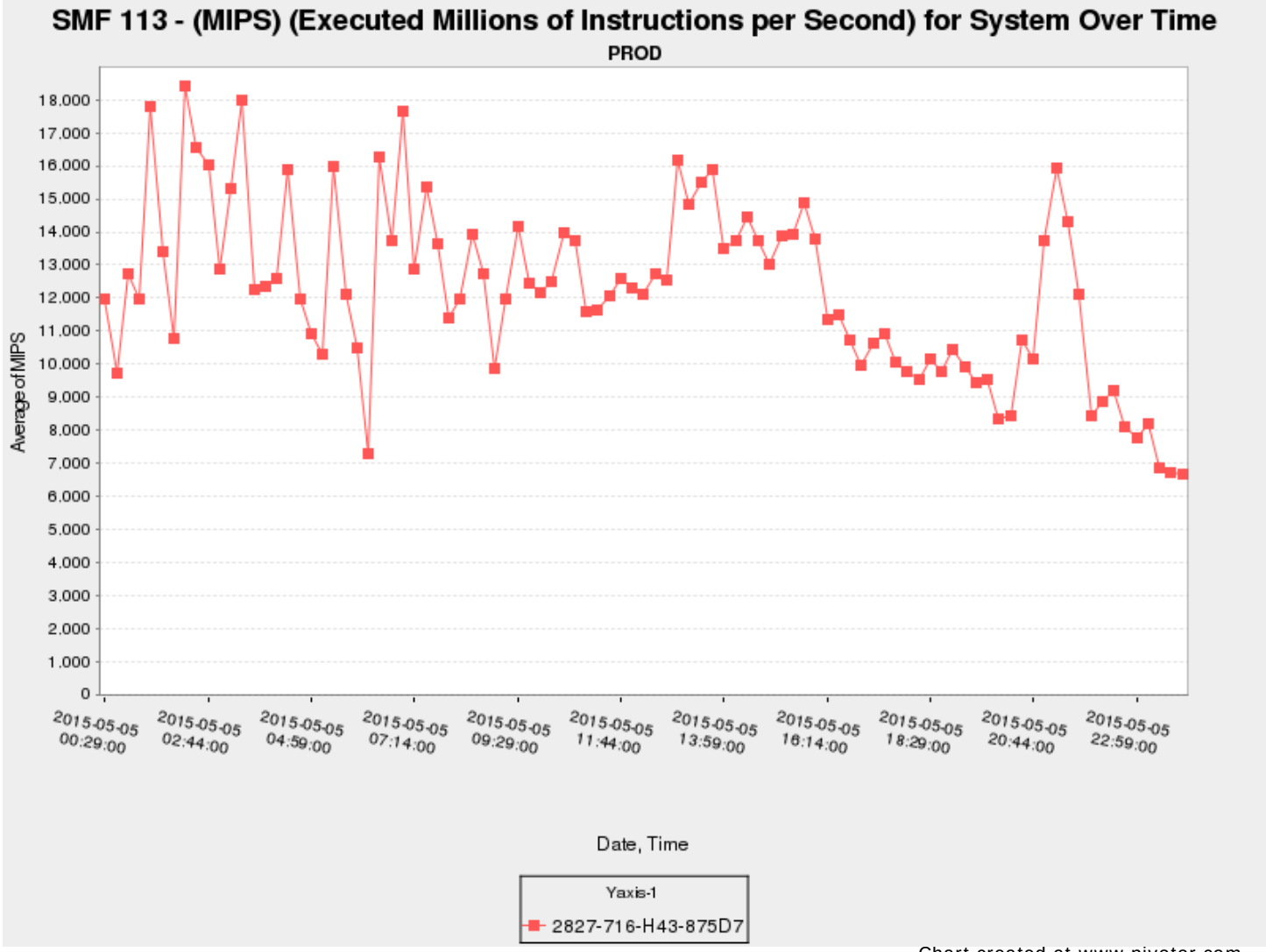


Chart created at www.pivotor.com

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

For each System, Average L1MP Over 24 Hours

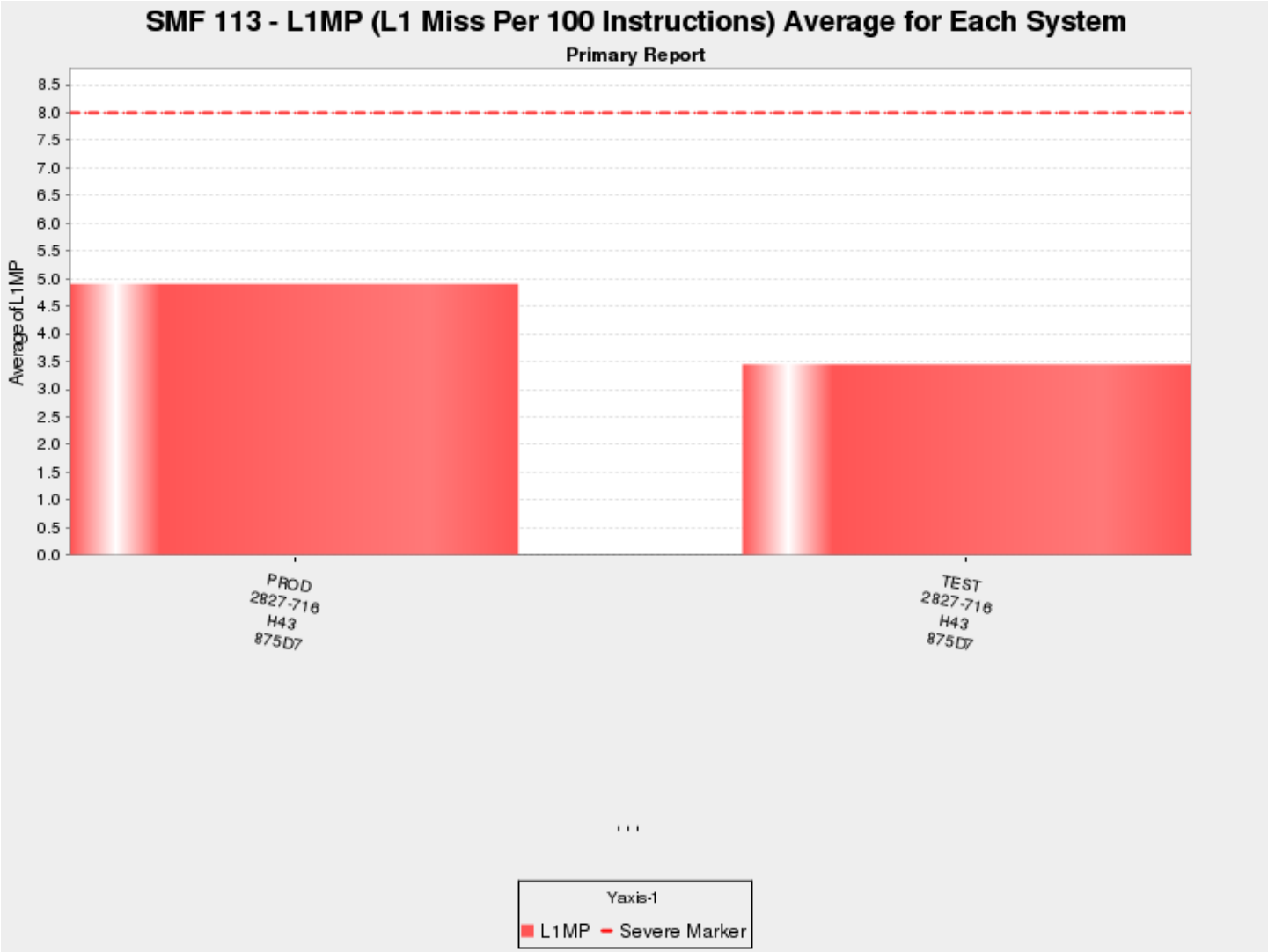


Chart created at www.pivotor.com

For each System, Average L1MP for Each Shift

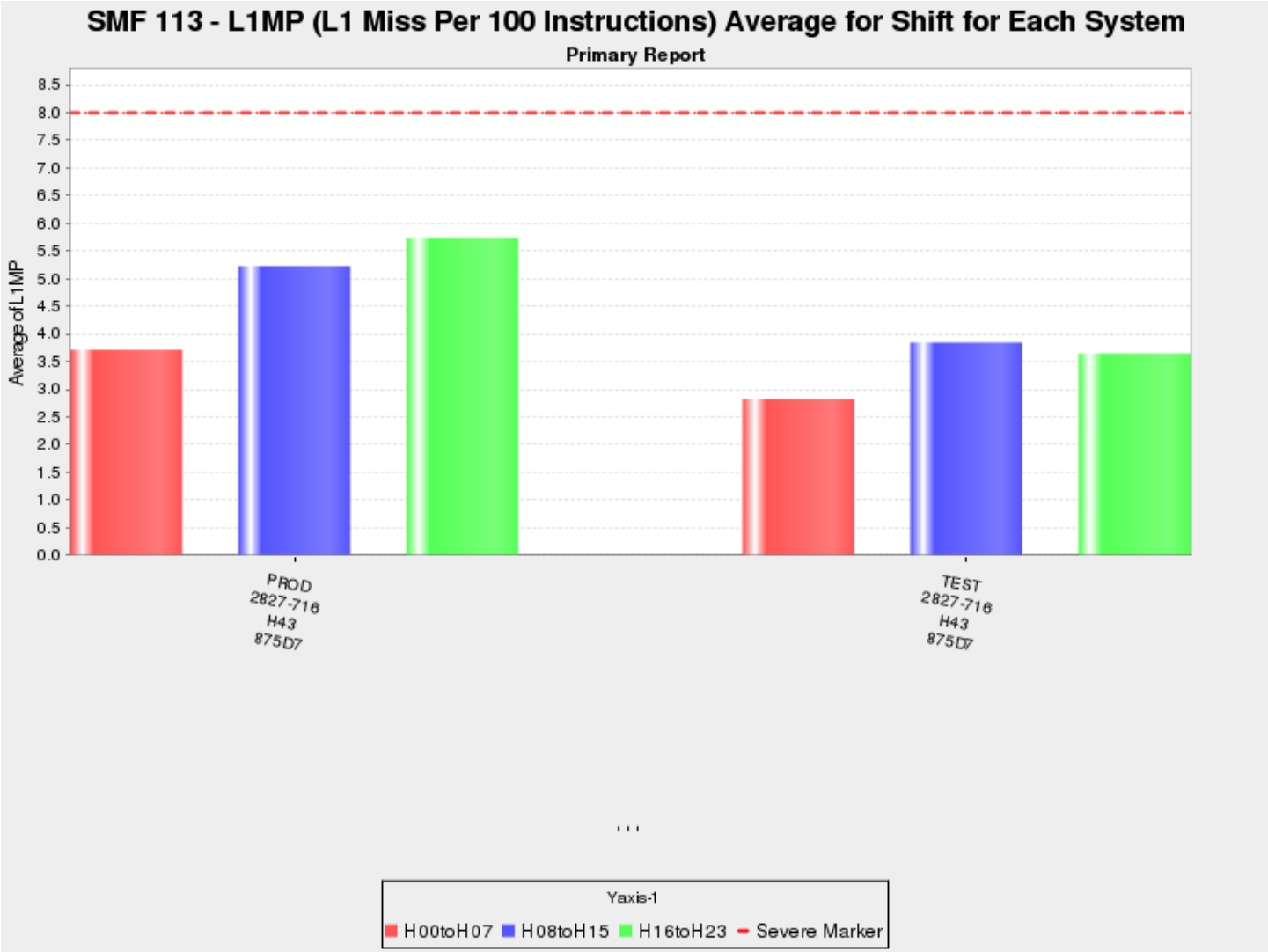


Chart created at www.pivotor.com

For each System, L1MP Over Time

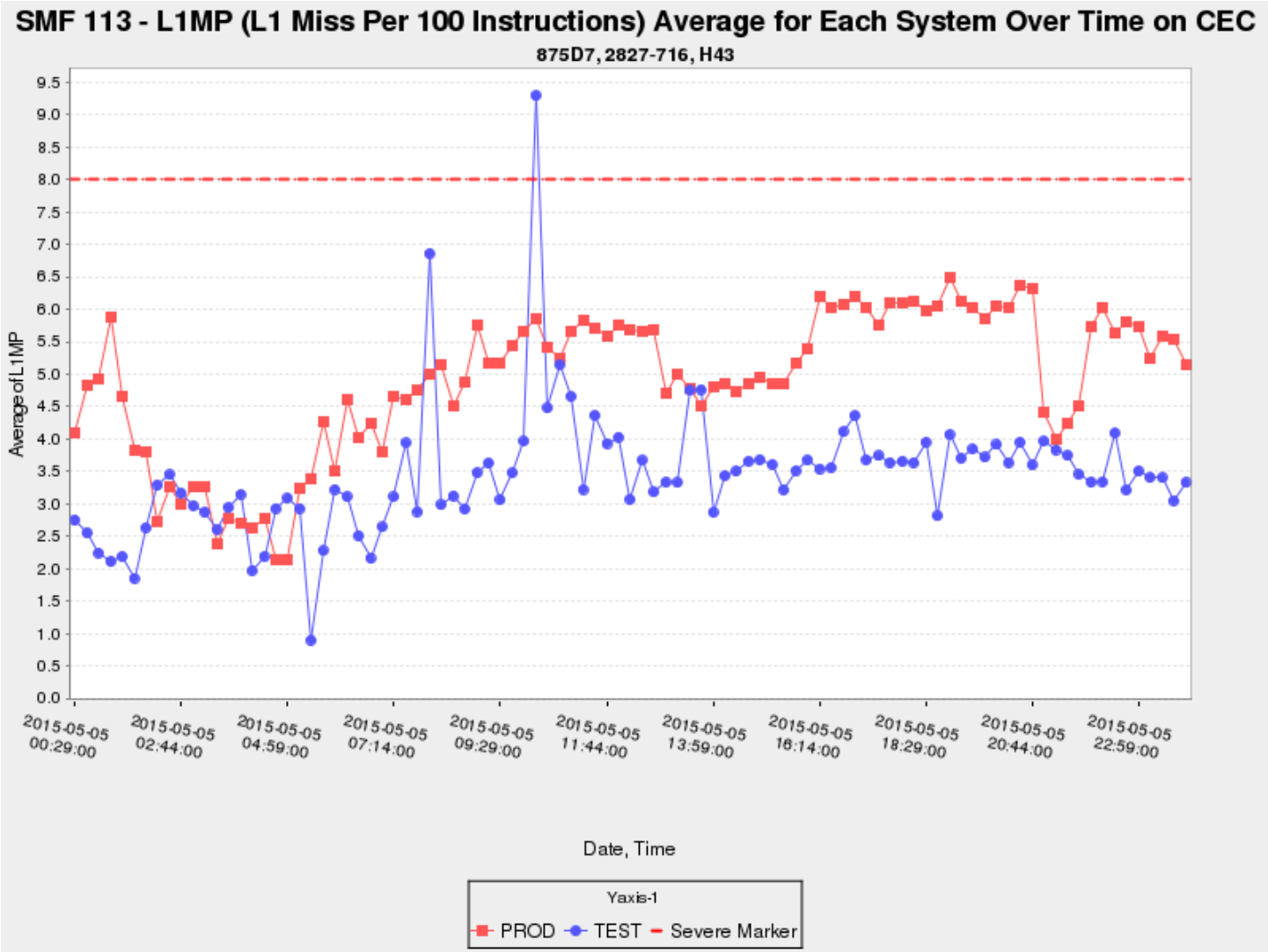


Chart created at www.pivotor.com

