#17321

Memory Management in the TB Age

Scott Chapman
Enterprise Performance Strategies, Inc.
scott.chapman@epstrategies.com

#SHAREorg

Contact, Copyright, and Trademark Notices

Questions?

Send email to Scott at scott.chapman@EPStrategies.com, or visit our website at http://www.epstrategies.com or http://www.pivotor.com.

Copyright Notice:

Trademarks:

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?

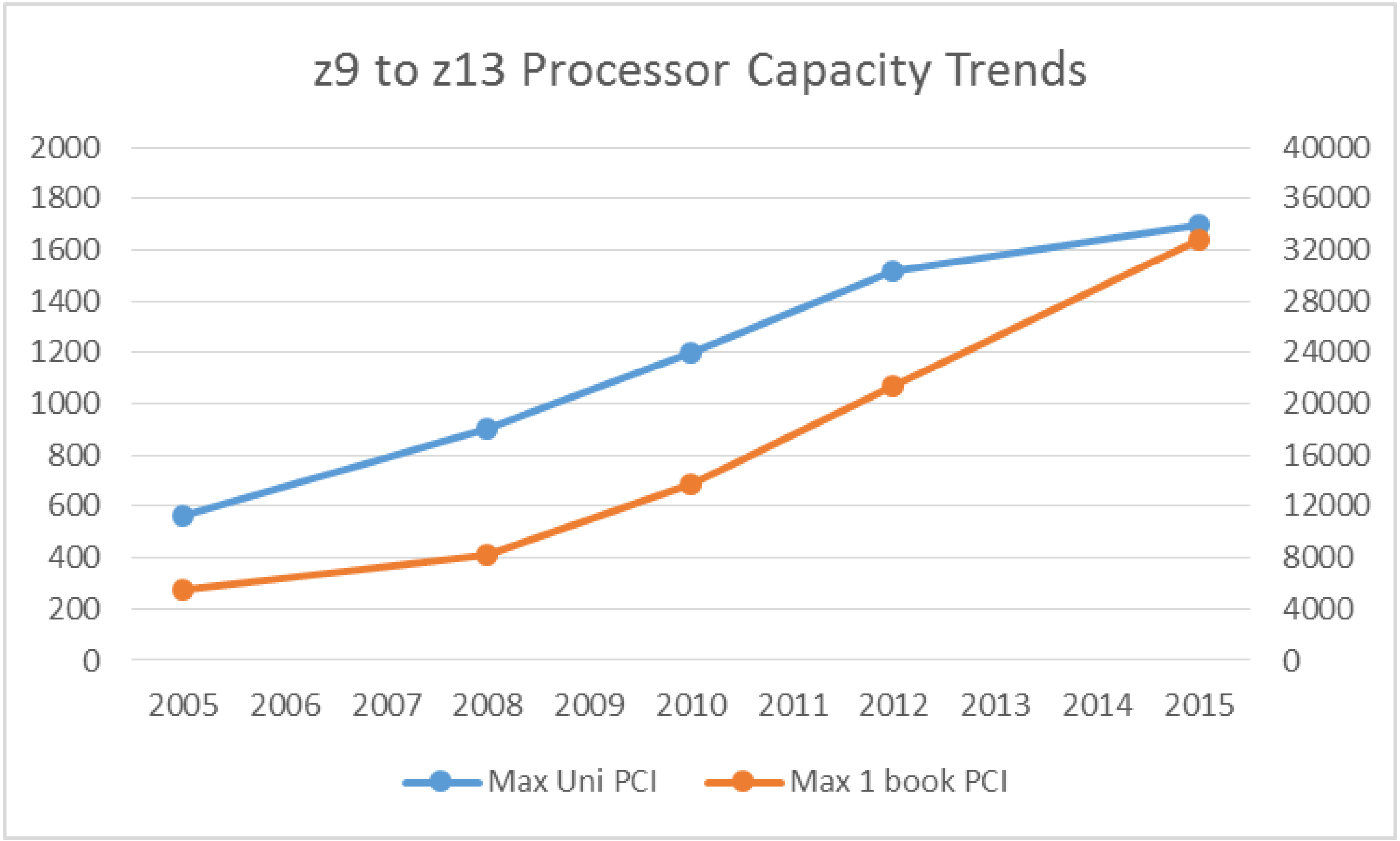
Scott Chapman

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

Agenda

- Review of processor speed
- Hierarchy of data accesses
- Review of processor caches and DAT
- Considerations: uses for more memory and things to watch out for
- Measurements you might want to track

Processors get faster and smarter



How do we keep the processor busy?

- Disk storage has gotten denser but not significantly faster in last 10 years
  - Average access times in low single-digit ms now common
  - But 10 years ago we were pushing access times into low single-digit ms range as well, just not as consistently

Clock cycles and effective capacity

- Ideally, you'd like to get real work done each clock cycle
- z Processor speeds are really fast
  - z10 – 4.4 Ghz
  - z196 – 5.2Ghz
  - zEC12 – 5.5Ghz
  - z13 – 5.0 Ghz

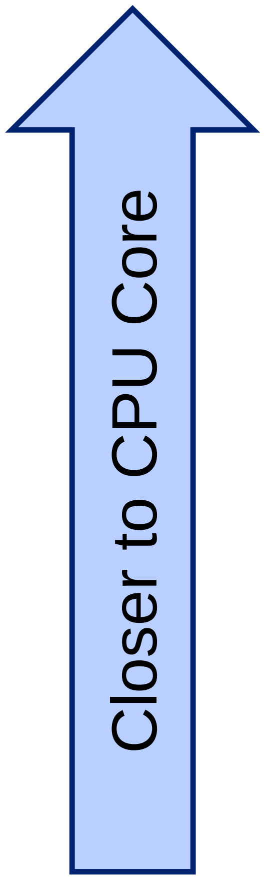
$\left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\}$ Billions of cycles per second
1 Clock cycle = fraction of a ***nano***second

- So 1ms to wait for an I/O = millions of clock cycles

How long is a cycle again?

- Just over 2 inches
  - Light, in a vacuum
  - Electrical signal in a circuit is much slower (40-70% of c)
  - 1 meter in fiber ~ 5 ns
- Need to make a round trip
- Signal paths aren’t as the mosquito flies
  - 7.7 Miles of wire in a zEC12 chip, >13 in z13
- Physical distance matters!

Data access hierarchy



- Register
- Memory
  - L1 Cache
  - L2 Cache
  - L3 Cache
  - L4 Cache
    - Local
    - Remote
  - Real
- Storage Class Memory
- Disk
  - Cache
  - SSD
  - Spinning
- Network

The farther the data is away from the processor, the more clock cycles will be spent accessing it.

Optimal performance & capacity utilization = keeping data as close to processor as possible!

The only good I/O is no I/O

- To keep the processor fed, data needs to be close and ready
  - Disk controller cache can help
  - SSDs can help read response times for cache misses
  - But the disk subsystem is still a long ways away
- Best way to improve I/O performance is to not do the I/O
  - Keep the data in memory
  - Finite limits on memory of course

Why not 4 TB of memory for every LPAR?

$ €

Memory costs

- Memory on z has historically been very expensive compared to other platforms
- Partly due to the robust nature of z memory
  - RAIM greatly enhances error detection and avoidance
  - Can sustain multiple component failures concurrently
    - Bit, lane, DRAM, DIMM, socket, even complete channels
  - Scott’s Mainframe Motto: Quick answers are nice, correct answers are required
  - Not an esoteric point: 2009 Google study: annual incidence of uncorrectable memory errors: 1.3%/machine, 0.22%/DIMM
- But what’s the biggest part of your mainframe budget?
  - Almost always: software, usually by a wide margin
  - (Not including staff costs, which can vary by geography)

Memory prices coming down, sizes going up

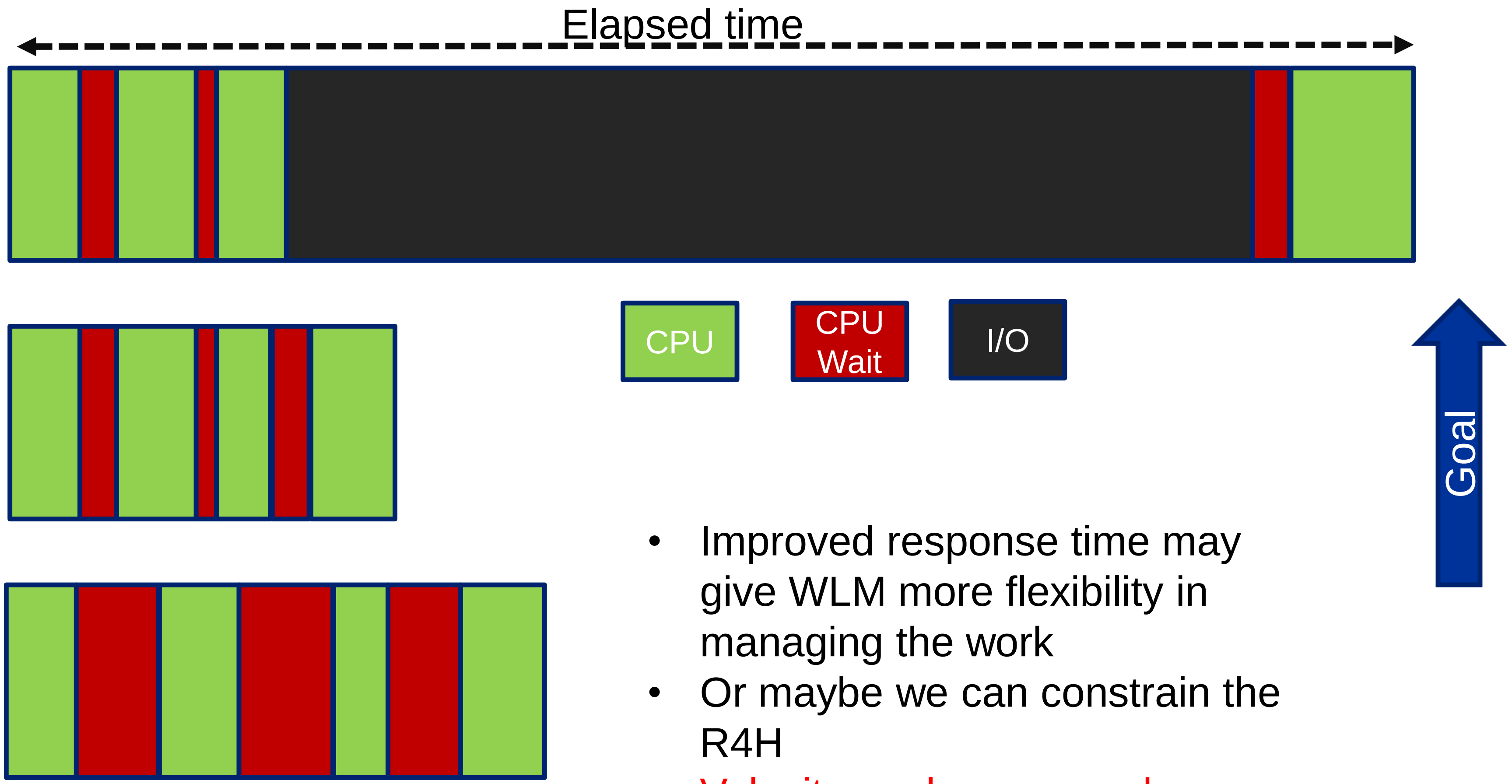
$ €

Year	Machine	Approx Max Memory / CEC	Max Memory / Book
2005	z9EC	0.5 TB	128 GB
2008	z10EC	1.5 TB	384 GB
2010	z196	3 TB	768 GB
2012	zEC12	3 TB	768 GB
2015	z13	10 TB	2560 GB

What can we do with more memory?

- Process more data
  - More data generated today, richer data types
  - Support more dev/test environments
- Performance – I/O avoidance
  - Meet business goals or exploit new business opportunities
  - Offset other constraints
- CPU reduction – generally, it can be fewer cycles spent to get data from memory than disk
  - Also, avoids things being pushed out of cache while waiting on I/O
  - Less CPU generally means less software cost
- Make ourselves more efficient – stop micro-managing storage
  - Staff costs the other big piece of the mainframe budget

Hypothetical Improvements

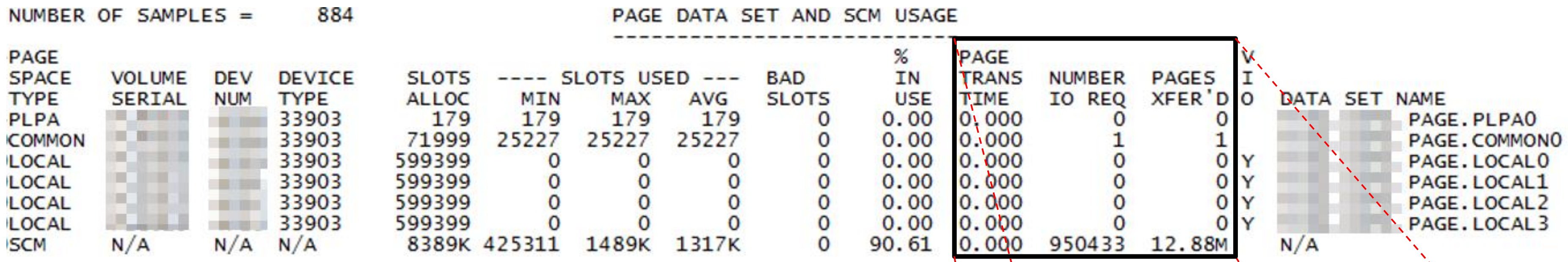


- Improved response time may give WLM more flexibility in managing the work
- Or maybe we can constrain the R4H
- Velocity goals may need adjustment

