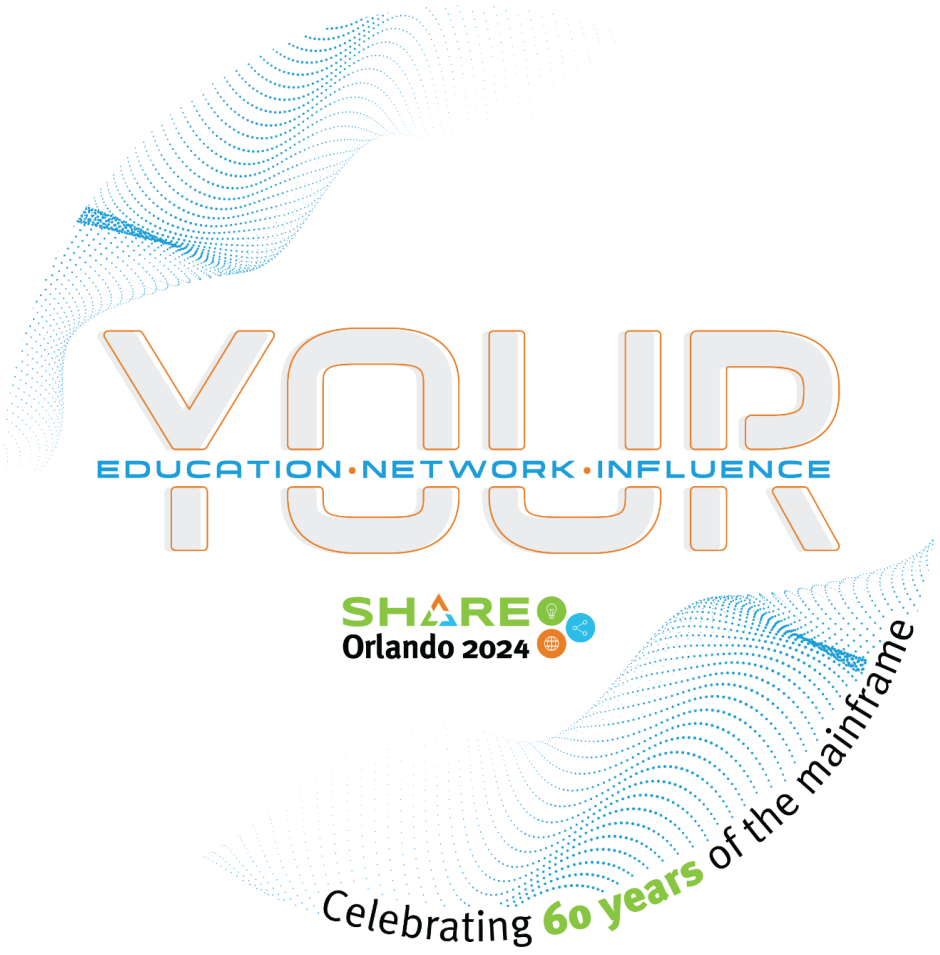


The Highs and Lows: How does HyperDispatch Really Impact CPU Efficiency?

Scott Chapman
Enterprise Performance Strategies, Inc.
Scott.chapman@EPStrategies.com

Contact, Copyright, and Trademarks

Questions?

Send email to performance.questions@EPStrategies.com, or visit our website at https://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

Abstract (why you're here!)

HiperDispatch has been around for a number of years now, but there is still a misunderstanding of the true differentials and effectiveness of logical processors designated as high, medium, and low. In addition, there is the seemingly never ending questions of how HiperDispatch determines the number of high, medium, and low pool processors for an LPAR. A common practice is to optimize LPAR configuration such that the most important LPARs have at least one high pool processor. But how much does this matter in real life? How much benefit can you expect to gain for your most-loved LPARs if you can give them an extra high-pool processor? How much might that hurt other LPARs?

During this session, Scott Chapman will dive deeper into HiperDispatch and help the attendees better understand the true meaning and effectiveness of each pool of processors.

EPS: We do z/OS performance…

- Pivotor - Reporting and analysis software and services
  - Not just reporting, but analysis-based reporting based on our expertise
- Education and instruction
  - We have taught our z/OS performance workshops all over the world
- Consulting
  - Performance war rooms: concentrated, highly productive group discussions and analysis
- Information
  - We present around the world and participate in online forums https://www.pivotor.com/content.html

z/OS Performance workshops available

During these workshops you will be analyzing your own data!

- WLM Performance and Re-evaluating Goals
  - February 19-23, 2024
- Parallel Sysplex and z/OS Performance Tuning
  - August 20-21, 2024
- Essential z/OS Performance Tuning
  - September 16-20, 2024
- Also… please make sure you are signed up for our free monthly z/OS educational webinars! (email contact@epstrategies.com)

Like what you see?