For System PROD, L1MP by Engine Type Over Time

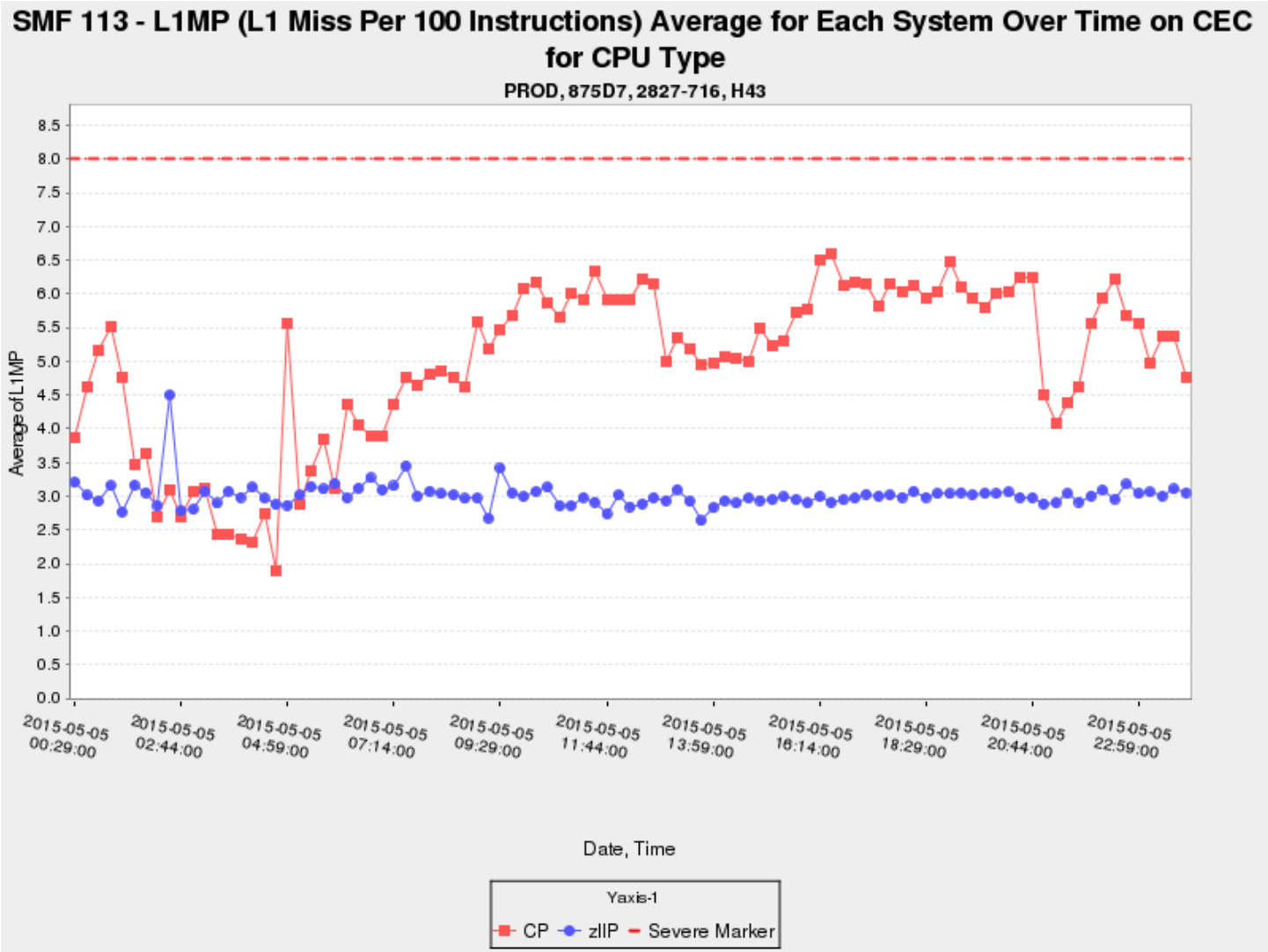


Chart created at www.pivotor.com

For System PROD, L1MP by Engine Over Time

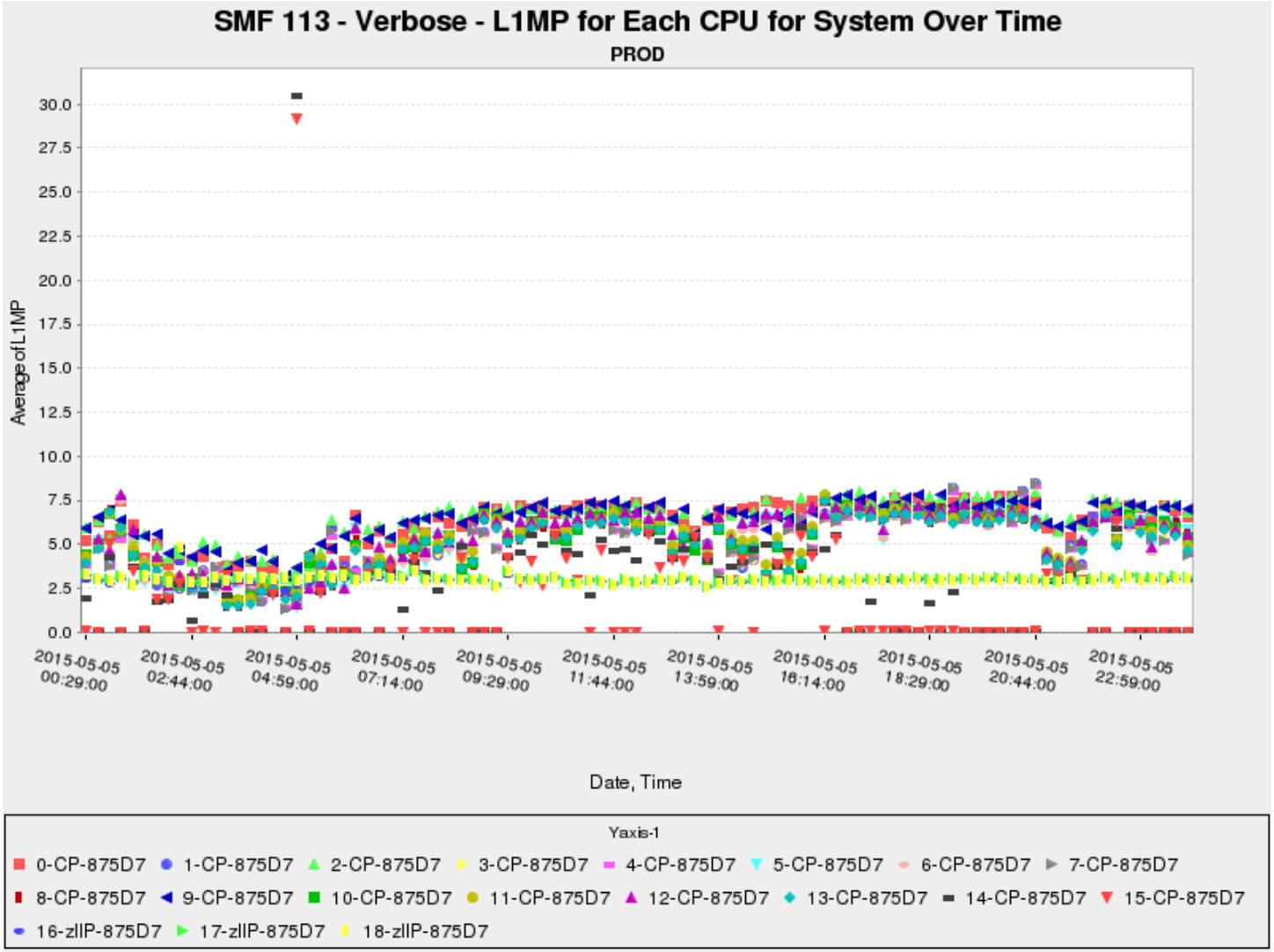
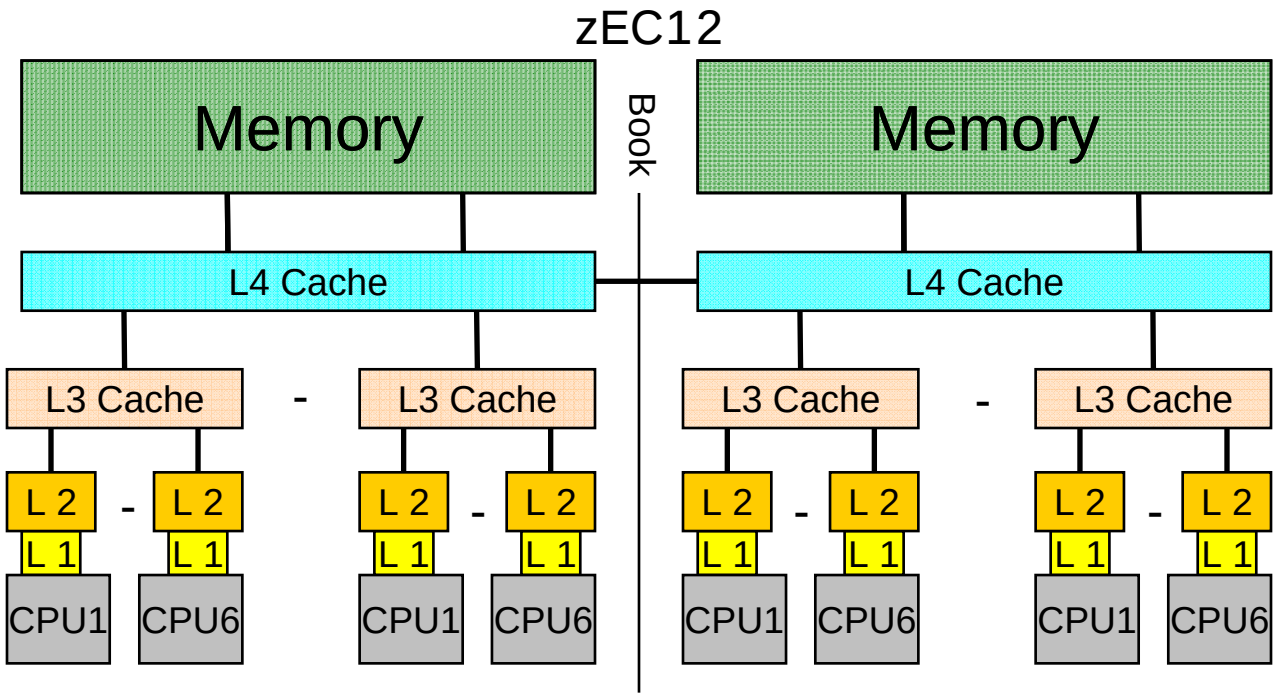


Chart created at www.pivotor.com

Components of L1 Sourced

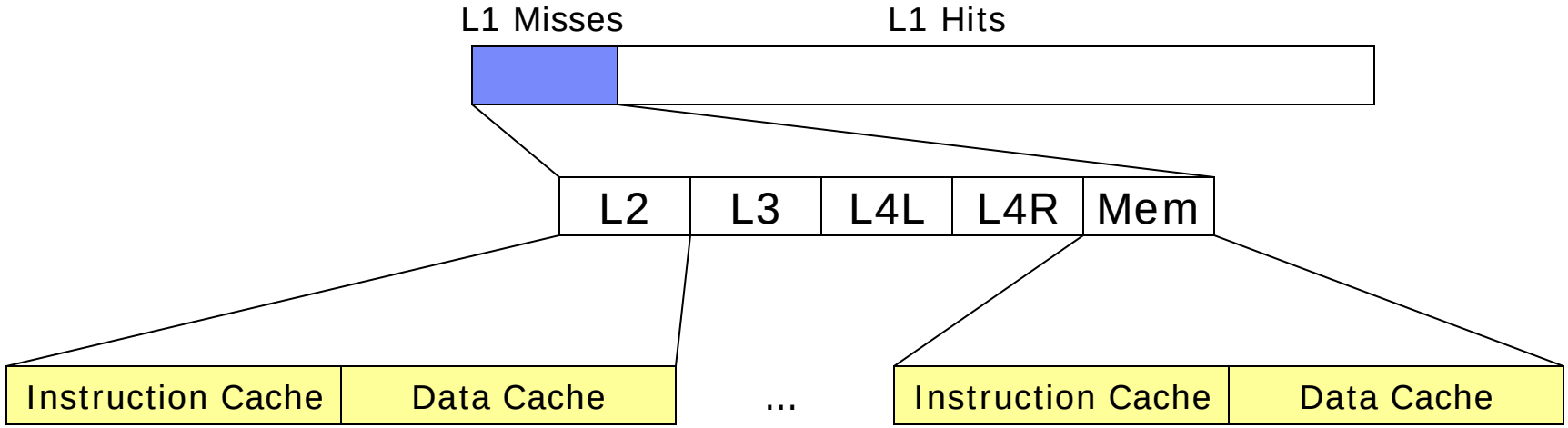
- If an L1 Miss Per 100 Occurs then the Instructions and Data needs to be Sourced from some other cache / memory location