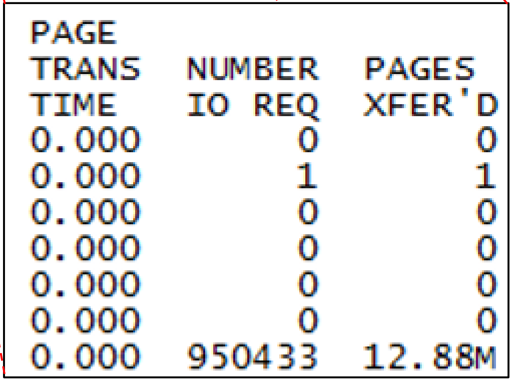
Storage Class Memory (SCM)

- Nowhere near as fast as main memory, but much faster than going to disk
  - Even SSD disk
  - Physically closer (inside the CEC)
  - Avoids going through FICON
- Cheaper than memory, more expensive than disk
- Large relative to current common memory sizes, small relative to disk
  - 1.4TB increments (up to 4 increments/CEC)
- Initial uses:
  - Paging, including pageable 1 MB pages
    - Ideal for large memory configurations
  - CF List structure storage (MQ shared queues)

Exercising SCM (zEC12 4xx)



```
NUMBER OF SAMPLES =      884                    PAGE DATA SET AND SCM USAGE
                                                ---------------------------
PAGE                                                                       %   PAGE                     V
SPACE    VOLUME   DEV  DEVICE     SLOTS  ---- SLOTS USED ---   BAD        IN   TRANS   NUMBER   PAGES   I
TYPE     SERIAL   NUM  TYPE       ALLOC     MIN     MAX    AVG SLOTS      USE  TIME    IO REQ  XFER'D  O  DATA SET NAME
PLPA   [redacted] [redacted] 33903   179     179     179    179    0     0.00  0.000        0       0     [redacted] PAGE.PLPA0
COMMON [redacted] [redacted] 33903 71999   25227   25227  25227    0     0.00  0.000        1       1     [redacted] PAGE.COMMON0
LOCAL  [redacted] [redacted] 33903 599399      0       0      0    0     0.00  0.000        0       0  Y  [redacted] PAGE.LOCAL0
LOCAL  [redacted] [redacted] 33903 599399      0       0      0    0     0.00  0.000        0       0  Y  [redacted] PAGE.LOCAL1
LOCAL  [redacted] [redacted] 33903 599399      0       0      0    0     0.00  0.000        0       0  Y  [redacted] PAGE.LOCAL2
LOCAL  [redacted] [redacted] 33903 599399      0       0      0    0     0.00  0.000        0       0  Y  [redacted] PAGE.LOCAL3
SCM      N/A      N/A  N/A       8389K  425311   1489K  1317K    0    90.61  0.000   950433  12.88M     N/A
```



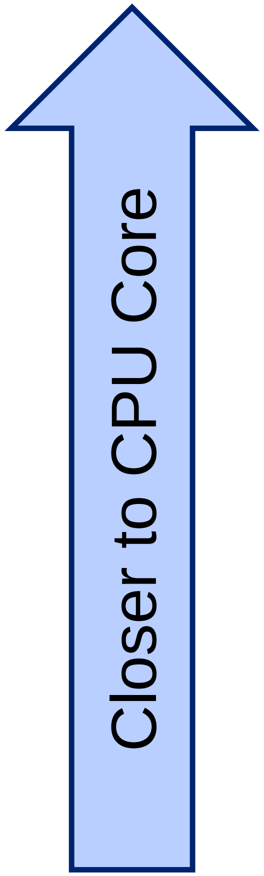
```
PAGE
TRANS   NUMBER    PAGES
TIME    IO REQ   XFER'D
0.000        0        0
0.000        1        1
0.000        0        0
0.000        0        0
0.000        0        0
0.000        0        0
0.000   950433   12.88M
```

- Note that "pages" really means 4K equivalent pages
- My calculations:
  - Average paging rate of 14311/sec
  - Average I/O rate of about 1056/sec
  - Page transfer time was about 0.000063 (63 microseconds)
- Much slower than memory, much faster than disk

Remember…

- Paging is not free—even with SCM!
- Avoid paging for production address spaces
  - Even though DB2 supports pageable large pages, you're probably better off page-fixing them
- But some SCM paging for dev/test regions might be acceptable
  - Depending on usage patterns
  - Allow idle environments to page out / in as they are needed
  - Maybe allow more dev/test regions without more real memory

Going deeper in the hierarchy

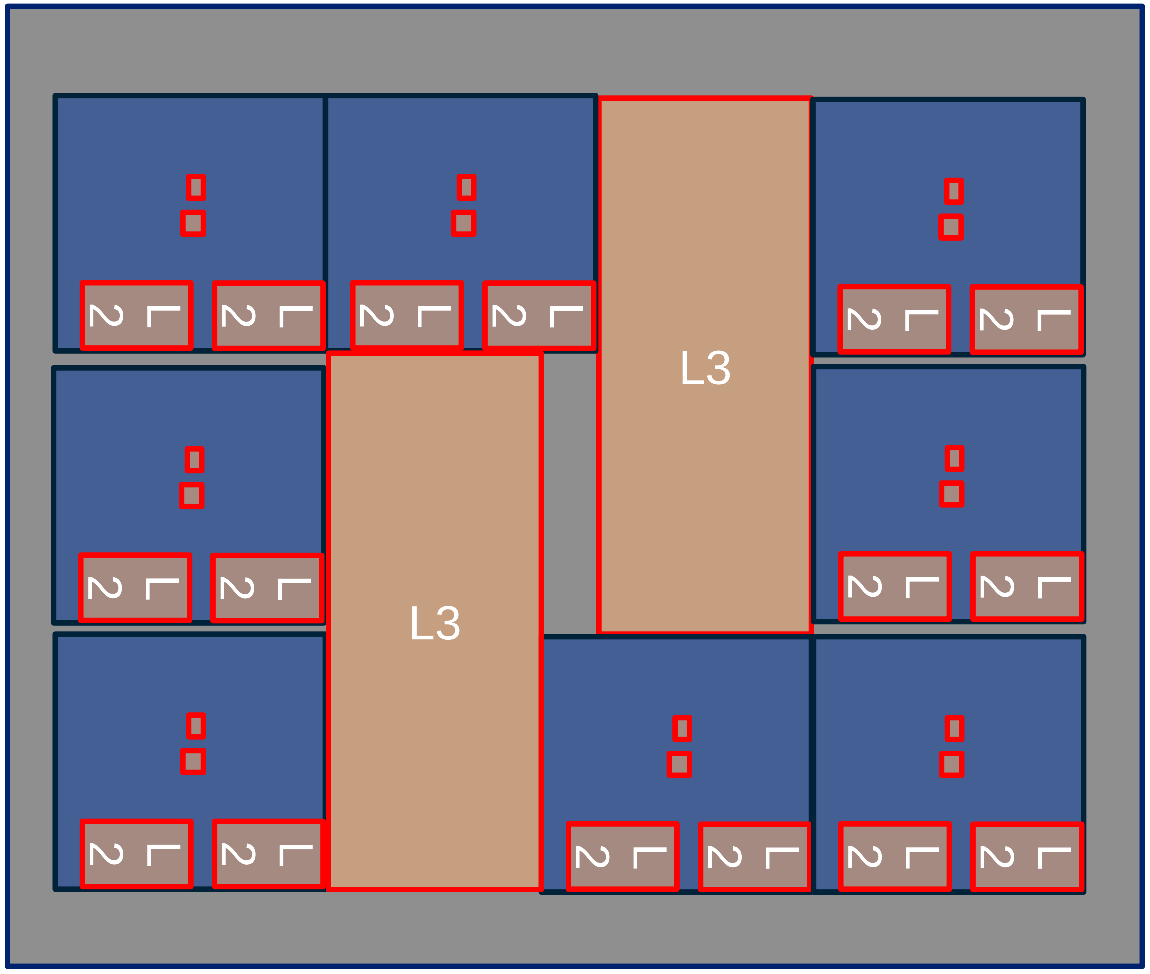


- Register
- Memory
  - L1 Cache
  - L2 Cache
  - L3 Cache
  - L4 Cache
    - Local
    - Remote
  - Real
- Storage Class Memory
- Disk
  - Cache
  - SSD
  - Spinning
- Network

The farther the data is away from the processor, the more clock cycles will be spent accessing it.

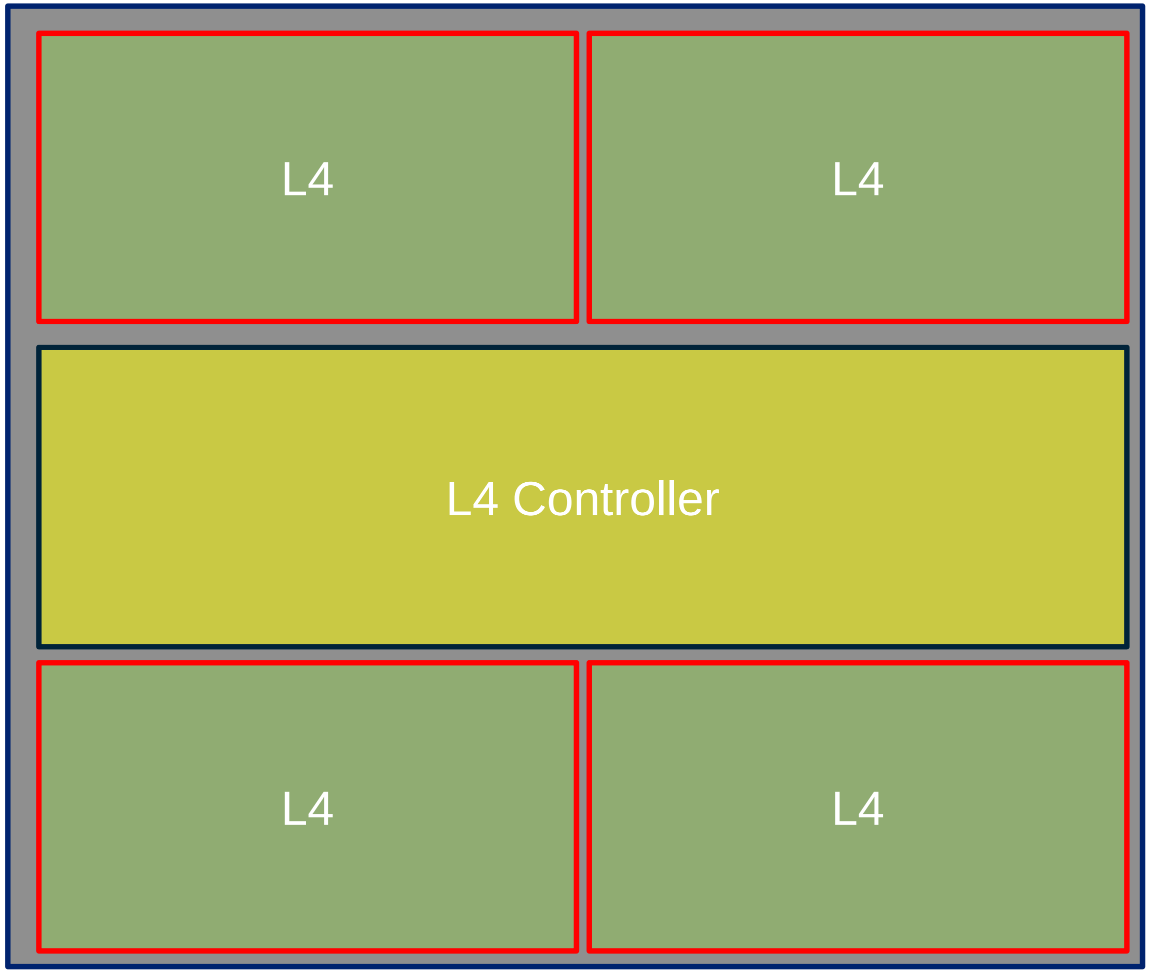
Optimal performance & capacity utilization = keeping data as close to processor as possible!