- 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

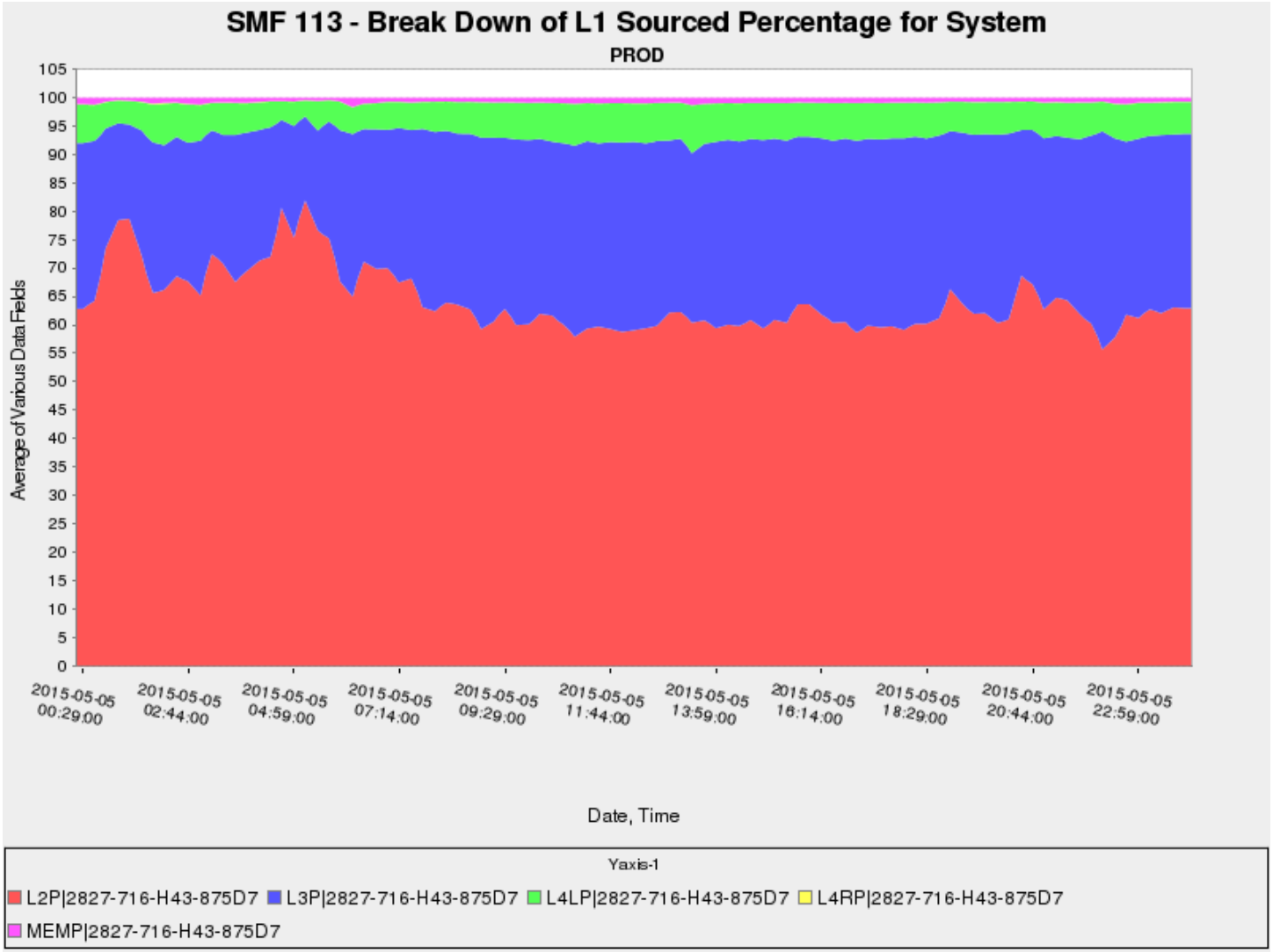


From Where is L1 Sourced?

- Answer
  - From L2 (Instruction and Data)
  - From L3 (Instruction and Data)
  - From L4 Local (Instruction and Data)
  - From L4 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 Source Percentage Breakdown for System PROD



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

A little more on
CPI – Cycles Per Instruction

Reminder: Processor capacity cycles are spent doing either

- Execution of workloads (instructions against data)

- Resolving cache misses

We want to measure the relative cost of resolving the cache misses

Cycle Usage of the Machine

- Reminder: Processor capacity cycles are spent doing either
  - Execution of workloads (instructions against data)
  - Resolving cache misses

- We want to understand the cost of the L1 cache misses
  - Unfortunately, we only know the total cycles spent to resolve the misses
  - We do not know the cost for each level of cache to resolve the miss

- Example:
  - Say we have an L1MP of 5% (5% L1 misses per 100 instructions)
  - Say 25% of those misses were resolved in the L4 Local

  - We know the number of instructions, and the number of cycles used
  - We also know the number of cycles used to resolve ALL misses
    - This is the CPU cost of the misses

  - But we do not know the cycles spent to resolve just the L4 local misses

(zEC12) Using CPI to Gain Some Insights to Cost to Resolve Misses

- Total Machine Cycles per Instruction (Actual CPI)

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

- Estimate Finite CPI (Est Finite CPI)
  - Think of this as penalty cycles per instruction if we just had the 'finite' cache
  - Cycles spent sourcing L1 = (B3+B4), thus

  Est Finite CPI = ((B3+B5) / B1) * (.54 + (0.04*RNI)) (note: scale for overlap)

- Estimated Instruction Complexity CPI (Est Instr Cmplx CPI)
  - Think of this as if there was in infinite amount of L1

  EIC_CPI = ((CPI) – (*Est Finite CPI*))

- Est SCPL1M
  - 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 constants .84 for the z10 and .59 for z196 and .54 for zEC12
    - Note a lower value for zEC12 to show improved overlapping

  Est SCPL1M = ((B3+B5) / (B2+B4)) * (.54 + (0.04*RNI))

(z196) Using CPI to Gain Some Insights to Cost to Resolve Misses

- Total Machine Cycles per Instruction (Actual CPI)

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

- Estimate Finite CPI (Est Finite CPI)
  - Think of this as penalty cycles per instruction (as if we had nothing put penalty to source)
  - Cycles spent sourcing L1 = (B3+B4), thus

  Est Finite CPI = ((B3+B5) / B1) * (.59 + (0.01*RNI)) (note: scale for overlap)

- Estimated Instruction Complexity CPI (Est Instr Cmplx CPI)
  - Think of this as the cost of the CPU out of the micro processor

  EIC_CPI = ((CPI) – (*Est Finite CPI*))

- Est SCPL1M
  - 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 constants .84 for the z10 and .59 for z196 and .54 for zEC12
    - Note a lower value for zEC12 to show improved overlapping

  Est SCPL1M = ((B3+B5) / (B2+B4)) * (.54 + (0.01*RNI))

(z10) Using CPI to Gain Some Insights to Cost to Resolve Misses

- Total Machine Cycles per Instruction (Actual CPI)

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

- Estimate Finite CPI (Est Finite CPI)
  - Think of this as penalty cycles per instruction (as if we had nothing put penalty to source)
  - Cycles spent sourcing L1 = (B3+B4), thus

  Est Finite CPI = ((B3+B5) / B1) * .84 (note: scale for overlap)

- Estimated Instruction Complexity CPI (Est Instr Cmplx CPI)
  - Think of this as CPI if there was an infinitely large L1 cache (i.e. no penalty cycles)

  EIC_CPI = ((CPI) – (*Est Finite CPI*))

- Est SCPL1M
  - 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 constants .84 for the z10 and .59 for z196 and .54 for zEC12
    - Note a lower value for zEC12 to show improved overlapping

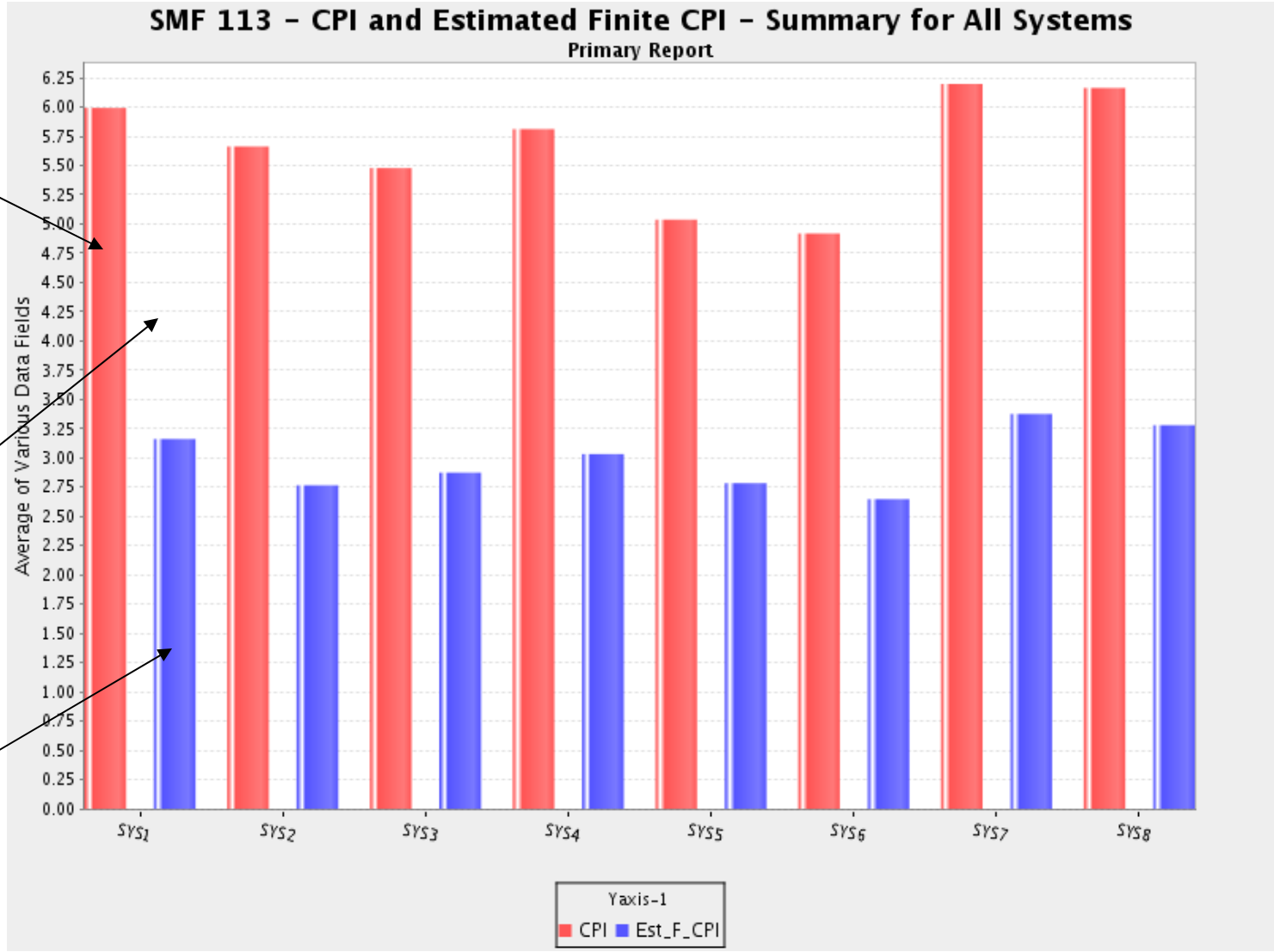
  Est SCPL1M = ((B3+B5) / (B2+B4)) * .84

CPI vs EF_CPI for each System

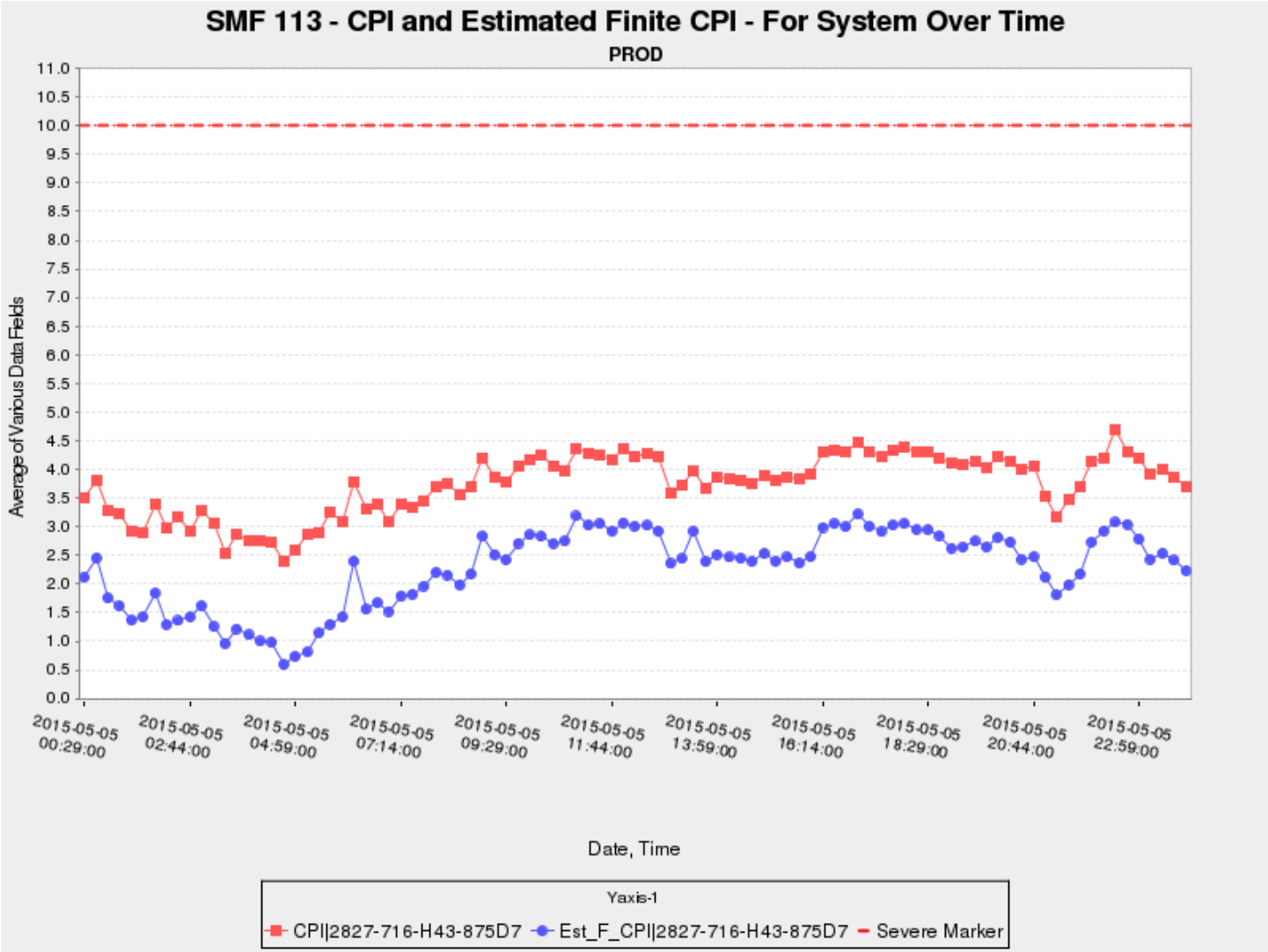
CPI

Instruction Complexity CPI

Estimated Finite CPI



CPI vs EF_CPI For System PROD Over Time



Relative Nest Intensity
and
Understanding Usage of the Memory Hierarchy

Reminder: Processor capacity cycles are spent doing either

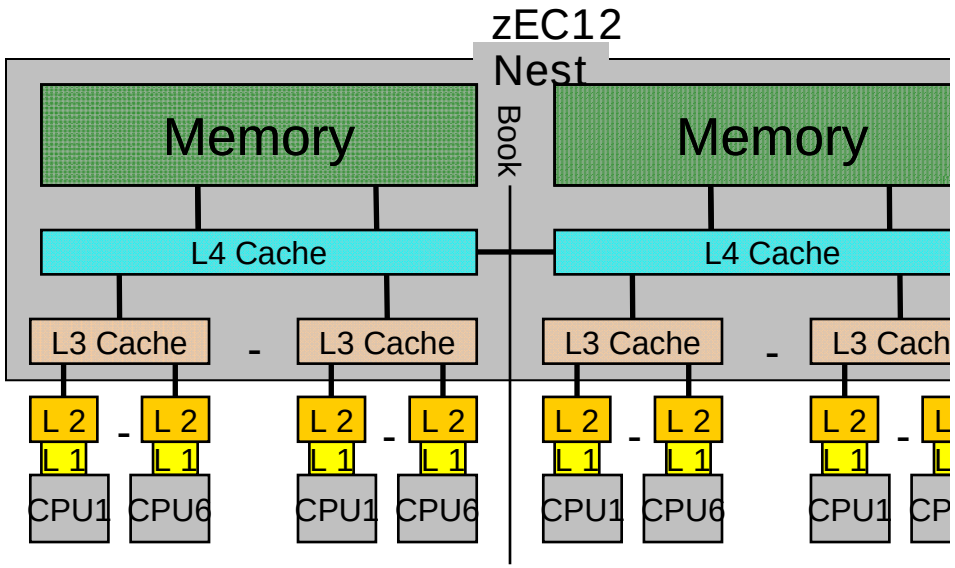
- Execution of workloads (instructions against data)

- Resolving cache misses

We want to measure the relative cost of resolving the cache misses

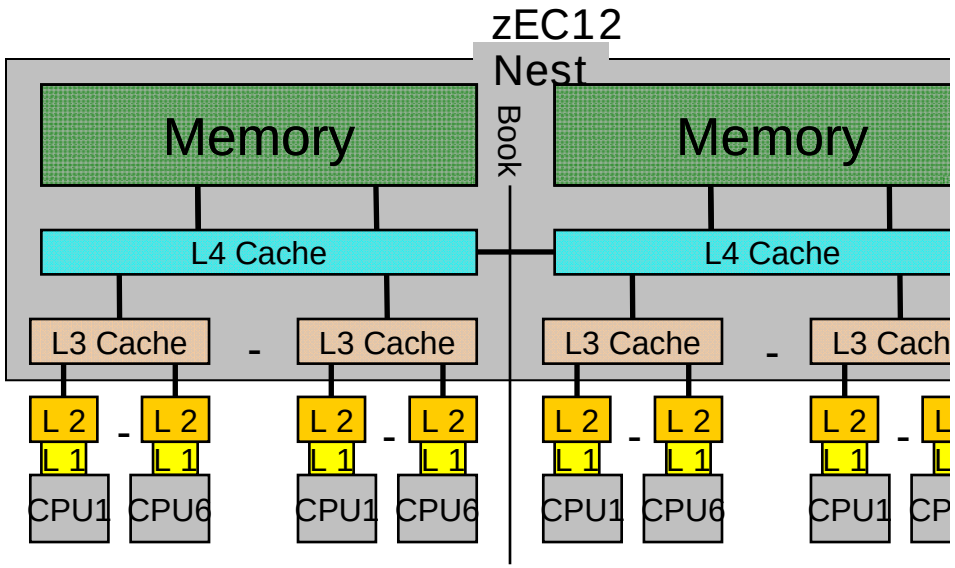
Usage of the Shared areas of the processor cache hierarchy (Nest) influences machine capacity

- How these shared areas of memory are used, and the cost of retrieving data and instruction from these areas results in a machine variability dependent on a wide variety of factors:
  - Machine configuration that influence usage of the processor design and the nest
  - PR/SM configuration, hypervisor usage, machine virtualization
  - z/OS operating system setup, parameters, WLM, and virtualization of the z/OS system resources
  - z/OS workload characteristics, workload types, workload dispatching, workload profiles, usage of resources (CPU, storage, I/O) by the workloads



Nest View of 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
- Usage of the shared area caches varies more based on
  - Machine configuration
  - PR/SM configuration
  - z/OS configuration and parameters
  - Workloads

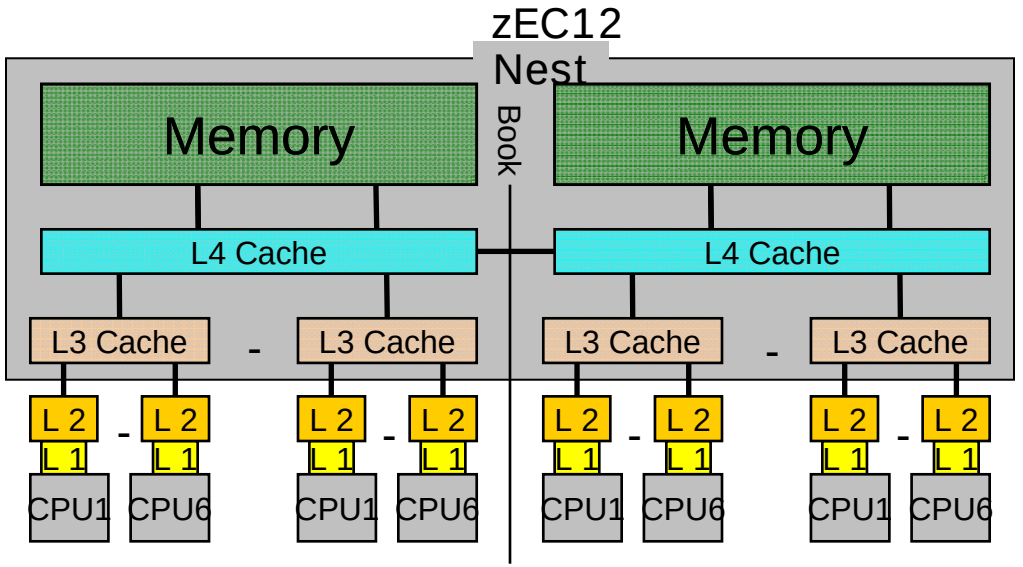


Evaluating Processor / Nest / Workload Relationship

- There is a desire to understand the variability of processor capacity relative to the workload ‘usage’ of the Nest

- The SMF 113 provides insight into the number of times L1 was sourced from the Nest

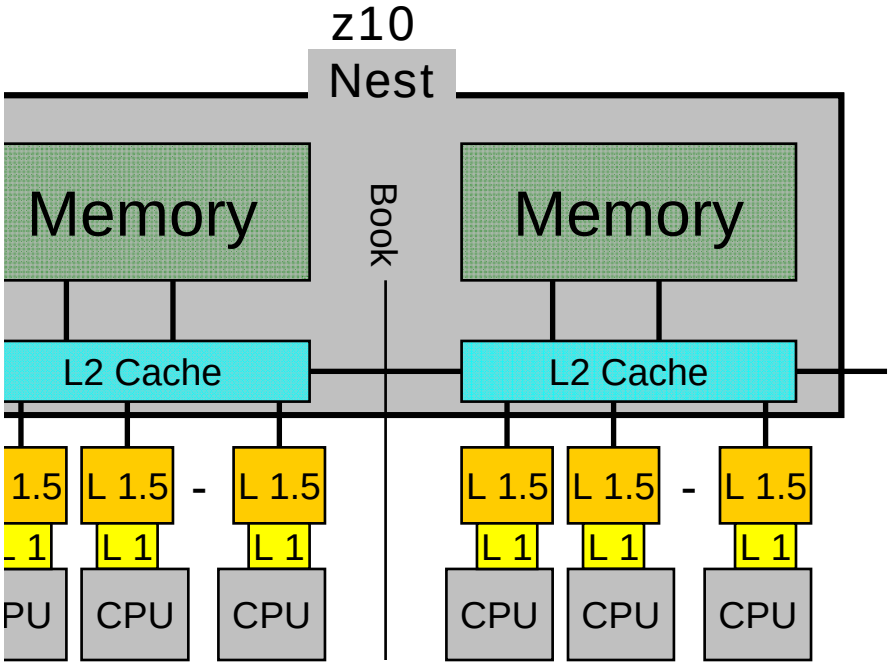
- Less Than Good News:
  The SMF 113 does not provide the penalty cycles for the individual levels of cache
  - Only total penalty cycles for all L1 sourcing (for I and D cache)

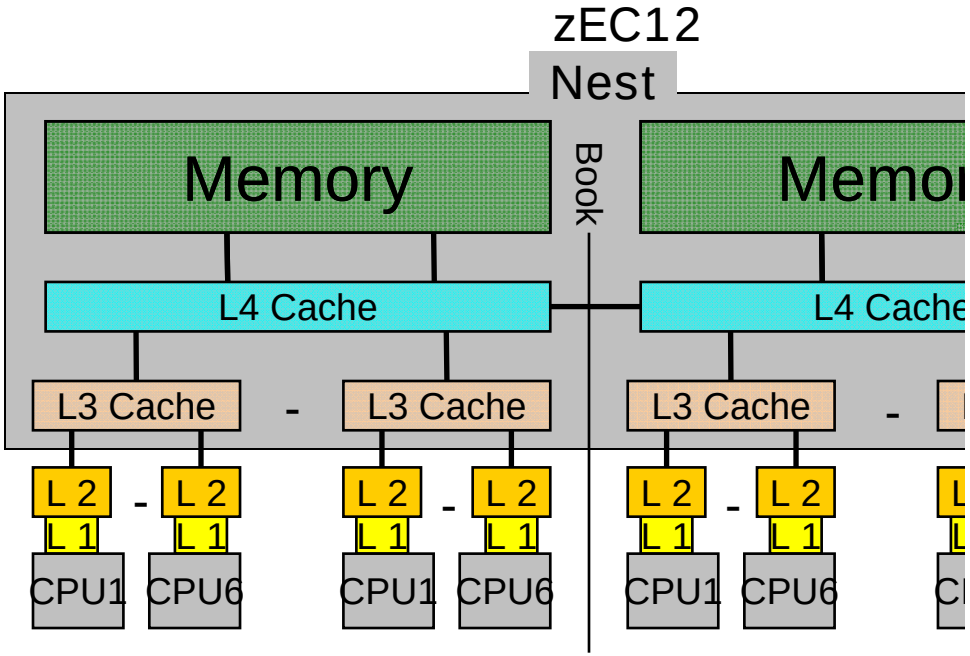


Relative Nest Intensity

- Think of RNI as a ‘stress factor’ on the memory hierarchy
  - New LSPR workloads will be based on RNI
  - It should be noted that the RNI is calculated such that if only the machine changes, the RNI profile for a customer would not (i.e. attempts to be ‘relative’)

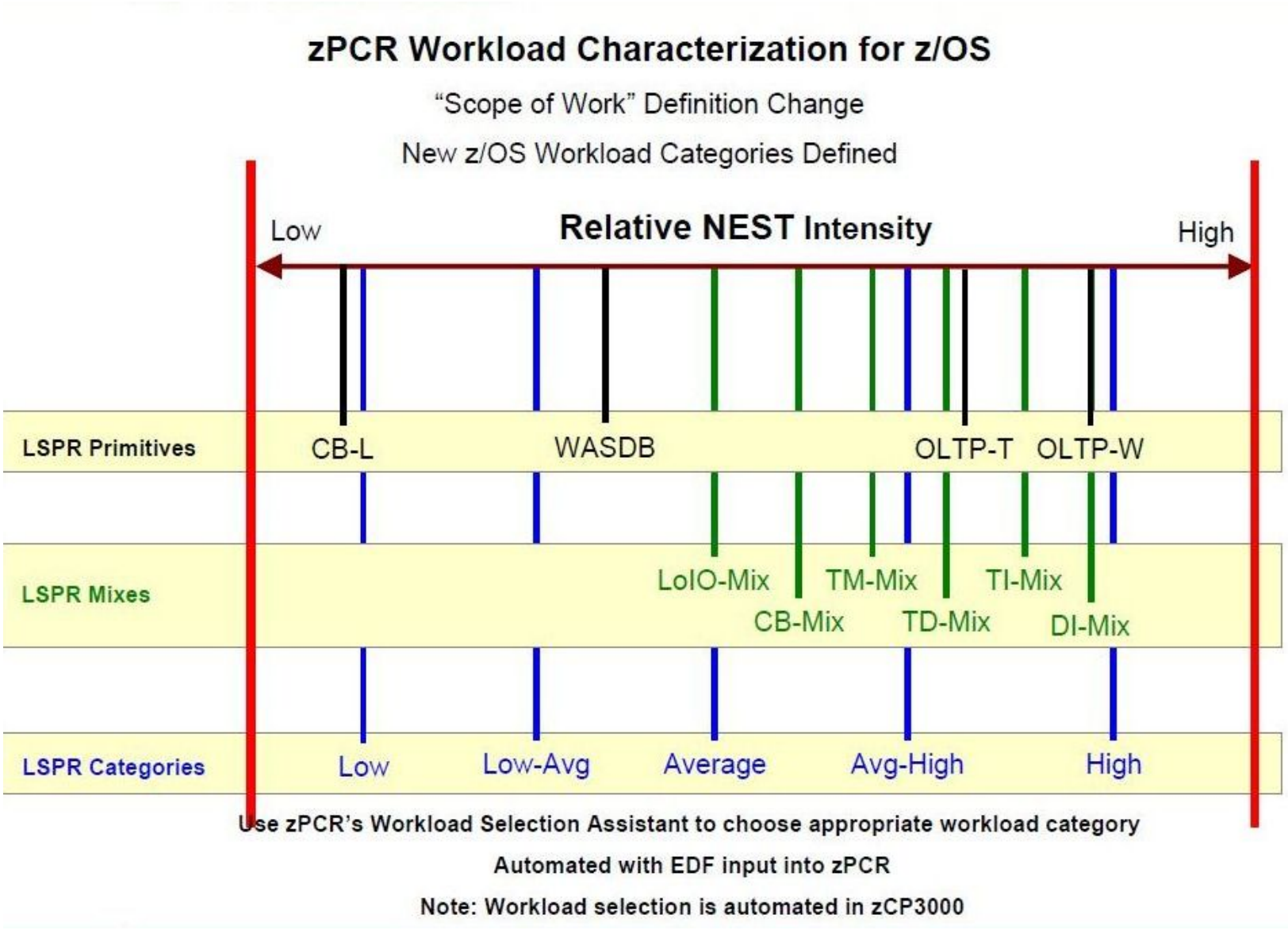
L1MP	RNI	Workload Hint
<3%	>= 0.75 < 0.75	AVERAGE LOW
3% to 6%	>1.0 0.6 to 1.0 < 0.6	HIGH AVERAGE LOW
>6%	>=0.75 < 0.75	HIGH AVERAGE