z13 Processor Chip Schematic



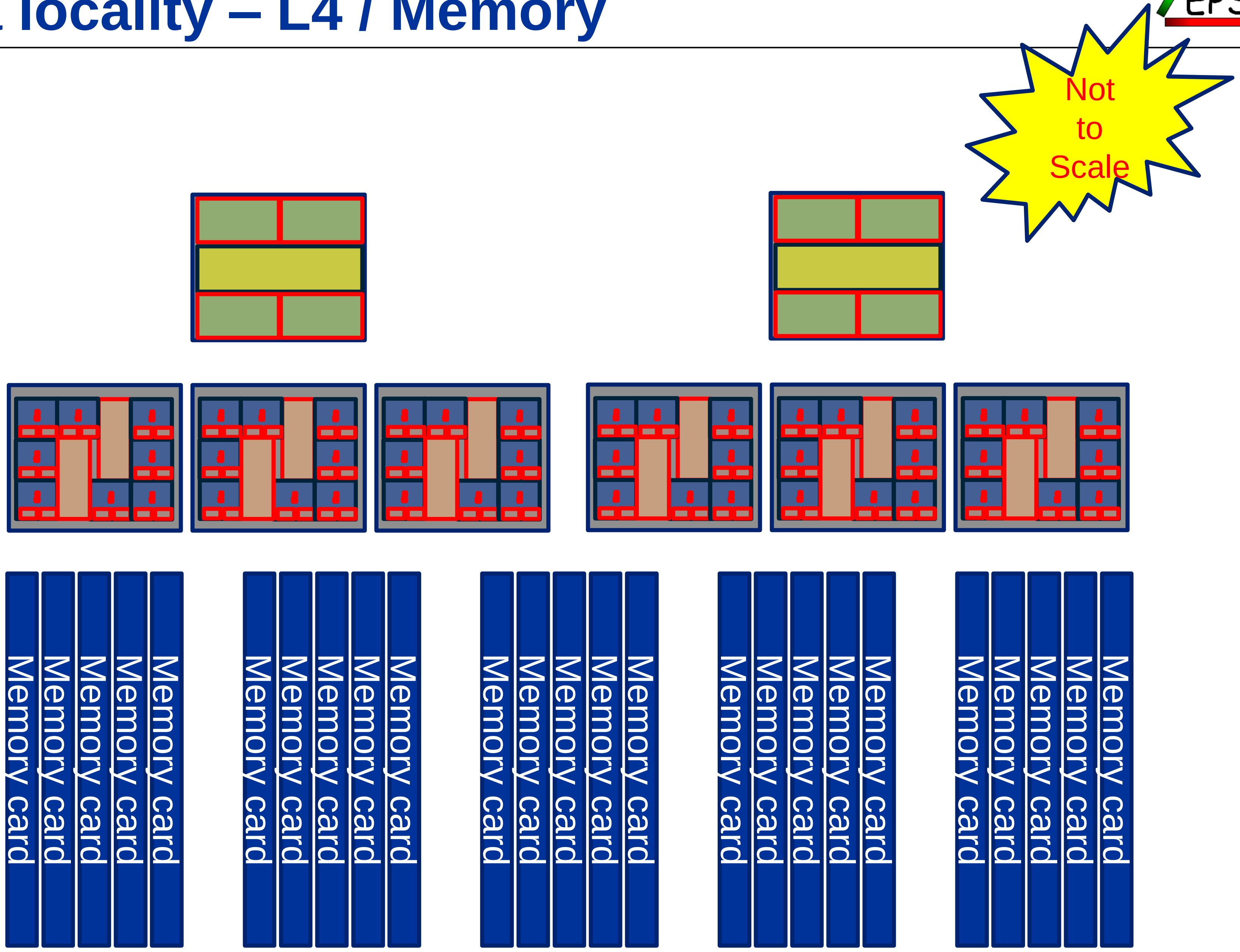
- Approximation based on IBM docs
- Cache sizes scaled relative to each other
- Physical location of L1 cache unclear

z13 Storage Control Chip Schematic



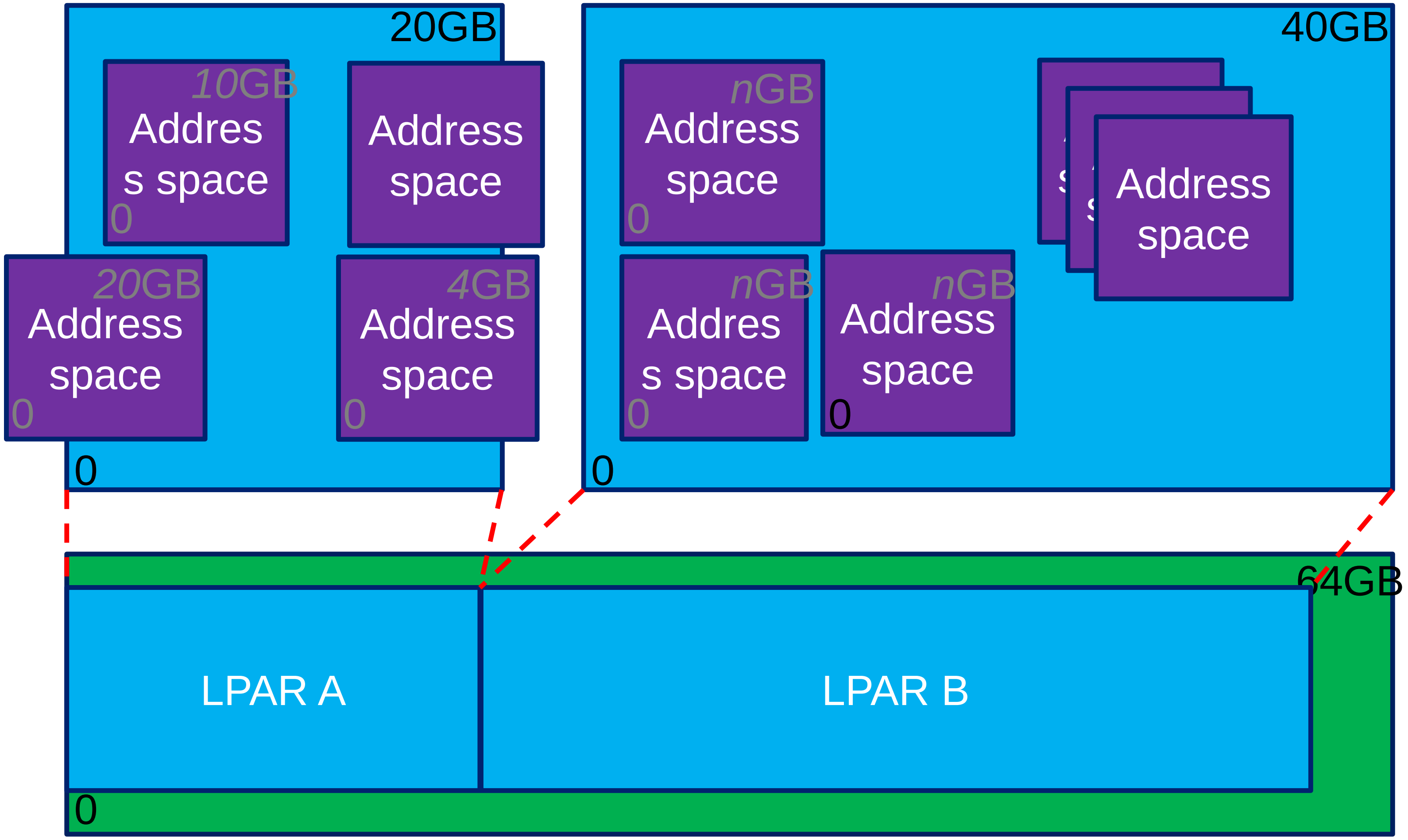
- Approximation based on IBM docs
- NIC directory embedded in the 4 L4 areas
- L4 Controller schematic simplified

Data locality – L4 / Memory



Memory address translation

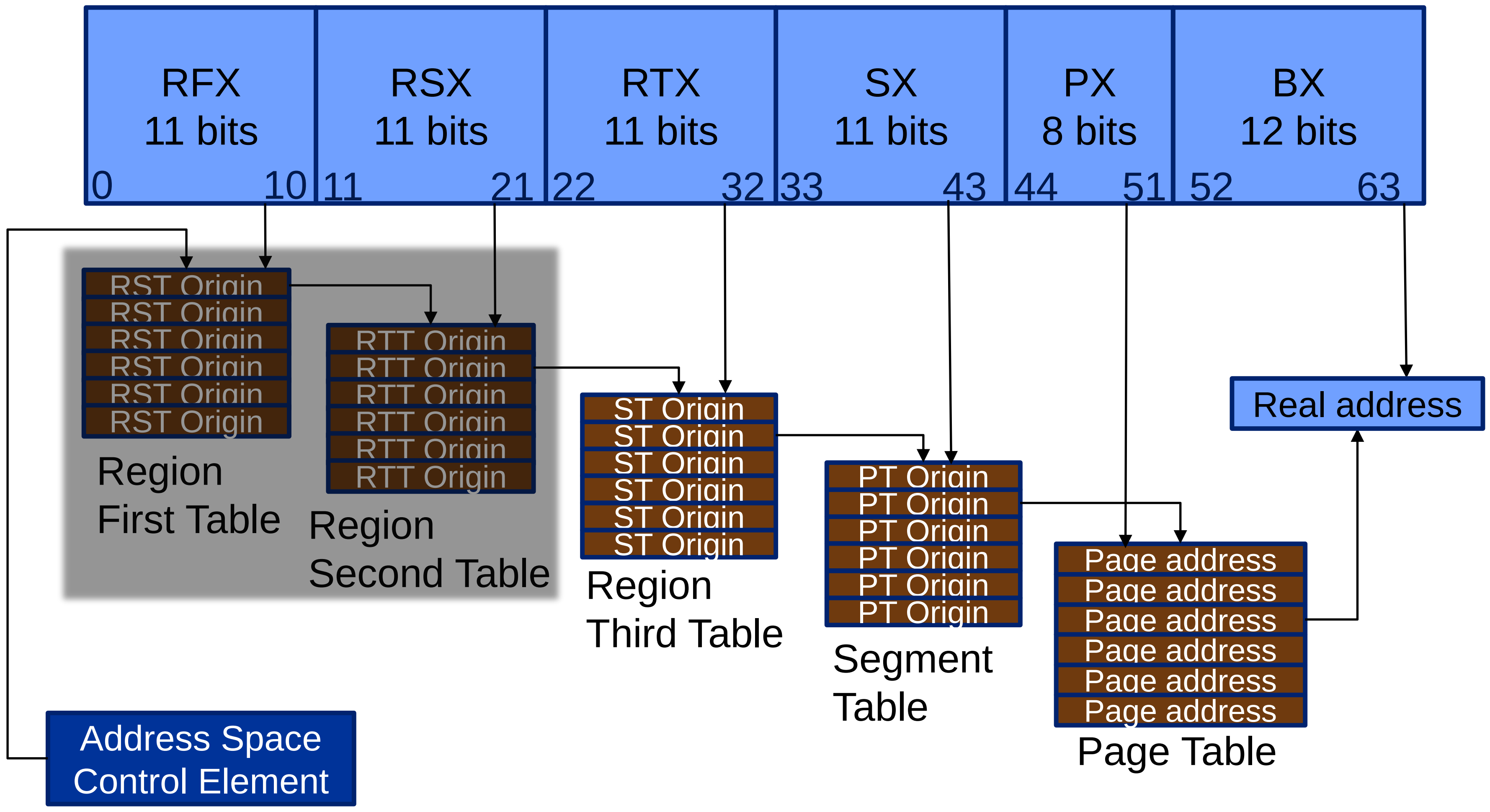
Dynamic Address Translation & Prefixing: Virtual -> Real -> Absolute



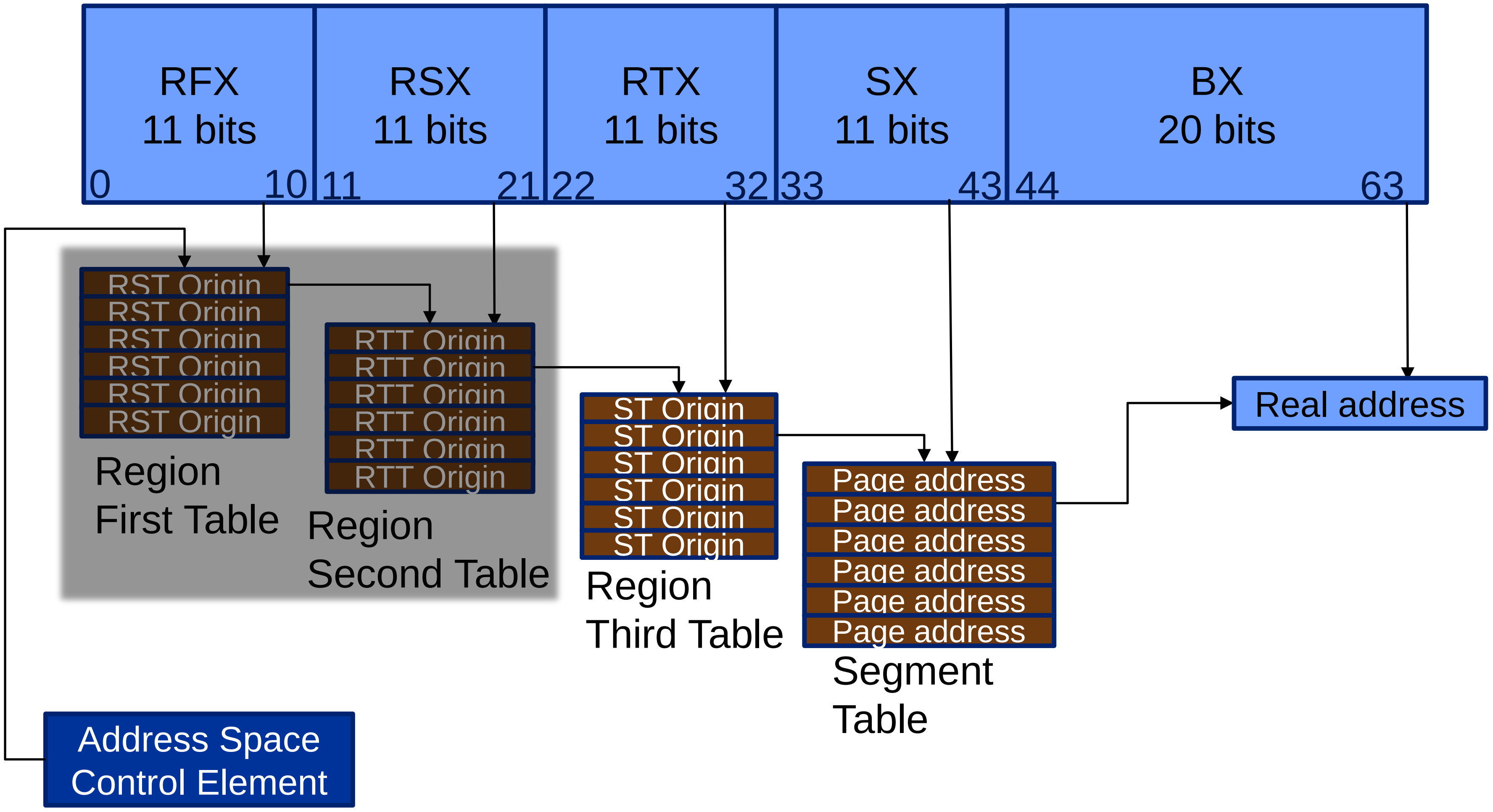
Dynamic Address Translation

- DAT performed using multiple tables that point to different ranges of storage
- DAT is not free!
- Result of DAT cached in Translation Look-aside Buffers (TLB)
- TLBs are in L1 cache and managed by the hardware
- Relatively small
- Flushed when DAT table changes
- 1MB & 2GB pages make TLBs more effective