Relative Nest Intensity and IBM LSPRs

(*Chart Source: John Burg, IBM ATS, SHARE 2013, session 13098 - *CPU MF 2013 Update and WSC Experiences Now More than Ever*)



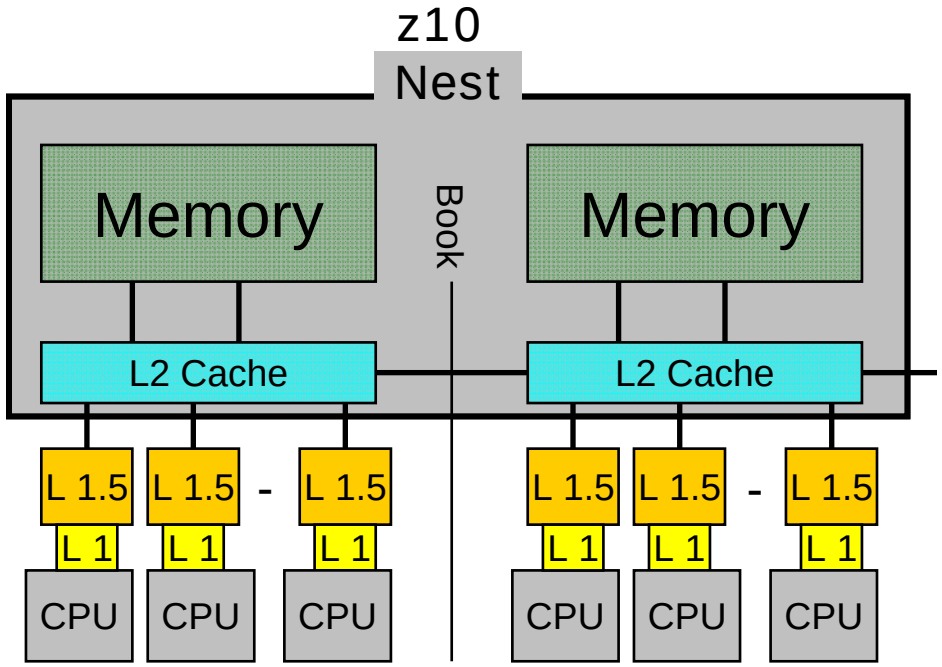
Relative Nest Intensity (for z10)

- *"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)

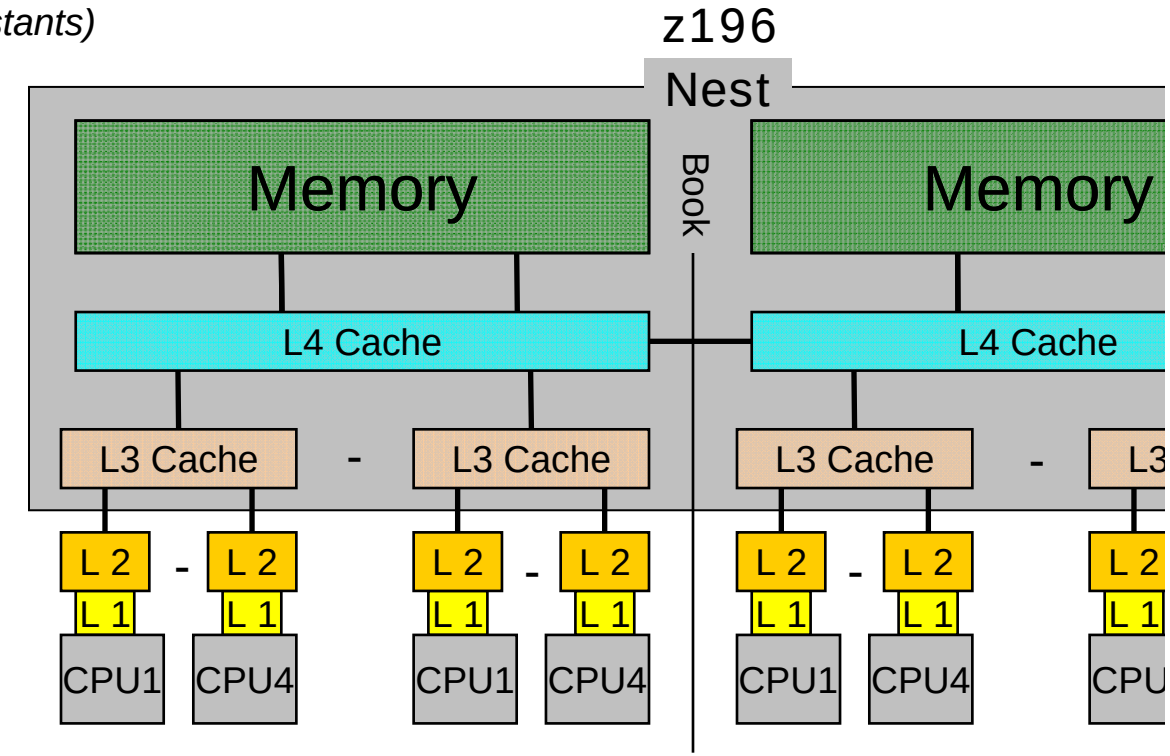
- *"Relative Nest Intensity reflects the distribution and latency of sourcing from shared caches and memory"* (J Burg, IBM)

  *RNI = 1.67 * ((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



Relative Nest Intensity (for zEC12)

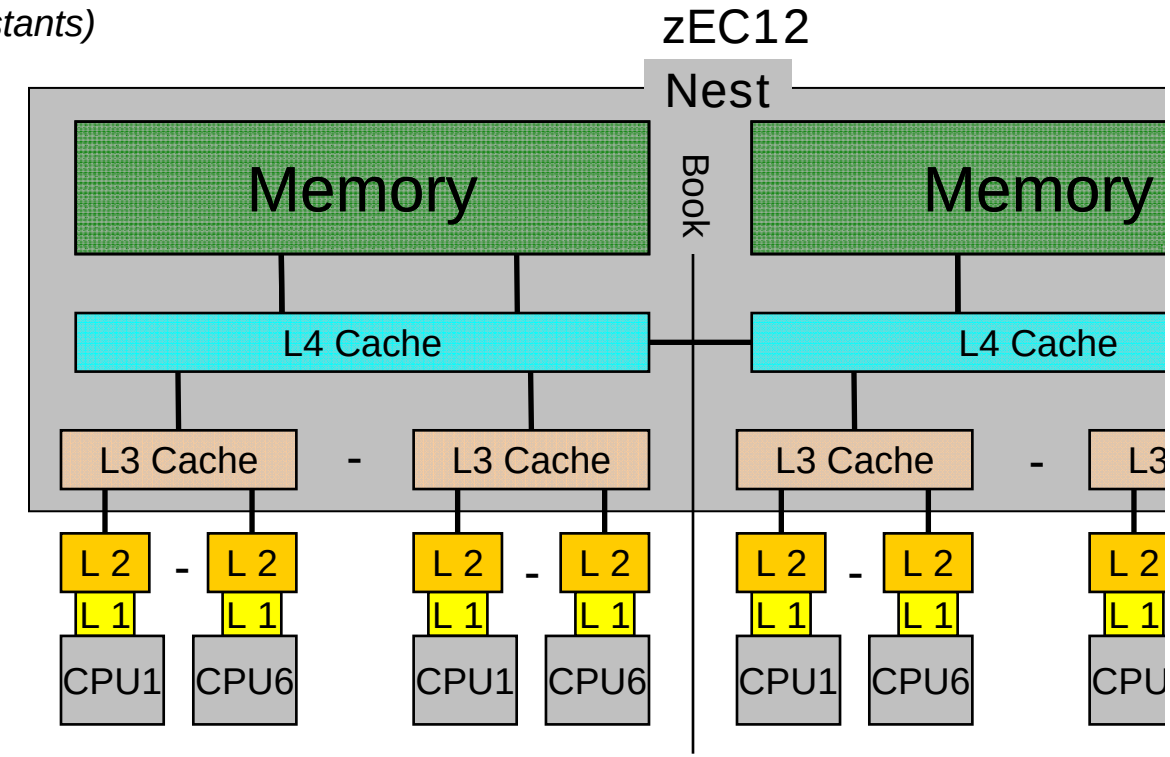
- *"Relative Nest Intensity reflects the distribution and latency of sourcing from shared caches and memory"* (J Burg, IBM)

  *RNI = 2.3 * ((0.4 * L3P) +*
  *(1.2 * L4LP) +*
  *(2.7 * L4RP) +*
  *(8.2 * 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



Relative Nest Intensity (for z13)

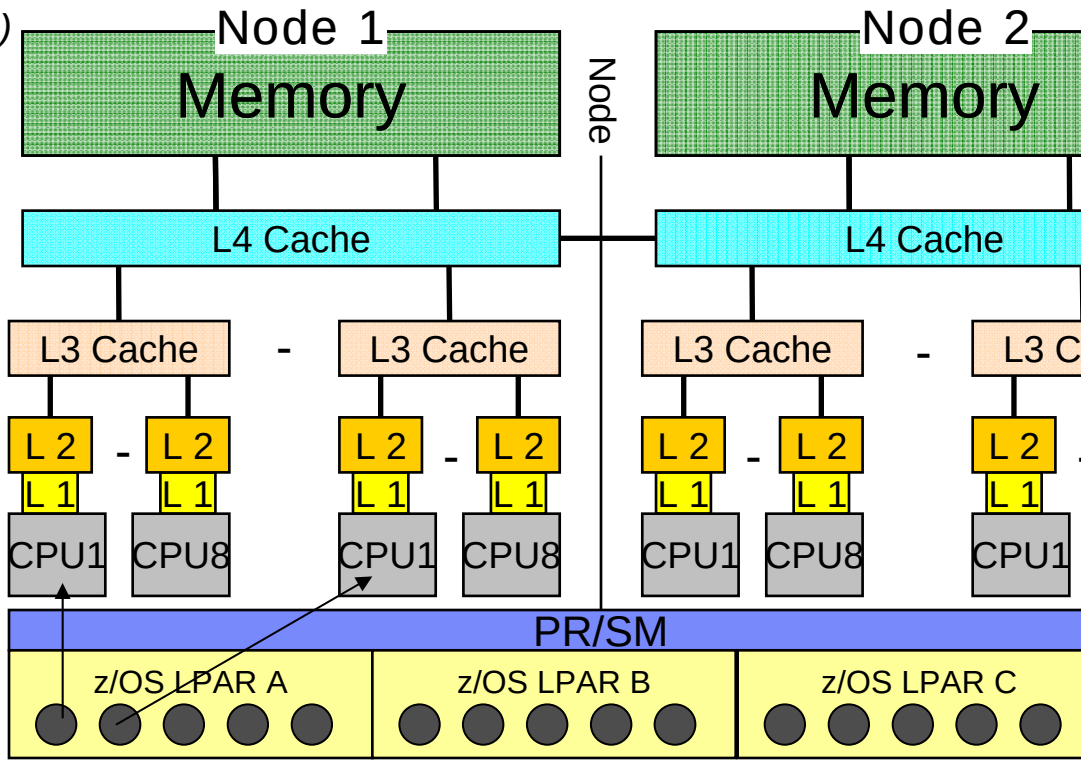
- *"Relative Nest Intensity reflects the distribution and latency of sourcing from shared caches and memory"* (J Burg, IBM)

  *RNI = 2.6 * ((0.4 * L3P) +*
  *(1.6 * L4LP) +*
  *(3.5 * 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



For each System, Average RNI Over 24 Hours

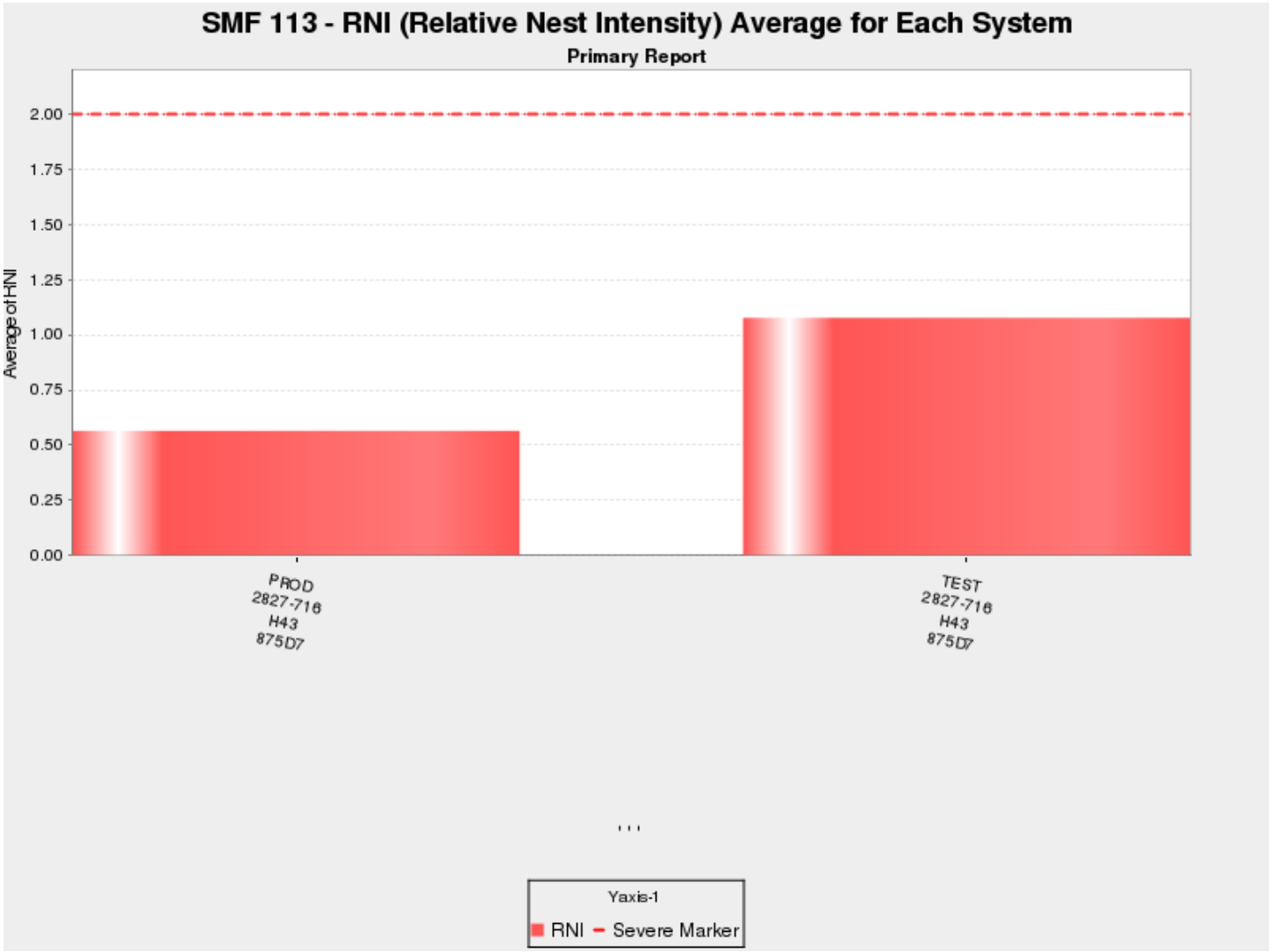
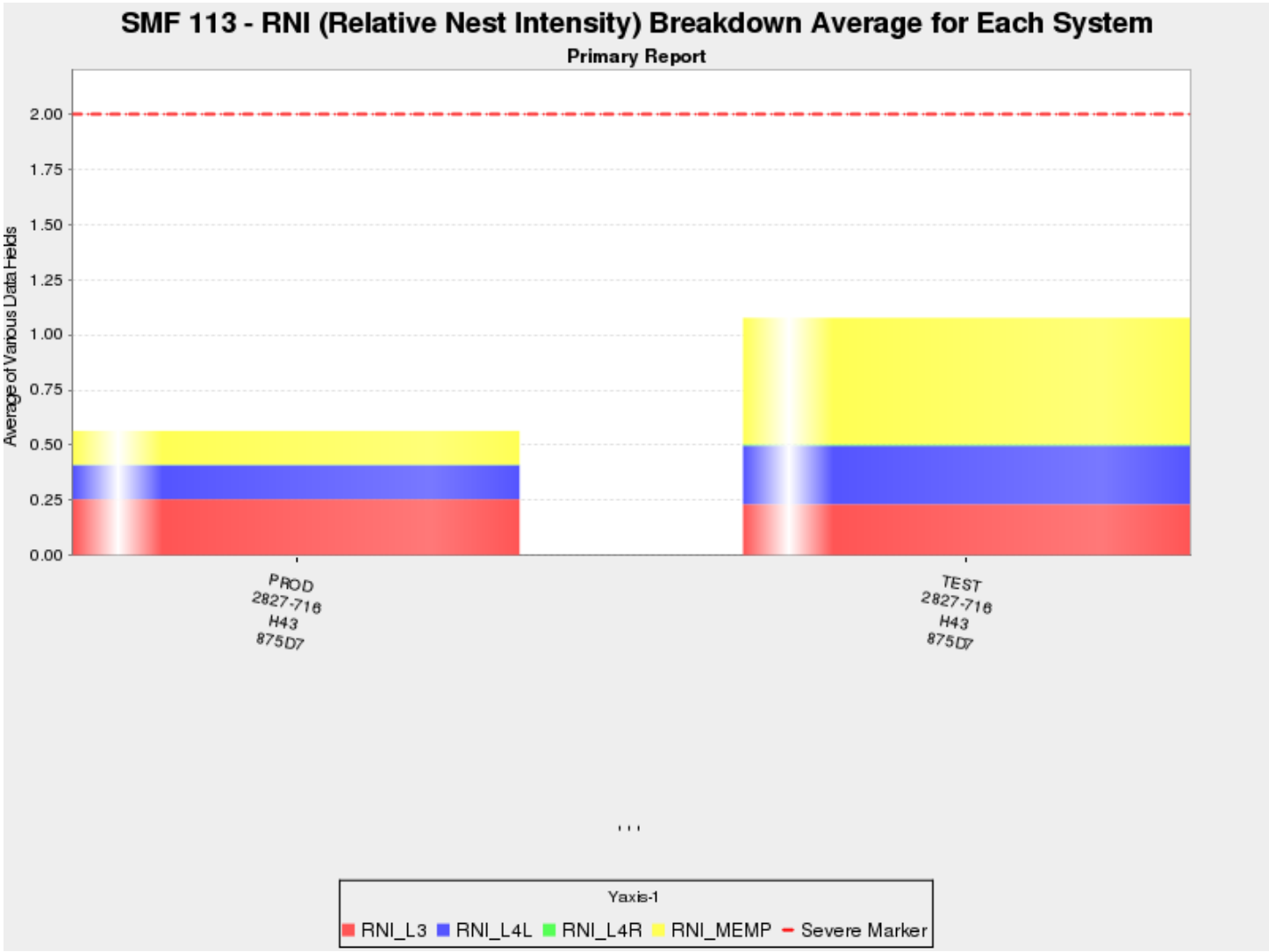
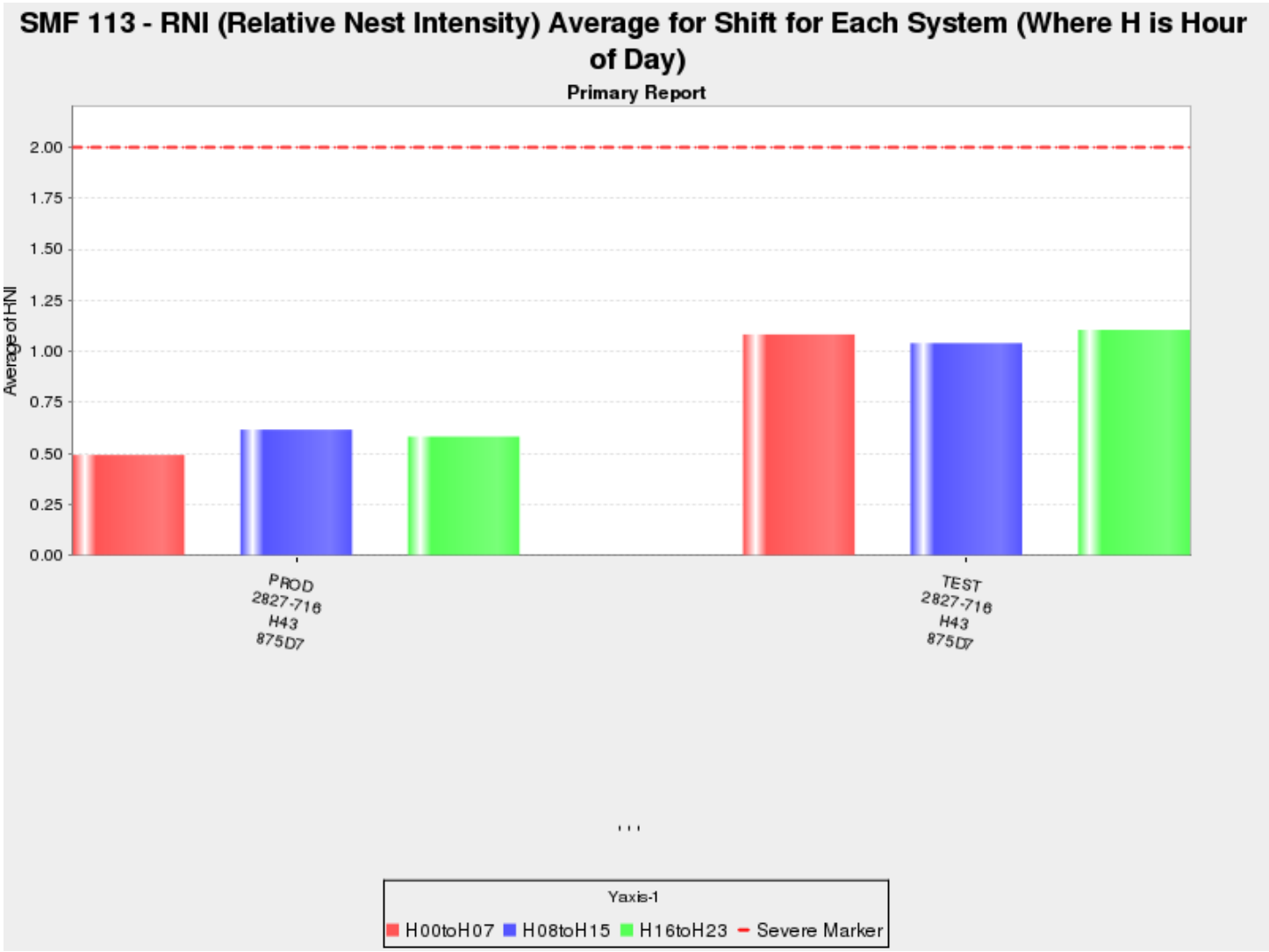