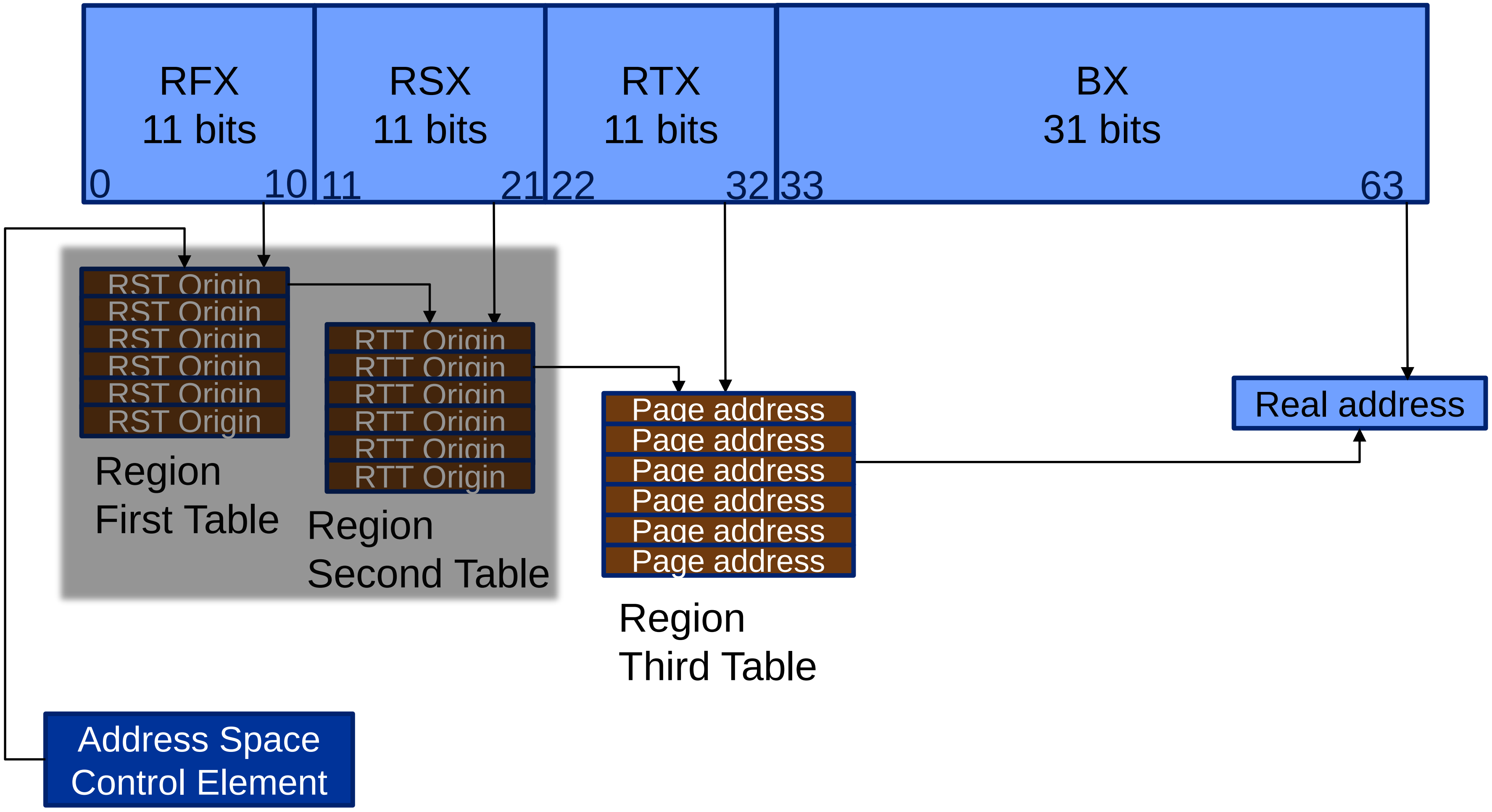
z/OS 64-bit Address Translation



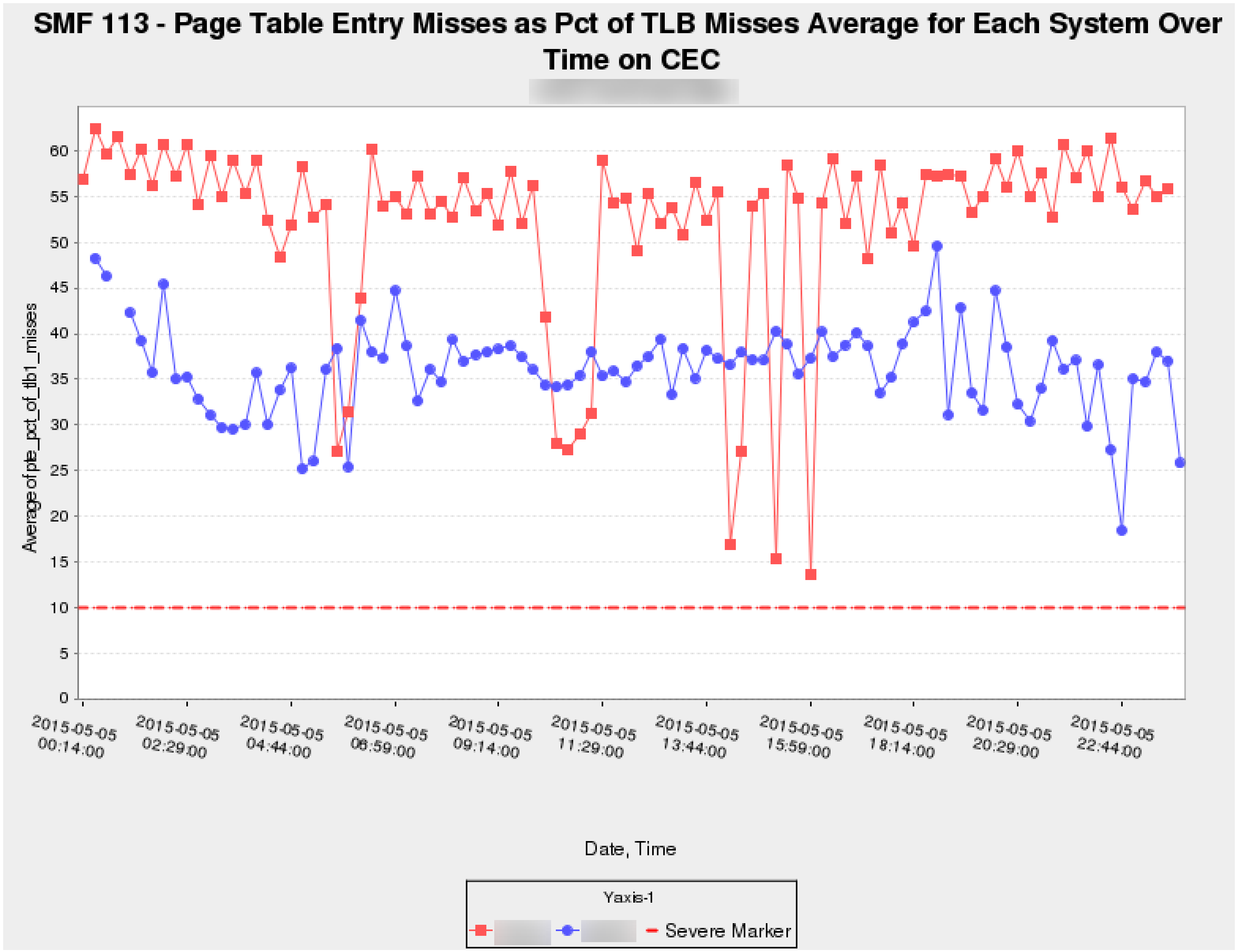
Large Page Address Translation



Giant Page Address Translation



TLB Misses attributed to PTE misses



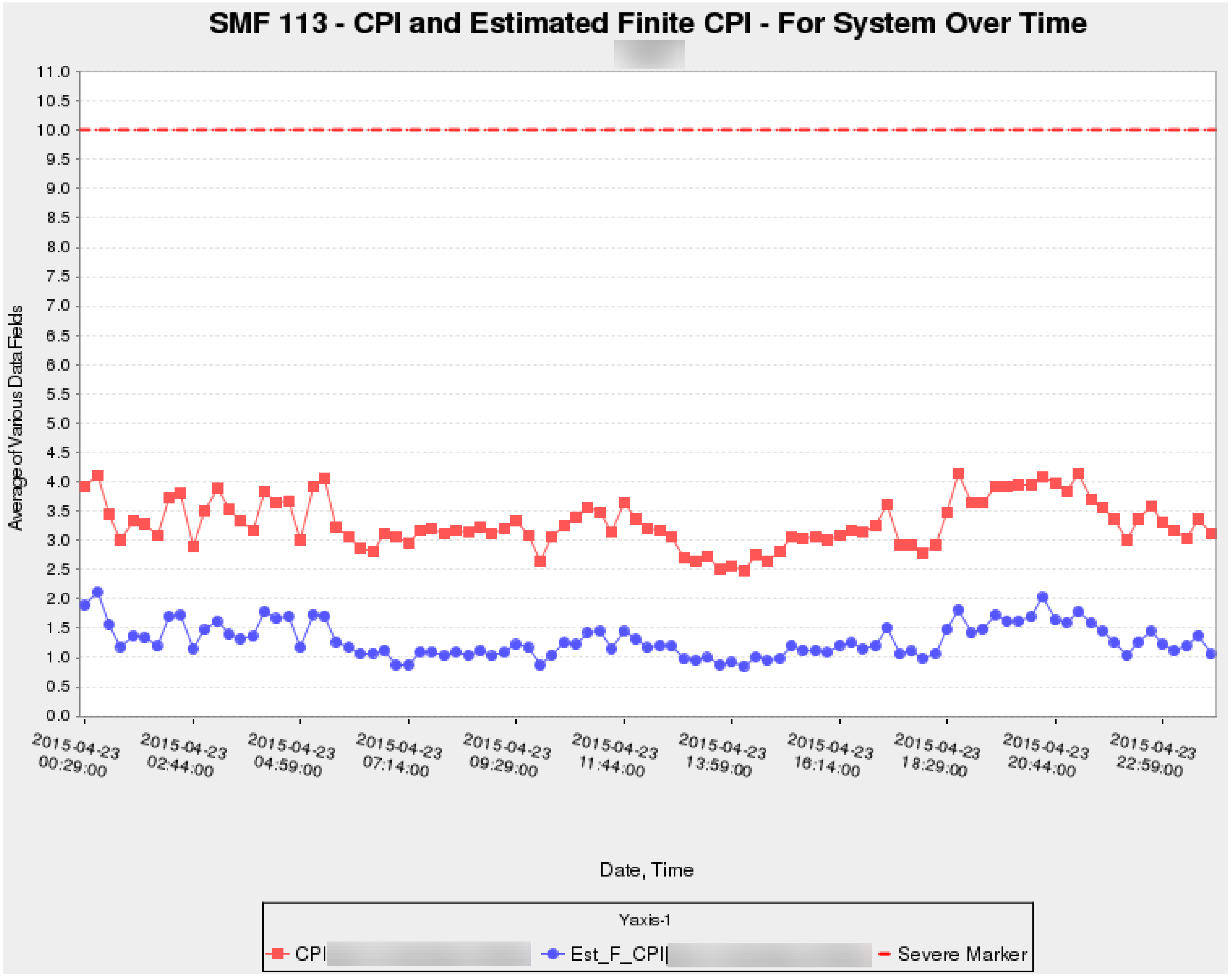
Cache utilization & performance

- Memory is far away from the processor core and relatively slow
- Effective use of processor cache is important to keeping the processor “fed”
- Cache effectiveness measurements are in the Hardware Instrumentation Services SMF 113 records
  - Requires z/OS 1.8 +PTFs & z10 GA2
- Enable HIS and record the 113 records
  - Required for effective capacity planning on upgrade