Chart created at www.pivotor.com

For each System, Average RNI Over 24 Hours Broken Down by Cache Area



For each System, Average RNI for Each Shift



For each System, RNI Over Time

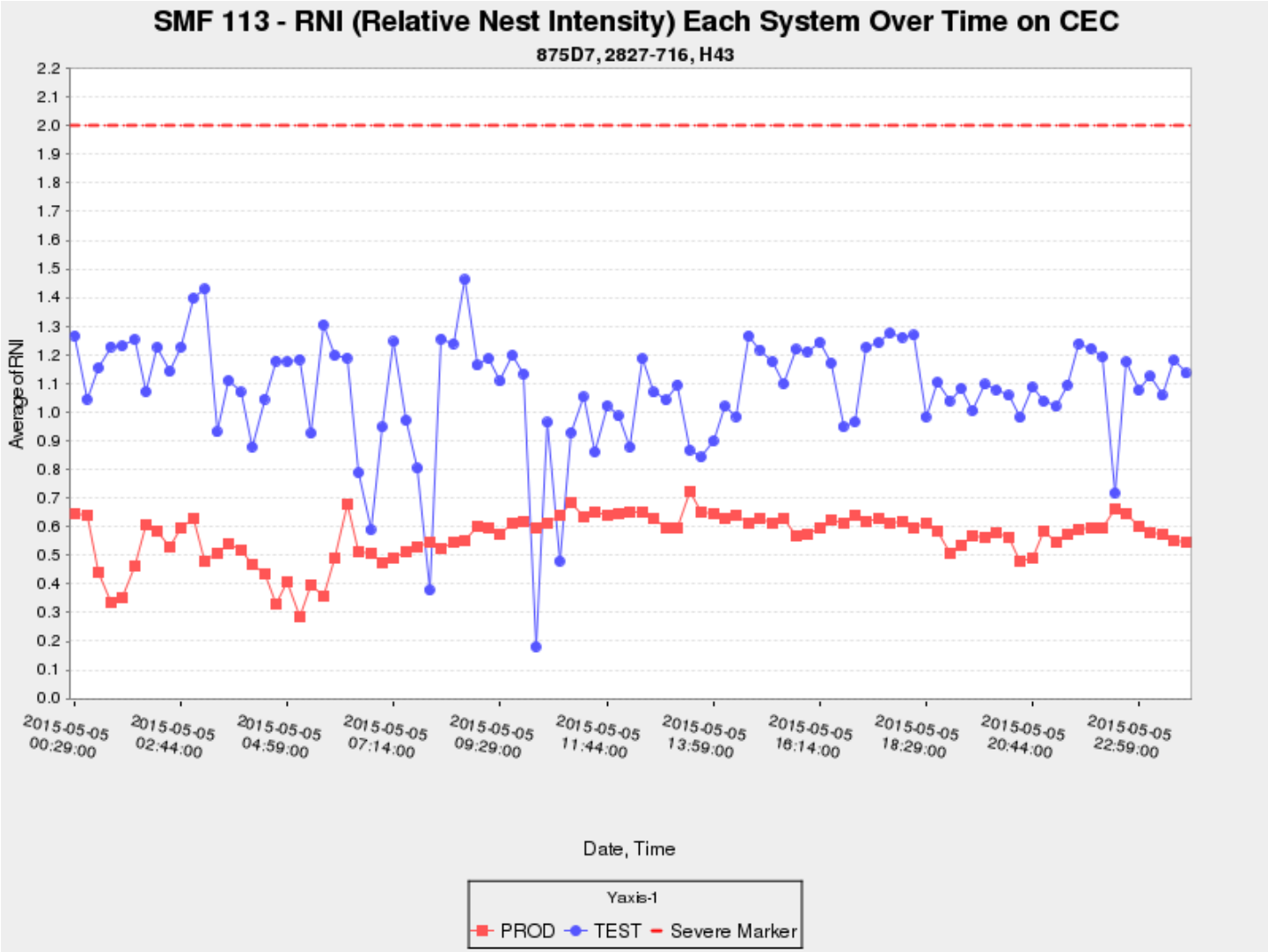


Chart created at www.pivotor.com

For System PROD, RNI Over Time Broken Down by Cache Area

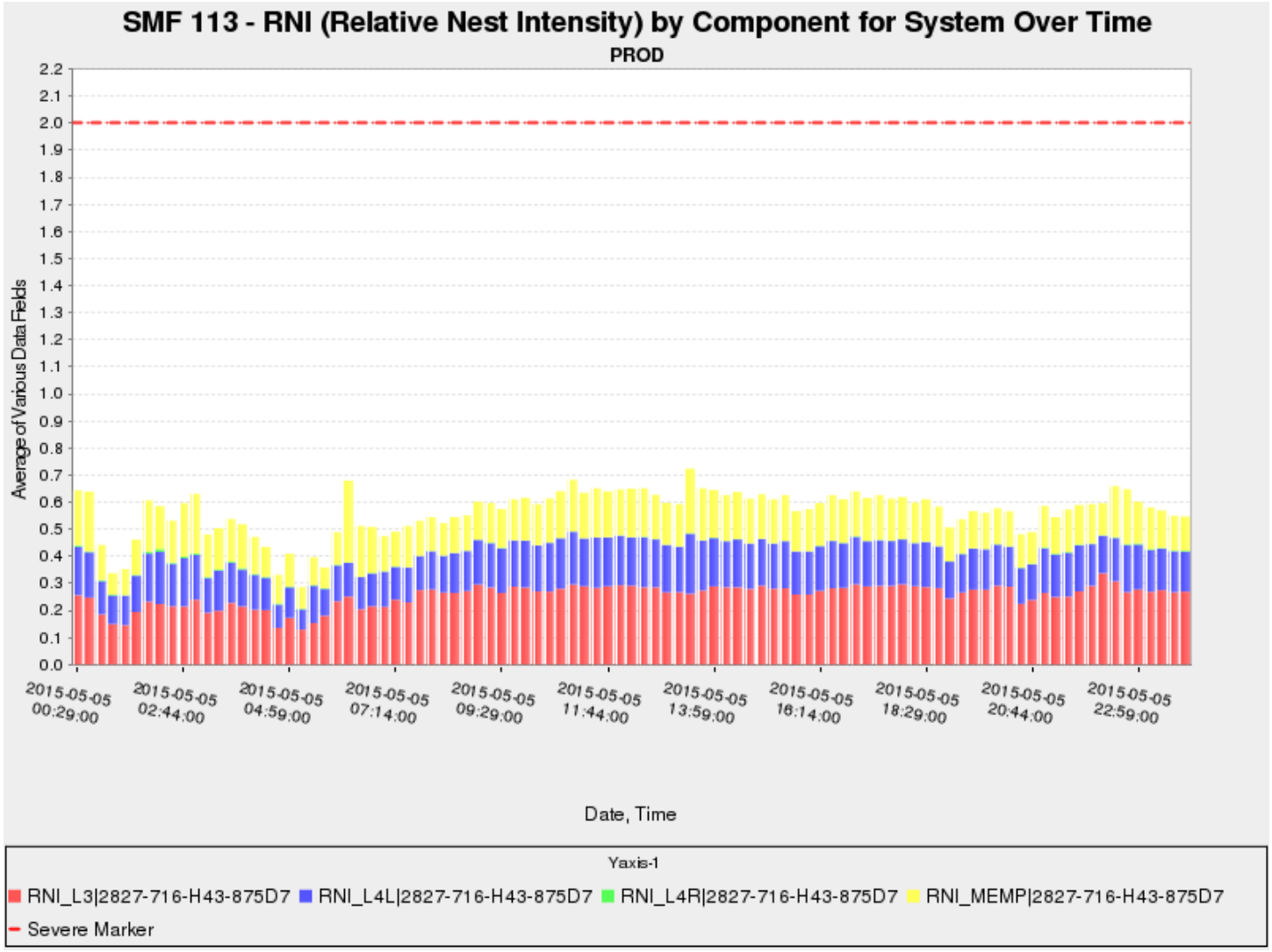
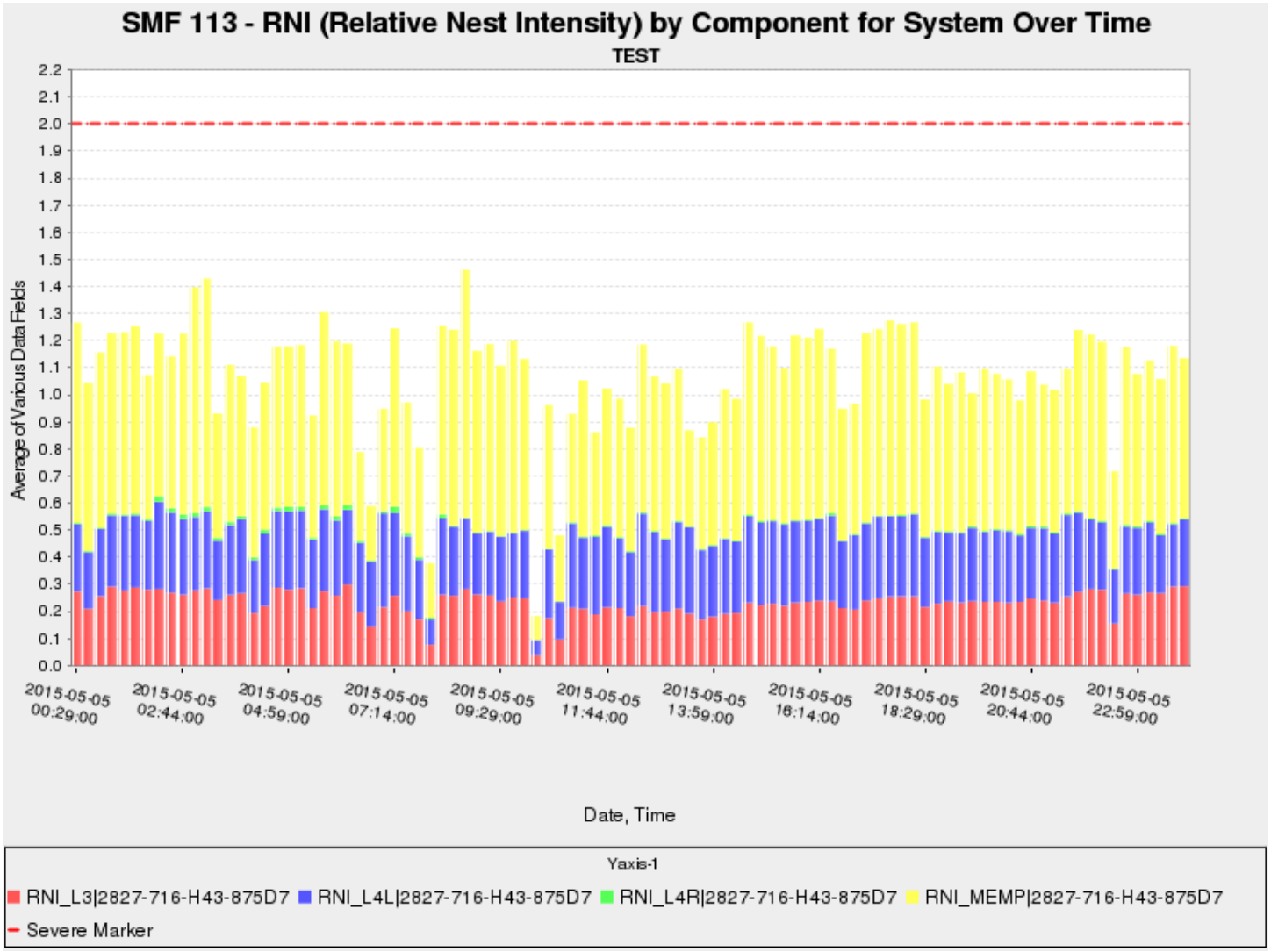
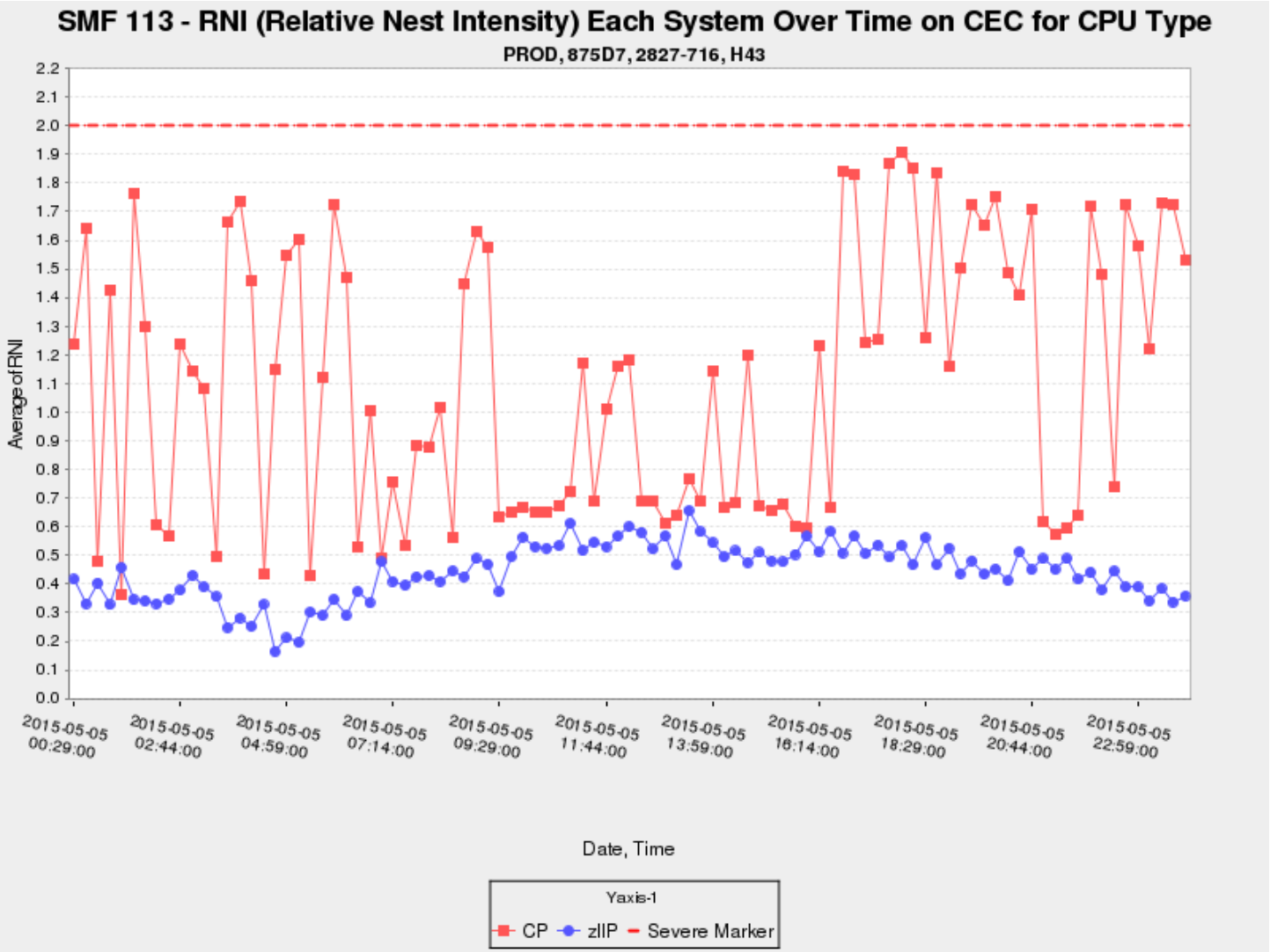


Chart created at www.pivotor.com

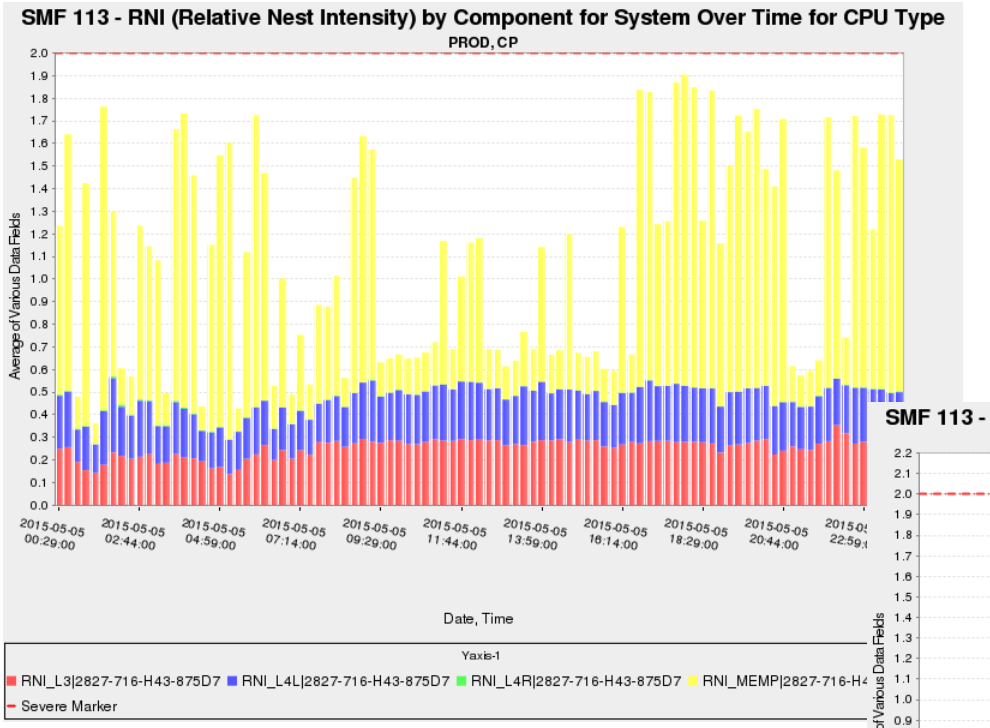
For System TEST, RNI Over Time Broken Down by Cache Area