SMF 113 Cache Measurements

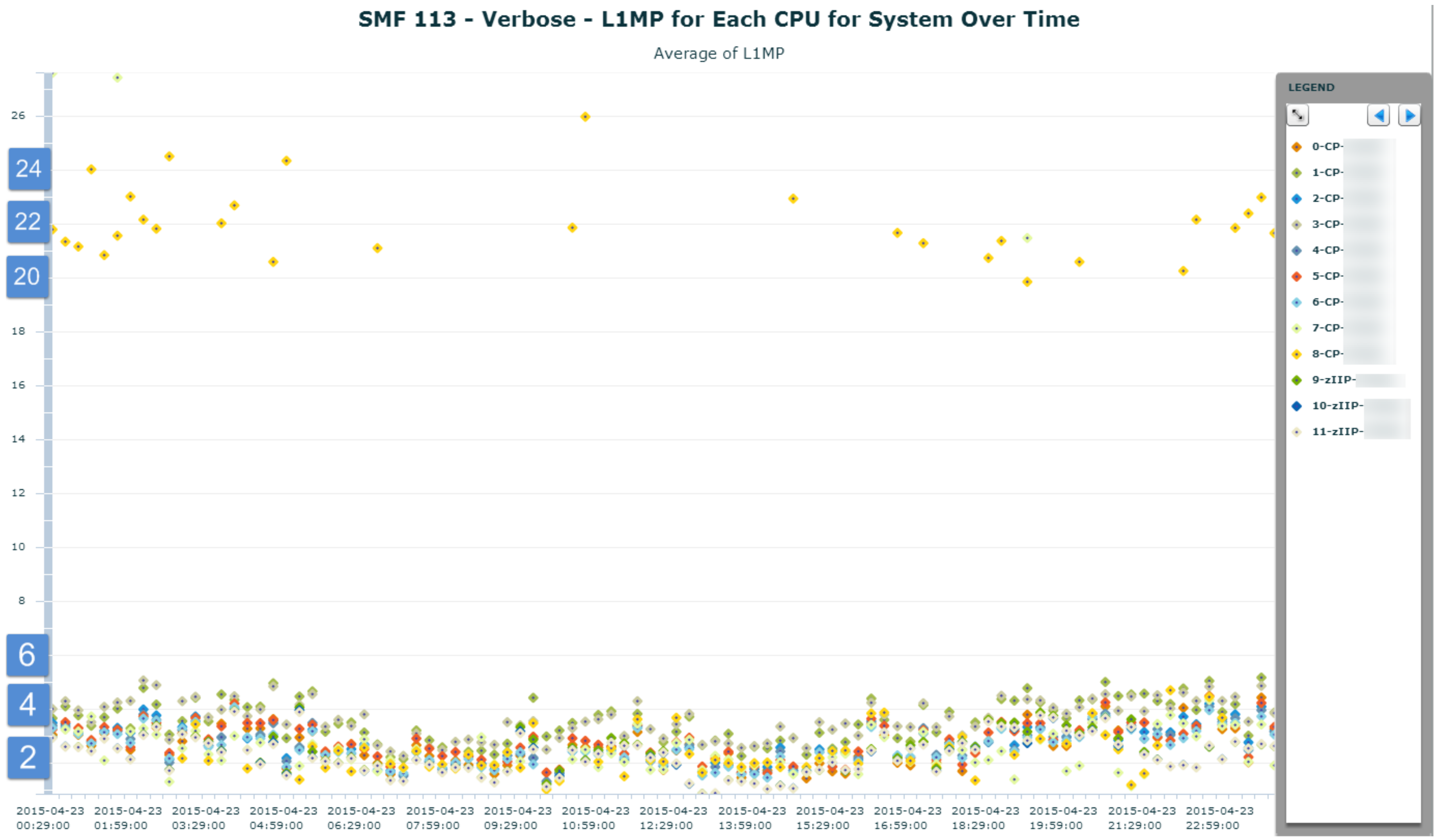
- L1MP – Level 1 Misses per 100 Instructions
- CPI – Cycles Per Instruction
  - Estimated Finite CPI – effectively: penalty cycles per instruction due to the fact that caches are finite
  - Estimated Instruction Complexity CPI – CPI as if there were no penalty cycles (completely effective L1 cache)
- RNI – Relative Nest Intensity
  - Combined calculation for effect of overheads to get to cache and memory
  - Used for capacity planning on upgrade
- TLB CPU Miss Percent of CPU – How much CPU time goes to resolving DAT during TLB misses

SMF113 CPI

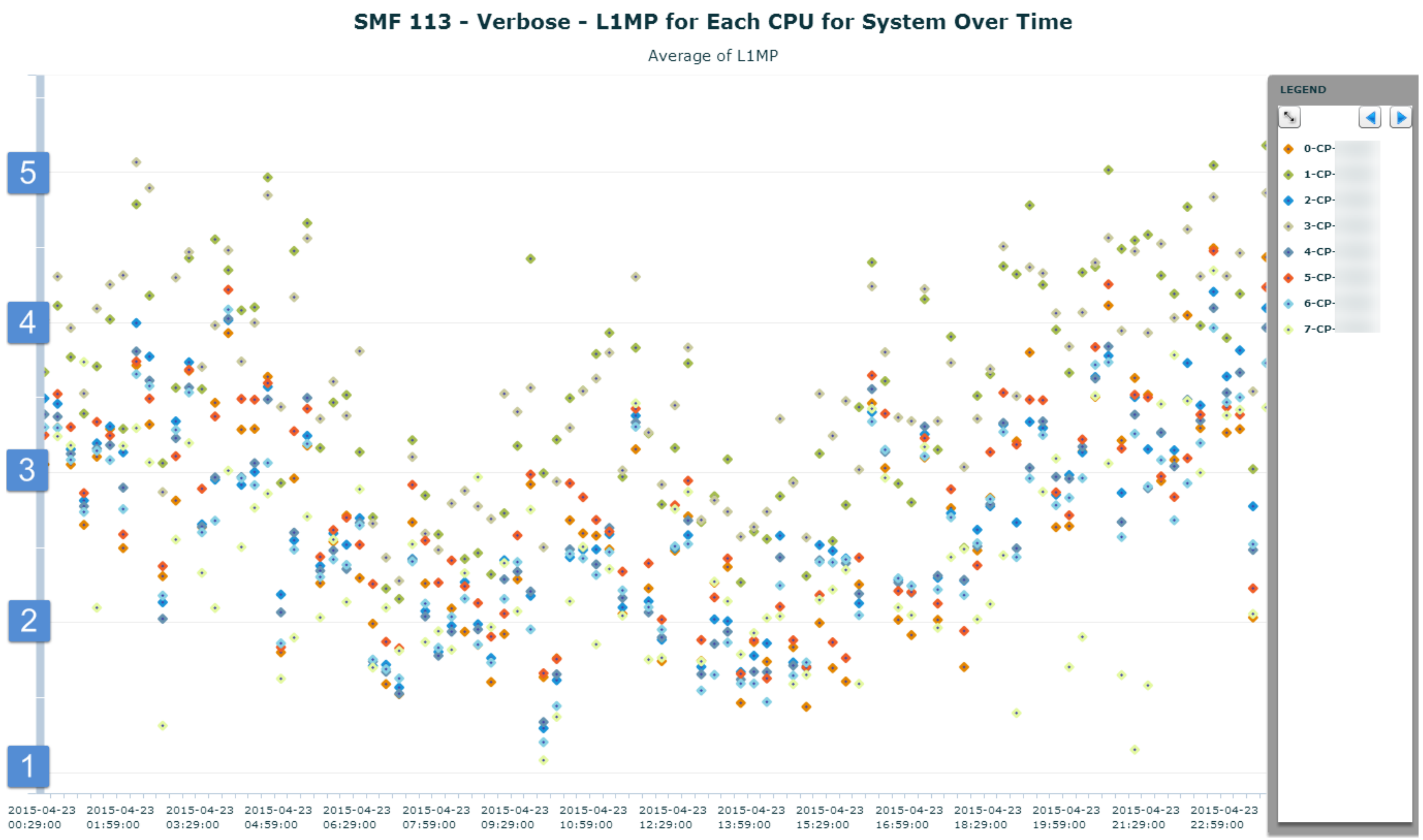


SMF 113 - CPI and Estimated Finite CPI - For System Over Time
Average of Various Data Fields
11.0
10.5
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2015-04-23 00:29:00
2015-04-23 02:44:00
2015-04-23 04:59:00
2015-04-23 07:14:00
2015-04-23 09:29:00
2015-04-23 11:44:00
2015-04-23 13:59:00
2015-04-23 16:14:00
2015-04-23 18:29:00
2015-04-23 20:44:00
2015-04-23 22:59:00
Date, Time
Yaxis-1
CPI
Est_F_CPI
Severe Marker

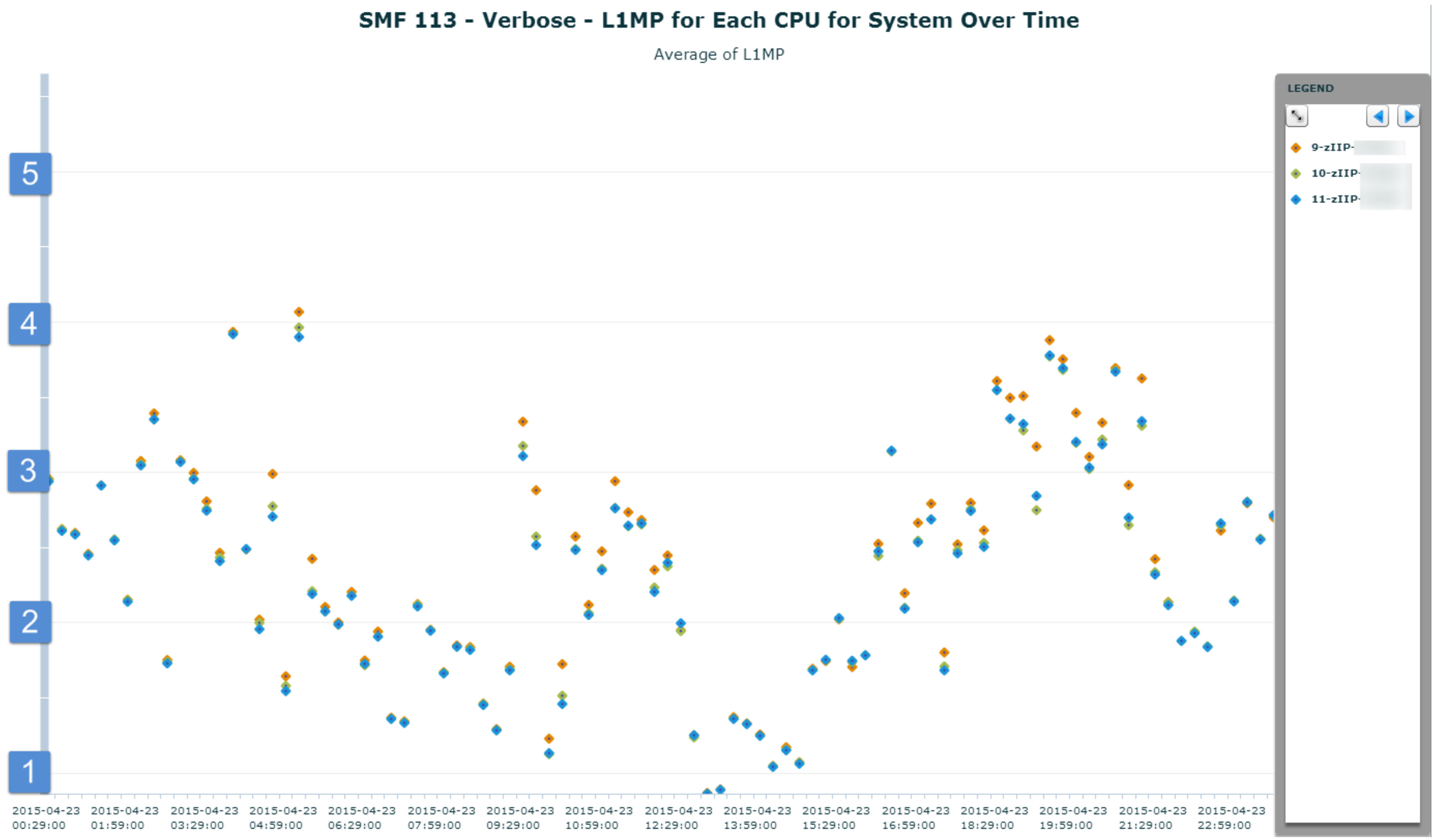
L1MP by CPU


SMF 113 - Verbose - L1MP for Each CPU for System Over Time
Average of L1MP
26
24
22
20
18
16
14
12
10
8
6
4
2
2015-04-23 00:29:00
2015-04-23 01:59:00
2015-04-23 03:29:00
2015-04-23 04:59:00
2015-04-23 06:29:00
2015-04-23 07:59:00
2015-04-23 09:29:00
2015-04-23 10:59:00
2015-04-23 12:29:00
2015-04-23 13:59:00
2015-04-23 15:29:00
2015-04-23 16:59:00
2015-04-23 18:29:00
2015-04-23 19:59:00
2015-04-23 21:29:00
2015-04-23 22:59:00
LEGEND
0-CP-
1-CP-
2-CP-
3-CP-
4-CP-
5-CP-
6-CP-
7-CP-
8-CP-
9-zIIP-
10-zIIP-
11-zIIP-

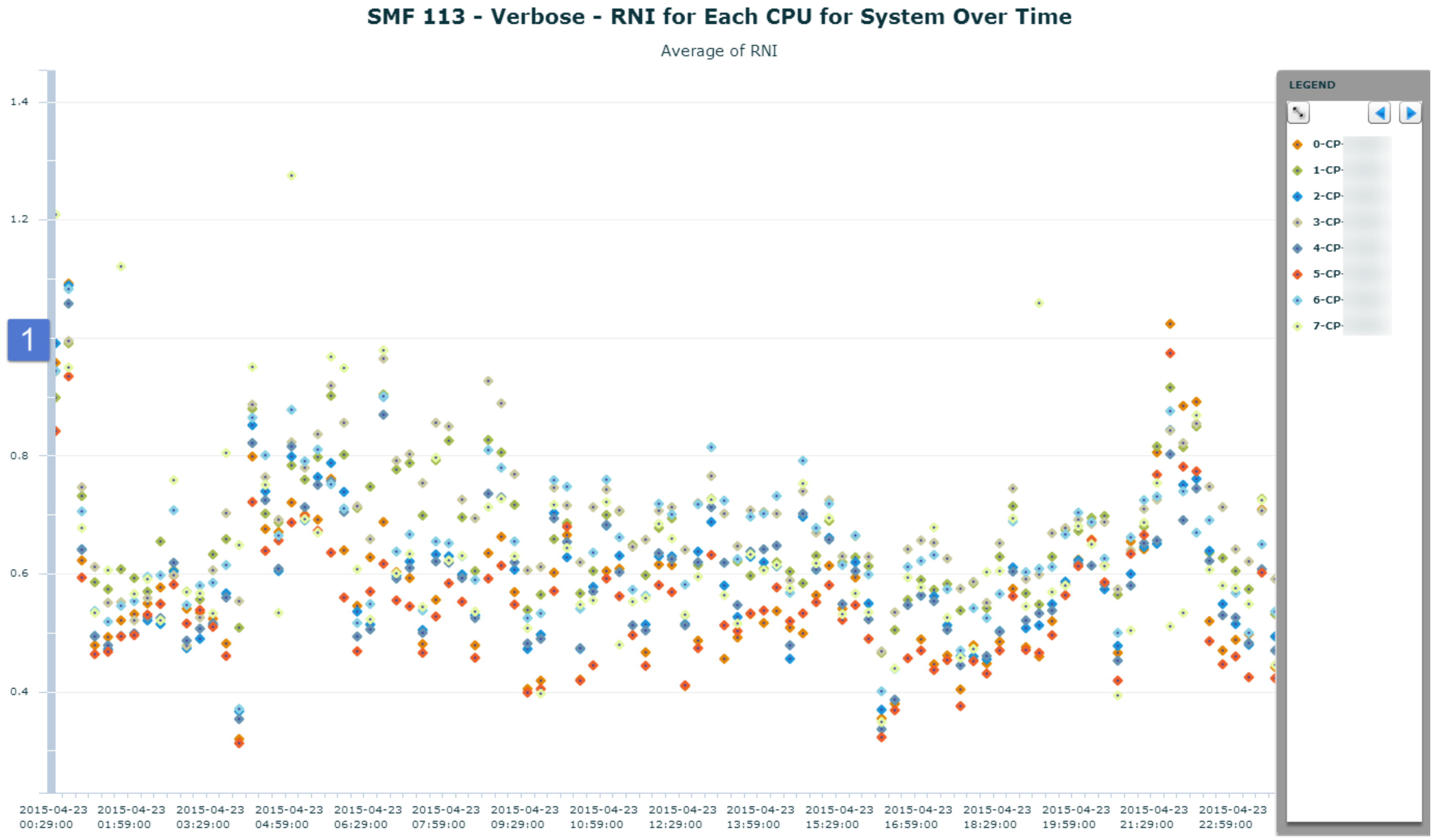
Zoomed in on just CPs



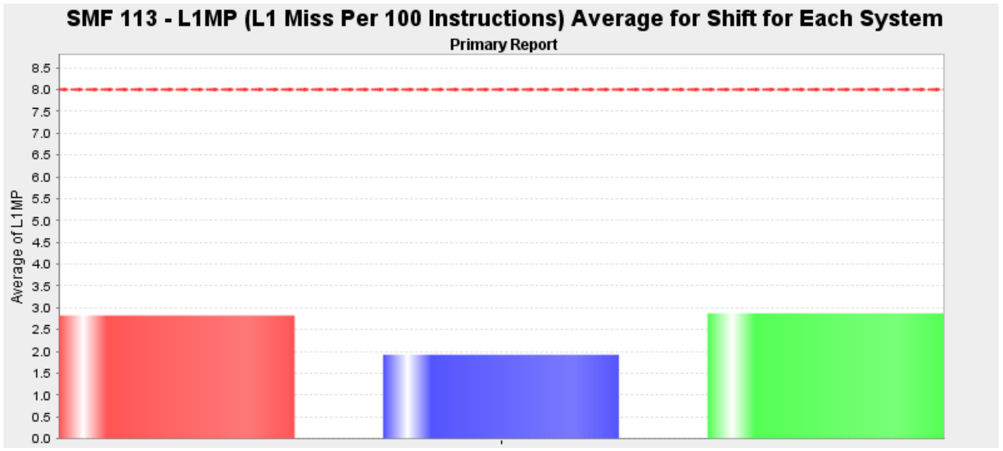
Zoomed in on zIIPs

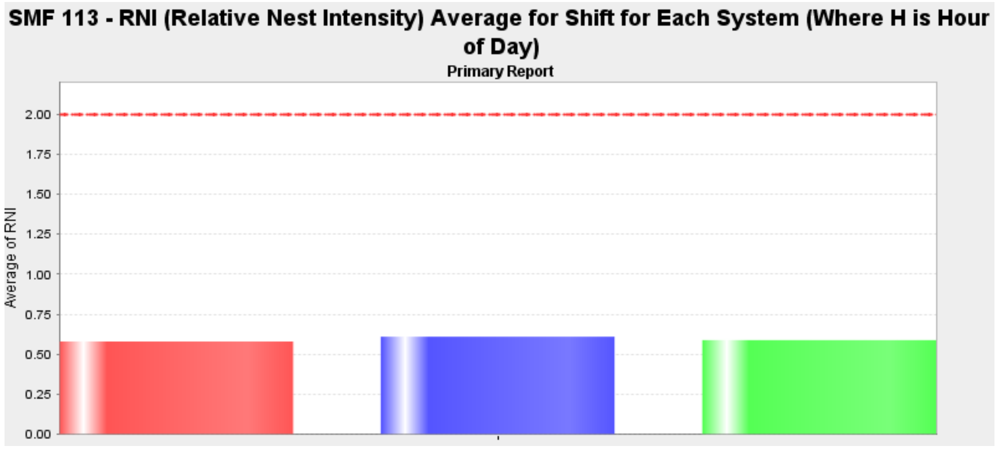


RNI – Evaluate over time



Combine L1MP and RNI for workload “hint”





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

How can you improve cache effectiveness?

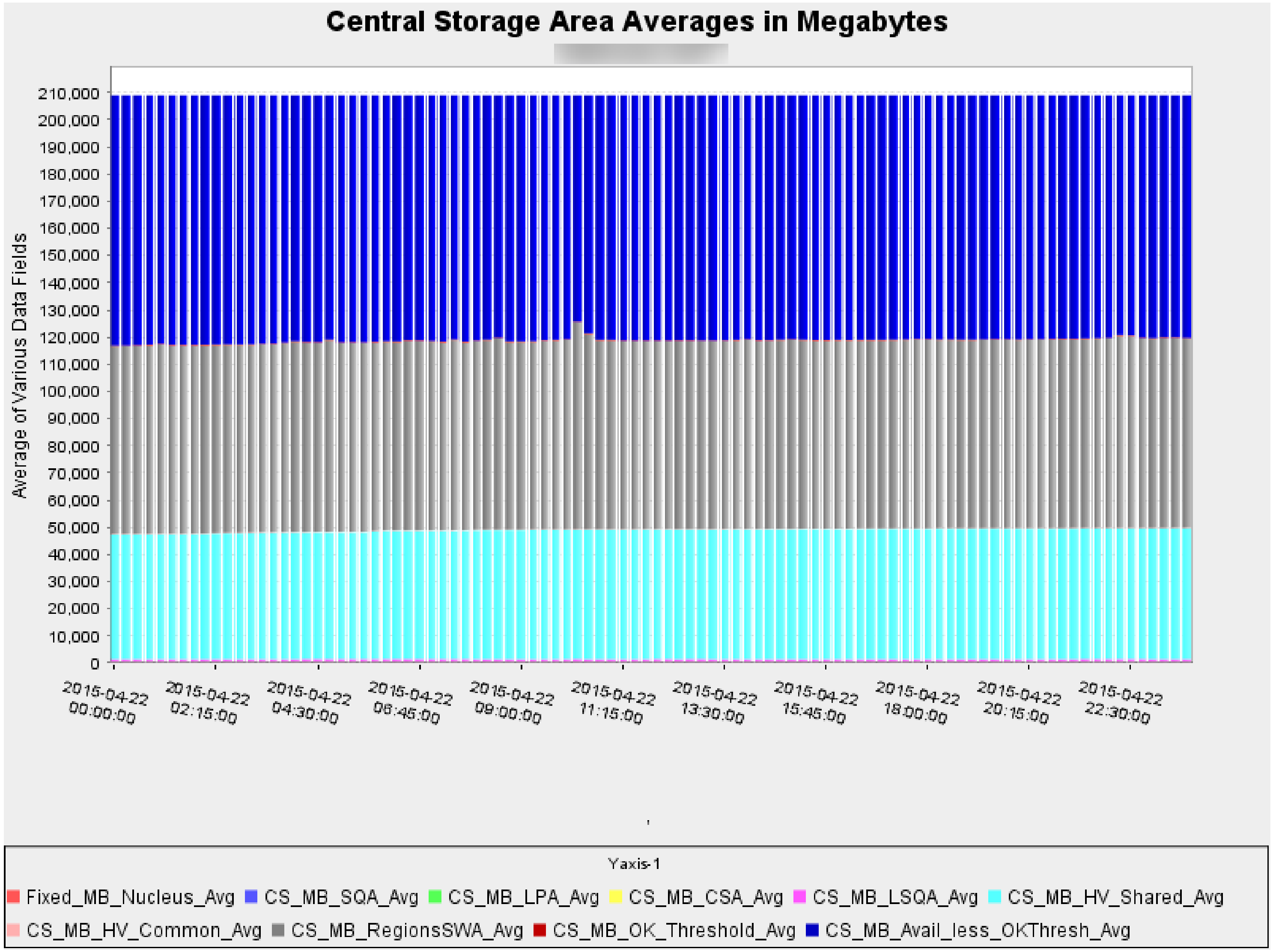
- Enable HiperDispatch
- Make good use of large pages
- Upgrade to newer machine

		Cache sizes (L4/book or drawer)						
		Per CP				Per Chip	NIC dir	Shared
Year	Machine	L1 Data	L1 Inst.	L2 Data	L2 Inst.	L3		L4
2005	z9 EC	256KB	256KB	40MB		n/a		n/a
2008	z10 EC	128KB	64KB	3MB		n/a		48MB
2010	z196	128KB	64KB	1.5MB		24MB		192MB
2012	zEC12	96K	64K	1MB	1MB	48MB		384MB
2015	z13	128K	96K	2MB	2MB	64MB	224MB	480MB

- Consider more/slower CPUs instead of fewer/faster
  - More CPUs = More L1/L2/TLB

Considerations

Are you using all your memory?



Do you have reserved memory?

- "Reserved" = "unused" = not allocated to any LPAR
- Some sites seem to have given all the memory to LPARs
  - This makes responding to new requirements difficult
- Some sites hold back some memory
  - Define some amount of reserved memory to each LPAR
  - Total reserved across all LPARs can be > actual reserved
    - Just means you can't bring it all LPAR's reserve elements online
    - Make sure reserved by LPAR < actual reserved, or won't be able to bring that LPAR's reserve element online
  - Consider specifying RSU=OFFLINE to make it reconfigurable
    - Then you can take all or part of it offline dynamically
    - May limit use of the area, e.g. not preferred for long-term pages
- Scott's recommendation: don't immediately give away all your memory

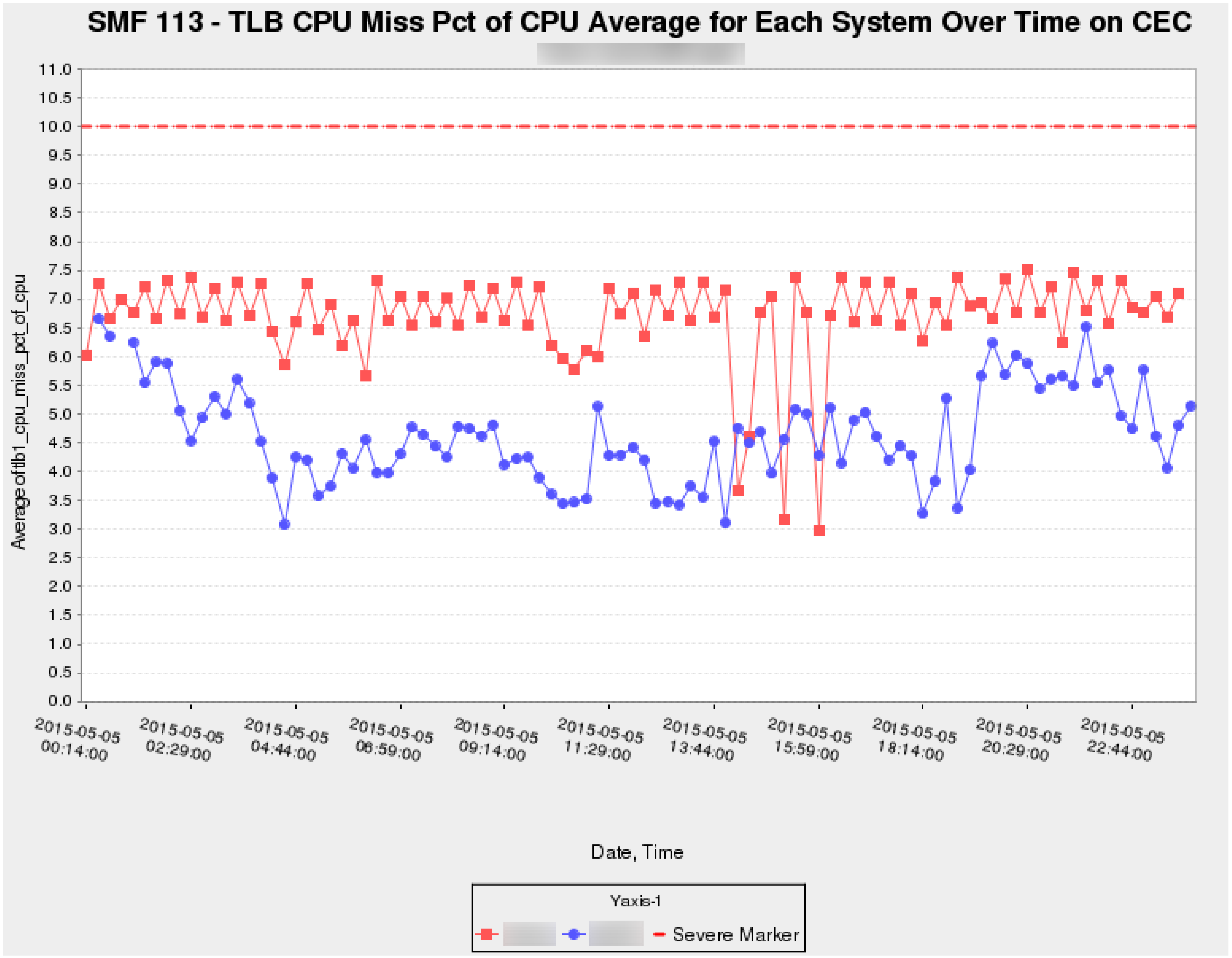
How active will that new memory be?