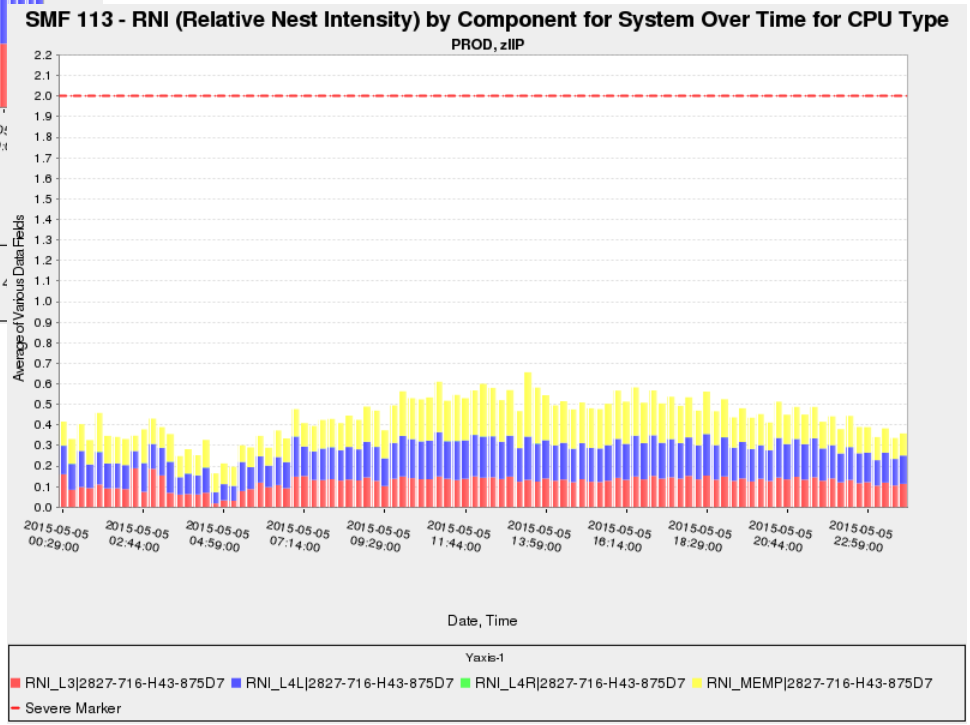


For System PROD, RNI by Engine Type Over Time



For System PROD, RNI by Engine Type Over Time





CPI versus RNI

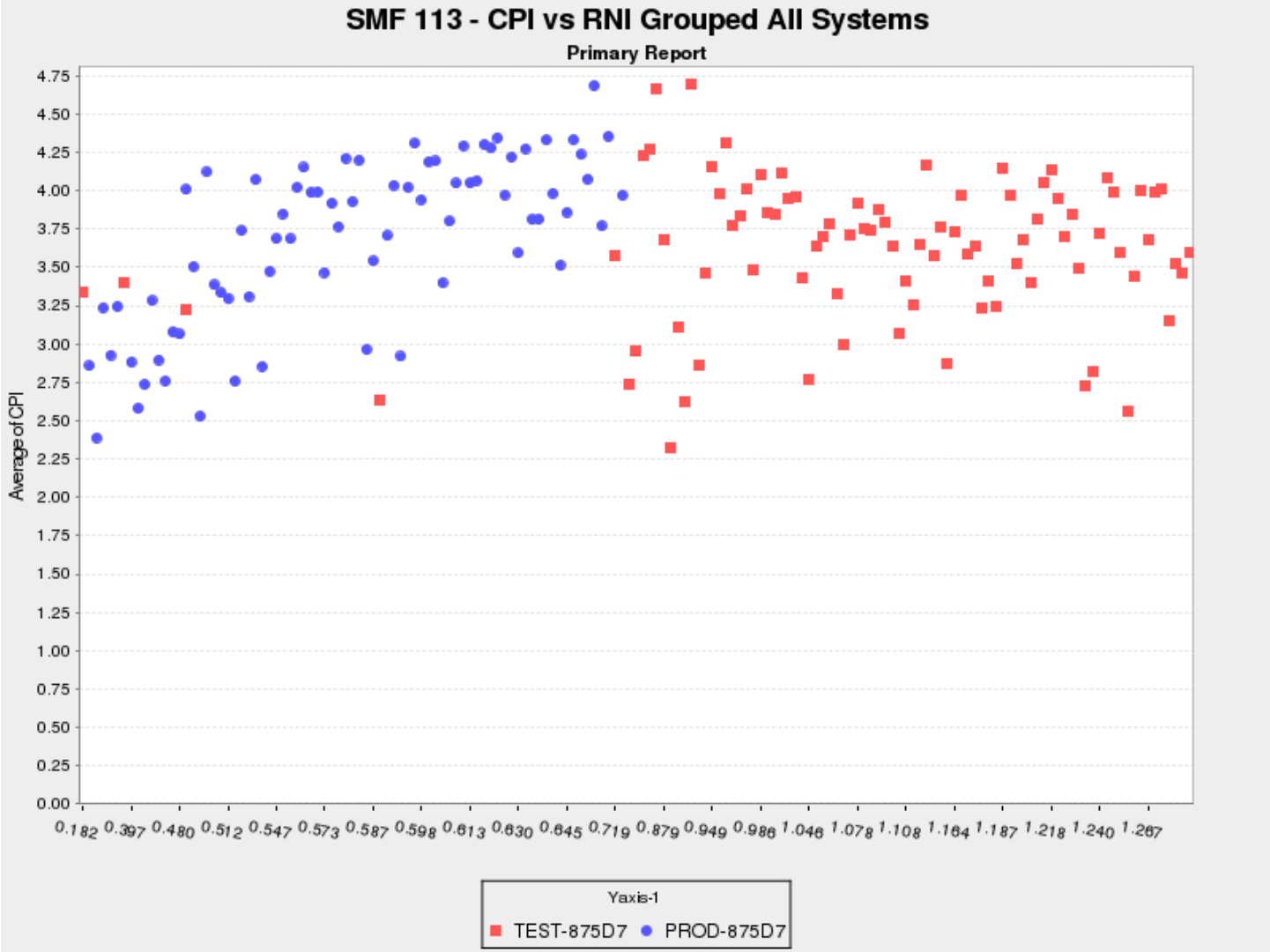


Chart created at www.pivotor.com

z10 to z196 Conversion
- CPI versus RNI (Different Sysplex Example)

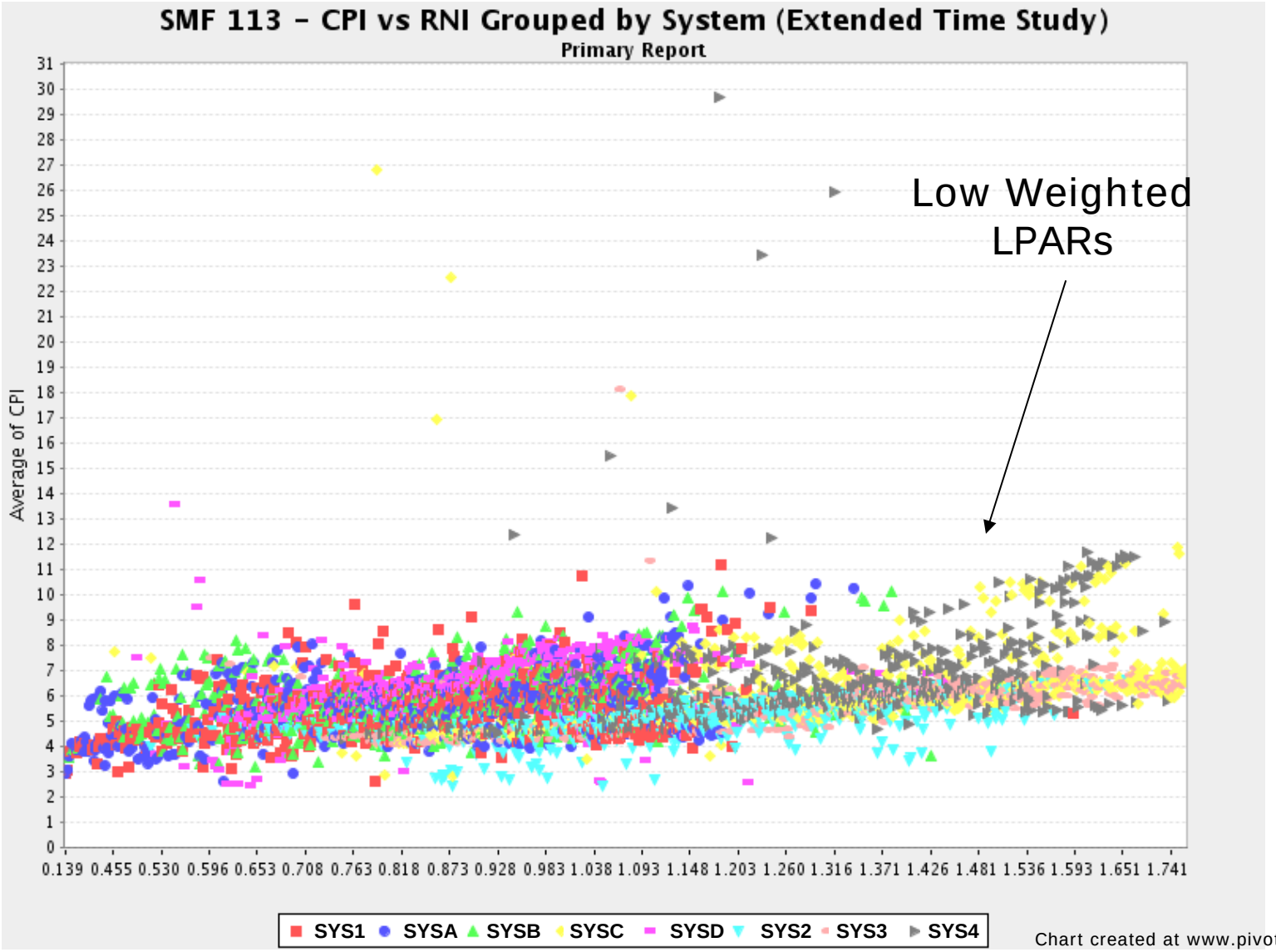


Chart created at www.pivotor.com

Key Influences of Processor Performance and Capacity

- What the IBM LSPRs attempt to do is help customers estimate processor sizing
- But IBM's workloads are in a control environment based on synthetic workloads
- Question: What are the key influences that result in variations of a particular processor's delivered capacity relative to a customer's environment and workload?
  - That is: Why your results may vary from results based off IBM's synthetic workloads?
- These key influences are:
  - Instruction complexity of one processor family to another
    - Based hard hardware architecture
  - Path length of the code executed by customer applications and transactions
    - This relates to code executed by applications / jobs / transactions / etc.
  - Usage of the Memory Hierarchy

Key Influence – Memory Hierarchy

- Usage of the Memory Hierarchy
  - Heavily influenced by key factors result potentially wide variations in realized capacity
  - From one processor family to another there are many design alternatives
    - Levels of cache, scope of cache, latency, etc.)
  - Configuration will influence usage of the memory hierarchy
    - LPAR configuration, competition between LPARs, options such as HiperDispatch, etc.
  - Exploitation by workloads will influence usage of the memory hierarchy
    - Transaction intensity, memory intensity, I/O intensity, application mixtures, competition of resource by applications, etc.
  - z/OS performance management and options
    - WLM management of resources, affinity nodes, IEAOPTxx opts, heap sizes, initiators, etc.
- Final result is that usage of memory hierarchy heavily influences a processor's delivered capacity and performance.
- So for processor sizing, LSPRs have started focusing on this

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 LSPR SMF 113 Based Workload Categories

- Traditional IBM workload categories are still being used to determine some base capacity values
  - CB-L (Long Batch), CB-J (Java Batch), WAS-DB, OLTP-W, OLTP-T, LoIO-mix, TI-mix
- LSPR changes with the z196 processors include 3 new LSPR categories based on the SMF 113 processor cache counter measurements
  - **LOW (RNI)**
    - Light usage of the memory hierarchy
  - **AVERAGE (RNI)**
    - Average usage by most customers of the memory hierarchy
    - Similar to the old LoIO mixed workload curve
  - **HIGH (RNI)**
    - Heavy usage of the memory hierarchy
    - Similar to old DI-mix workload curve
    - These replace the previous workload mixtures
    - Allows the LSPR capacity curves to be based on the underlying hardware sensitivities
- Allows customers to map their Nest Intensity to LSPRs for processor sizings

LSPR Table Example – Post SMF 113

zEnterprise 196
(System z9 2094-701 = 1.00)

Processor	#CP	PCI**	MSU***	Low*	Average*	High*
2817-701	1	1202	150	2.14	2.15	2.06
2817-702	2	2272	281	4.15	4.06	3.78
2817-703	3	3311	408	6.13	5.92	5.46
2817-704	4	4320	531	8.06	7.72	7.08
2817-705	5	5300	650	9.96	9.47	8.66
2817-706	6	6251	766	11.82	11.17	10.19
2817-707	7	7175	879	13.65	12.82	11.68
2817-708	8	8072	988	15.44	14.42	13.12
2817-709	9	8943	1091	17.19	15.97	14.52
2817-710	10	9788	1191	18.92	17.49	15.88
2817-711	11	10609	1286	20.61	18.95	17.21
2817-712	12	11407	1381	22.27	20.38	18.49
2817-713	13	12181	1473	23.89	21.76	19.74
2817-714	14	12932	1562	25.49	23.1	20.95
2817-715	15	13662	1648	27.06	24.41	22.12
2817-716	16	14371	1731	28.59	25.67	23.26

Reports / SMF 113 Processing/Discussion 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

Presentation Summary

- New SMF 30 Instruction Count fields
- 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!

Performance Workshops Available

During these workshops you will be analyzing your own data!

- WLM Performance and Re-evaluating of Goals
  - Instructor: Peter Enrico and Scott Chapman
  - September 28 – October 2, 2015 – Columbus, Ohio, USA

- Parallel Sysplex and z/OS Performance Tuning
  (Web / Internet Based!)
  - Instructor: Peter Enrico and Scott Chapman
  - November 17 – 19, 2015

- Essential z/OS Performance Tuning Workshop
  - Instructors: Peter Enrico, Scott Chapman, Tom Beretvas
  - October 19 - 23, 2015 – Dallas, Texas, USA

- z/OS Capacity Planning and Performance Analysis
  - Instructor: Ray Wicks