- How is your total cache size going to increase relative to your memory size?
- This may be a non-issue depending on how you plan on using that new memory
- But it may be something to consider: consider more/slower instead of fewer/faster processors
  - More CPUs = more L1/L2 cache
  - Be sure to investigate single-CP workloads running in unconstrained times before doing this
    - If they're already suffering CPU delay, more/slower CPUs may be fine

TLB miss overhead

- If you add a whole bunch of memory as 4K pages you may see overhead due to TLB misses go up
  - Again, workload-dependent of course
- Consider using 1M pages wherever possible
  - Probably should be the default wherever you can
  - Non-authorized allocation nonpageable large pages must be permitted to IARRSM.LRGPAGES
- Plan your LFA size & check that you got what you thought you were getting
  - DISPLAY VIRTSTOR,LFAREA
- If you have giant amounts of storage, consider giant pages

Estimated impact of TLB Misses



DB2 Buffer Pools

- Going “Biggie Size” on DB2 buffer pools probably a good use of memory
- Start with BPs and/or objects that do the most I/O
  - BPs that don’t do a lot of I/O won’t give you a lot of benefit
- Don’t forget about the group buffer pools
  - Catterall’s ROT is probably still good: sum(local BPs) * 0.4
  - May get benefit from super-sizing the group BPs without increasing local BPs, but I’d prefer to make locals bigger
    - Keep the data as close to the CPU as possible
- Consider PGSTEAL(NONE) in DB2 10+ for pinning objects in a BP
  - DB2 will load the BP on first access and then assume no I/O needed for subsequent access
- Use Large (v10+) or Giant (v11+) pages

Other DB2 areas

- Prepared statement cache
- RID Pool
  - DB2 10 default is 400 MB, will use workfile database if not enough space in RID Pool
- Sort pool
  - Careful: this is per concurrent sort
  - V10 changed default to 10MB, max 128MB
- Utilities: consider giving more memory for improved performance

Sort

- Some shops restrict sort’s use of memory, either all the time, or perhaps at certain times of the day
- For example: restrict sort’s use of memory during the online day
  - This thinking made a lot of sense when memory was extremely precious
  - Today, CPU cycles may be more precious than memory because increased CPU usage = increased software cost
- It may be time to rethink your sort memory limits
  - But don’t crazy – Sort can run away with a lot of memory really quickly

Eliminate IEFUSI limits??

- IEFUSI (SMF Step Initiation Exit) is often used to limit the amount of memory a job (or STC) step can acquire
  - The idea being to prevent something from running away with lots of storage and causing severe performance issues
- Scott’s opinion: IEFUSI is still a good idea, but rethink limits
  - It used to be that most application jobs wouldn’t need more than 10s of MBs
  - Today Java batch jobs could easily use a GB or more
- Don’t forget to limit 64-bit storage

What about more Java?

- Java is definitely viable on the mainframe
- Java programmers may be more generally available than COBOL
  - JZOS API allows easy access to z/OS constructs
- Running on the specialty engines can save money
- One of the concerns in the past has been that Java will almost certainly require more memory to run the same function
  - If we have more memory, this is *less* of a concern, but…

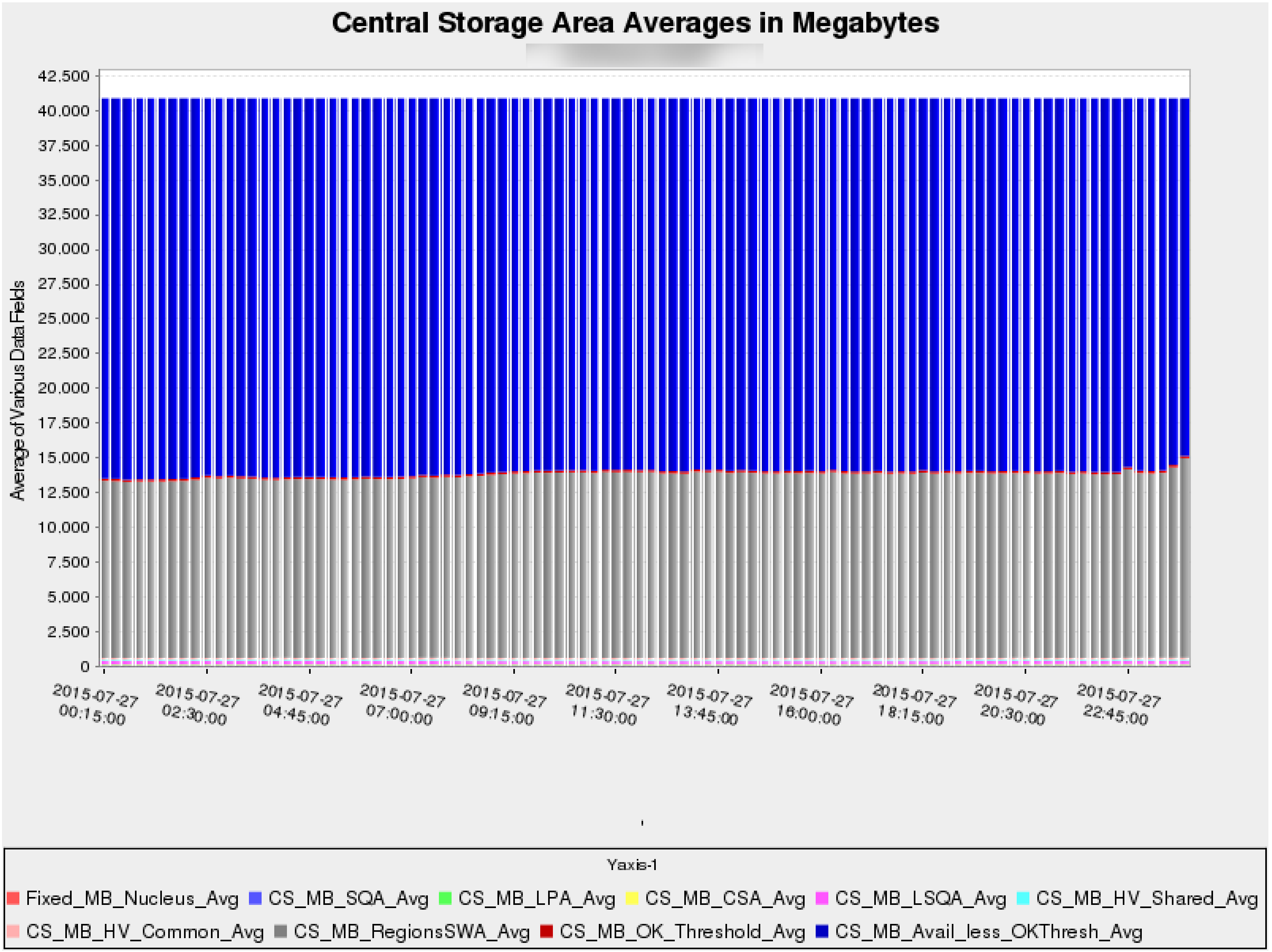
Avoid unnecessarily large JVMs

- If you give a long-running JVM an arbitrarily large heap, it will use it
  - Garbage collection may cause paging spikes if heap has been paged out
- Avoid micromanagement, but every application doesn't need GBs of heap space by default
- Avoid min heap = max heap until you've determined actual heap requirements
  - Smaller area of activity = better cache locality of reference
- Unfortunately, sizing requires testing
  - But probably not required for many batch programs: give them a max of 128-256MB and many will likely be fine
  - Choice of JDBC driver and settings may influence heap requirements
  - Having IEFUSI enforce some limit is probably a good idea

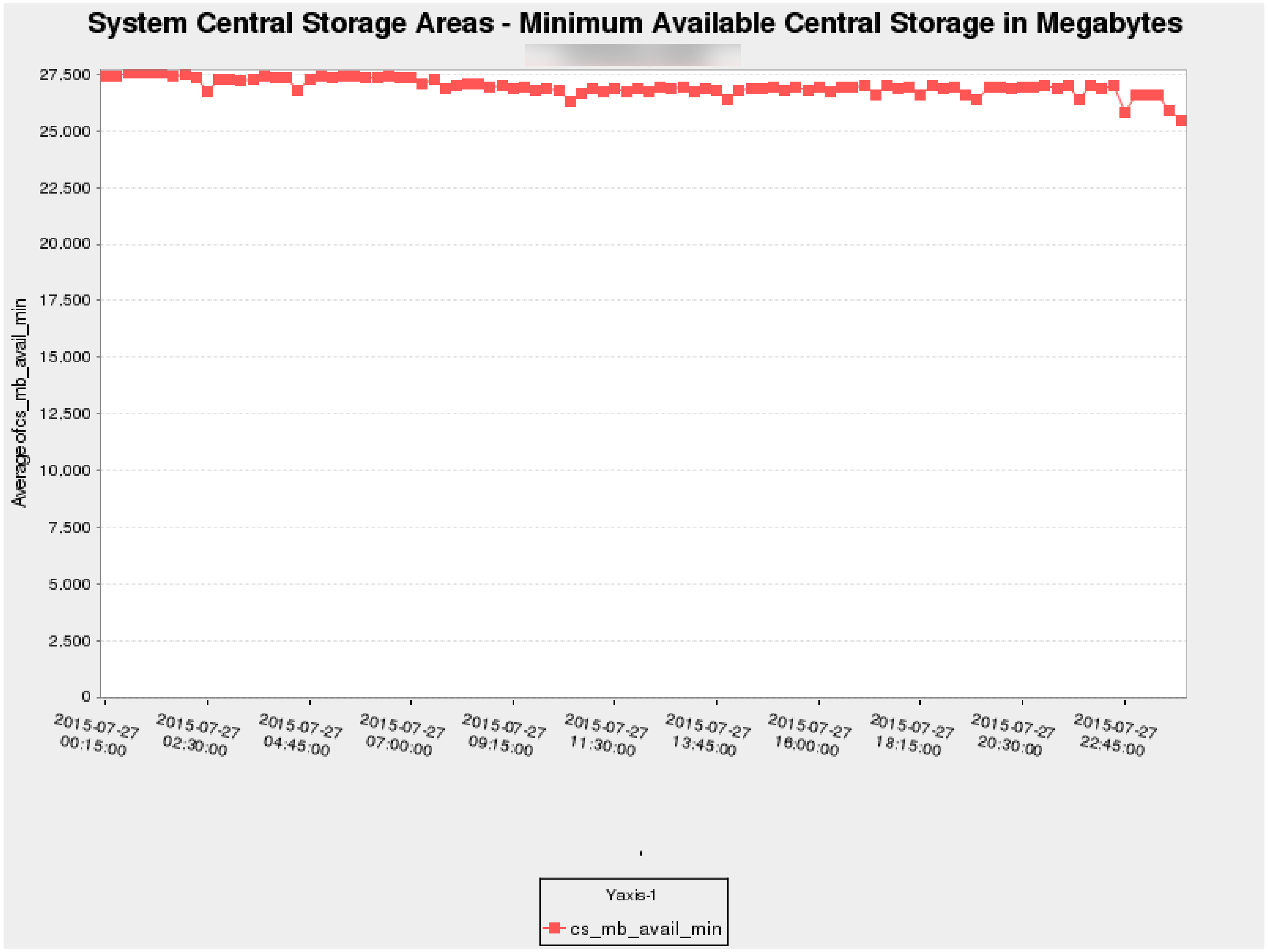
Just a reminder…

Measurements to monitor

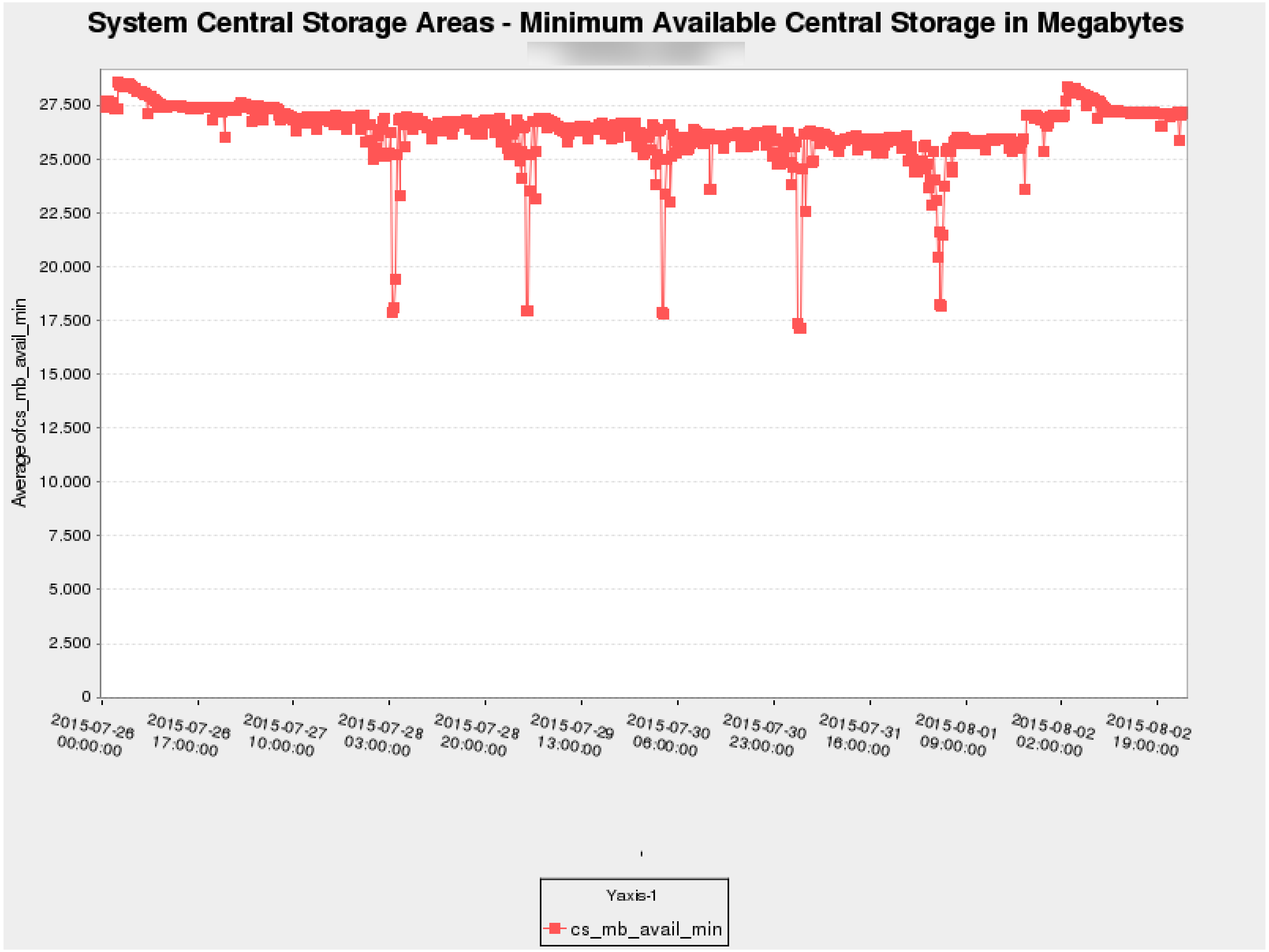
Available storage - Average



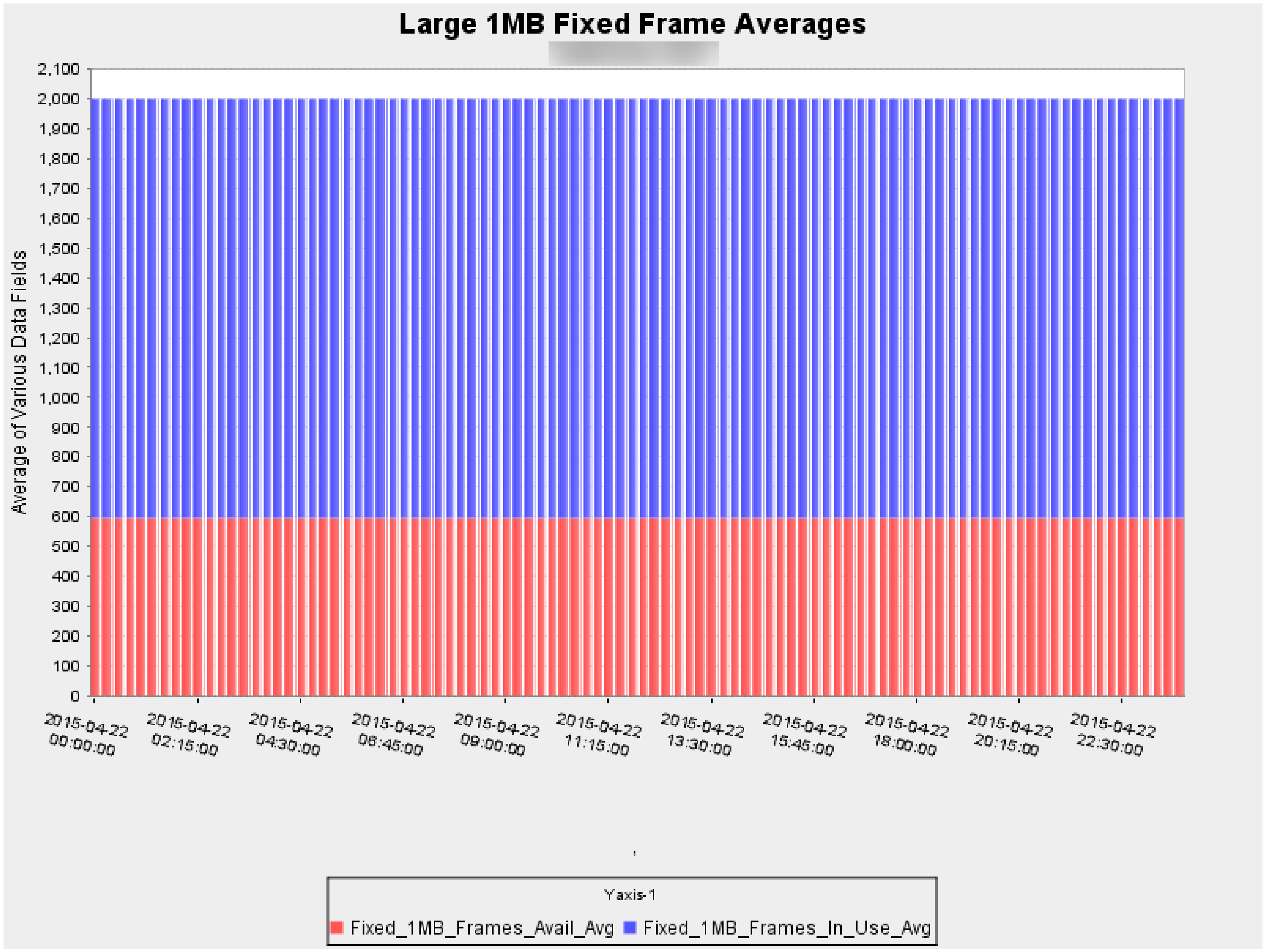
Available storage - Minimum



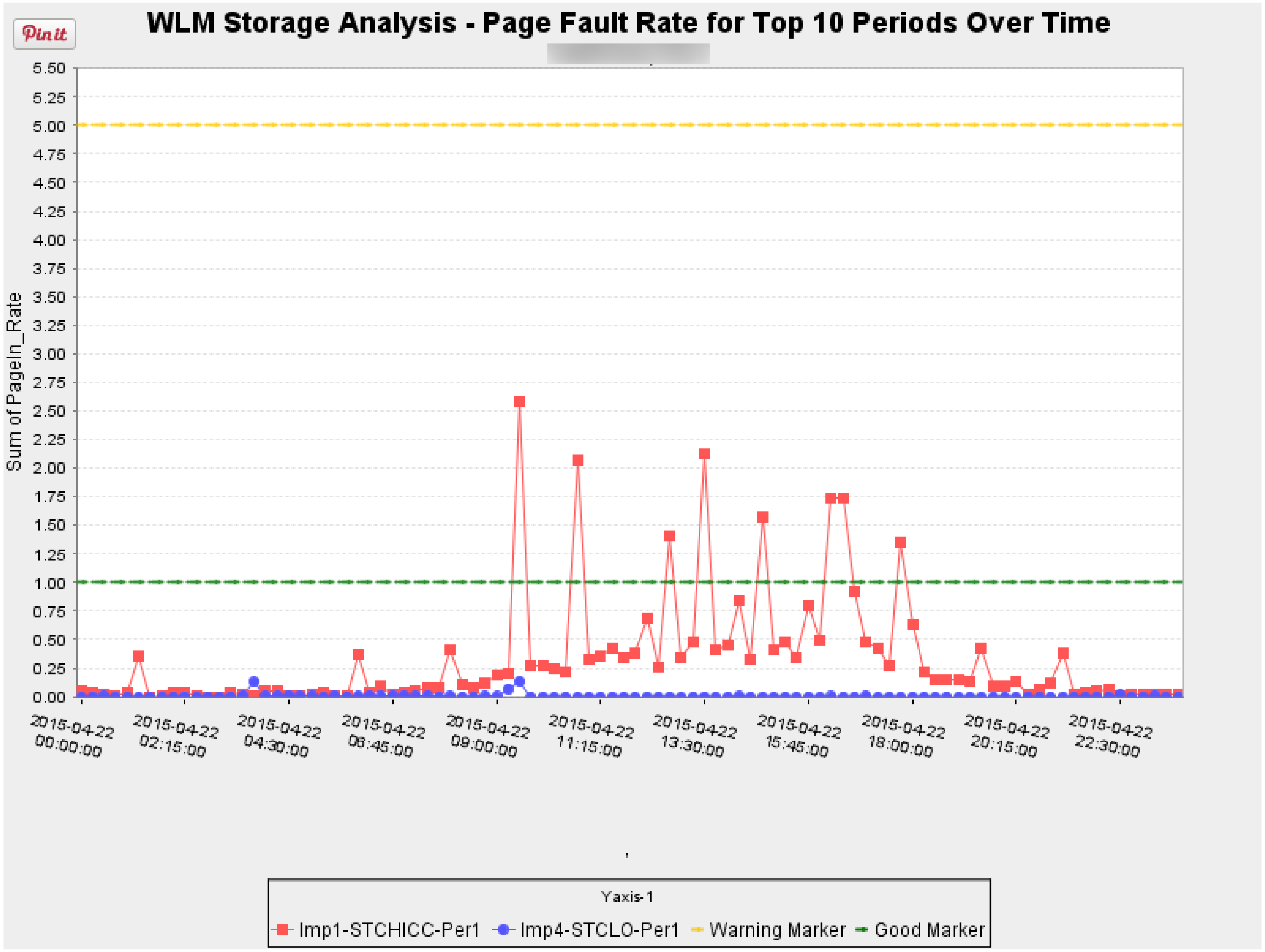
Available storage - Minimum



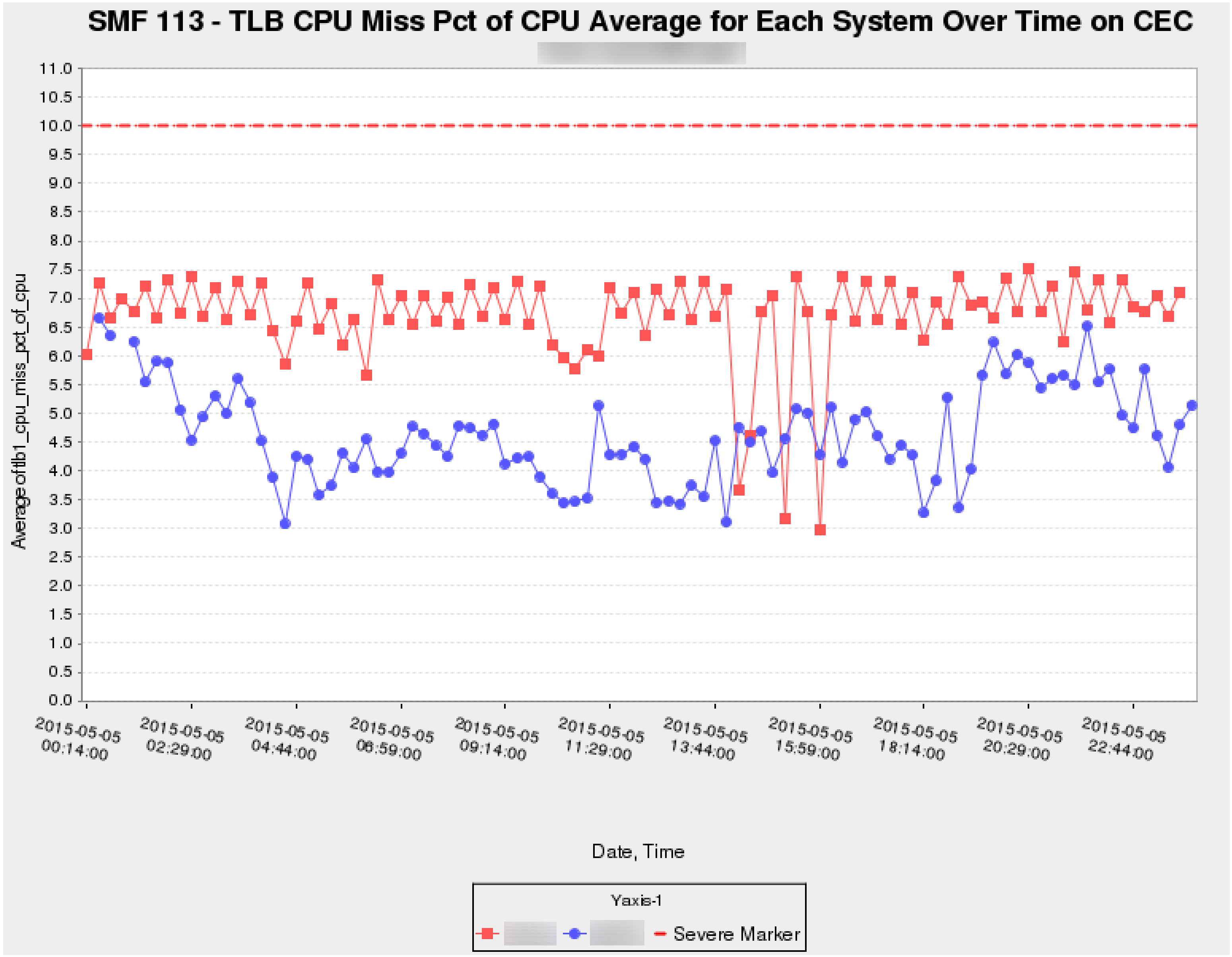
1 MB pages used / available



Paging by workload



TLB overhead



Summary

- Memory is getting cheaper, and is cheaper than CPU, so leverage memory to save CPU
- SCM can reduce cost of paging
  - Some paging may be acceptable for certain workloads
- Consider adjusting old memory limits
- Many memory statistics available, keep an eye on a few key ones as you start using more memory
  - And hopefully IBM will provide us with some more