- The z/OS Performance Graphs you see here come from Pivotor

- If you don't see them in your performance reporting tool, or you just want a free cursory performance review of your environment, let us know!
  - We're always happy to process a day's worth of data and show you the results
  - See also: http://pivotor.com/cursoryReview.html

- We also have a free Pivotor offering available as well
  - 1 System, SMF 70-72 only, 7 Day retention
  - That still encompasses over 100 reports!

EPS presentations this week

What	Who	When	Where
CPU Critical: A modern revisit of a classic WLM option	Peter Enrico Scott Chapman	Mon 4:00	Salon 12
30th Anniversary of Parallel Sysplex: A Retrospective and Lessons Learned	Peter Enrico	Tue 10:30	Salon 21
z/OS Performance Spotlight: Some Top Things You May Not Know	Peter Enrico Scott Chapman	Tue 1:00	Salon 15
The Highs and Lows: How Does HyperDispatch Really Impact CPU Efficiency?	Scott Chapman	Thu 10:30	Salon 21
Configuring LPARs to Optimize Performance	Scott Chapman	Thu 2:30	Salon 21

Agenda

- Brief overview of HiperDispatch
- Medium pool rules
- Common HiperDispatch expectations & measurements
- What do we see in real life measurements?
- Warning Track Interrupts
- Conclusion: how much should you worry about this?

HiperDispatch Overview

HiperDispatch History

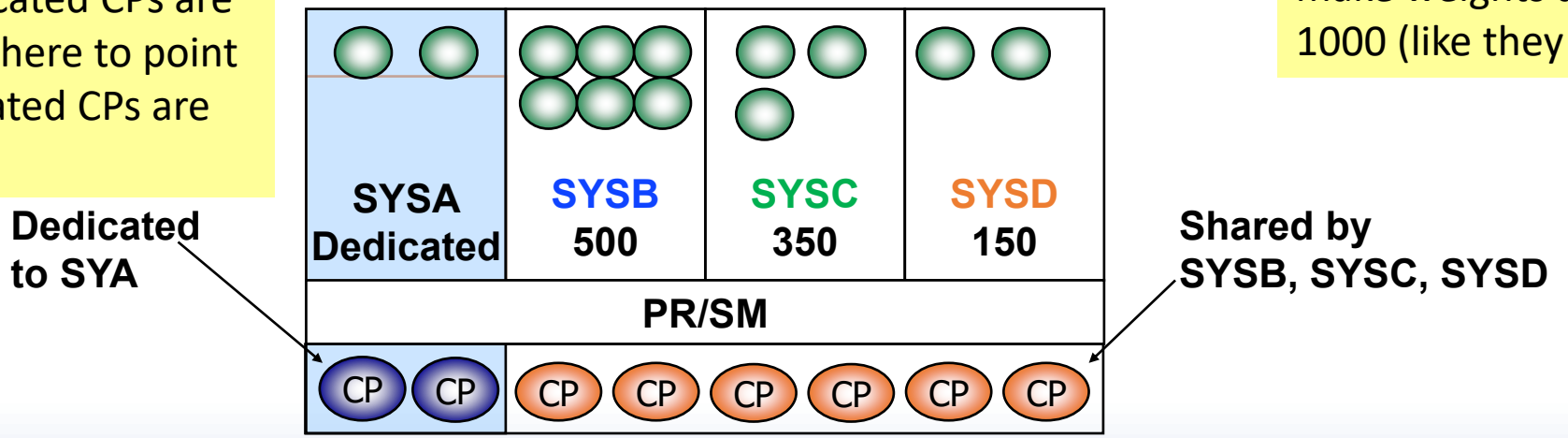
- HiperDispatch was introduced on the z10 in 2008
- Goal was to improve performance through improved cache coherency
  - Basically: don't needlessly split work across lots of CPs if you can keep like work on a smaller number of CPs
  - Mitigates the "short CP" problem
    - Caused by having high ratio of logical to physical CPs
- Changed both PR/SM and z/OS dispatching
- Was originally optional, but default and expectation is now "On"
  - Required in some configurations and if using SMT

Guaranteed Share as Processors

- Each LPAR's share can be translated into a number of processors
  - $LPAR\ Guaranteed\ Processors = LPAR\ Share\ * Shared\ Processor\ Count$
- In below example, there are 6 shared processors so:
  - SYSB = 500/1000 * 6 = 3 processors
  - SYSC = 350/1000 * 6 = 2.1 processors
  - SYSD = 150/1000 * 6 = 0.9 processors

LPARs with dedicated CPs are rare, but shown here to point out those dedicated CPs are separate

For ease of use, try to make weights add up to 1000 (like they do here).

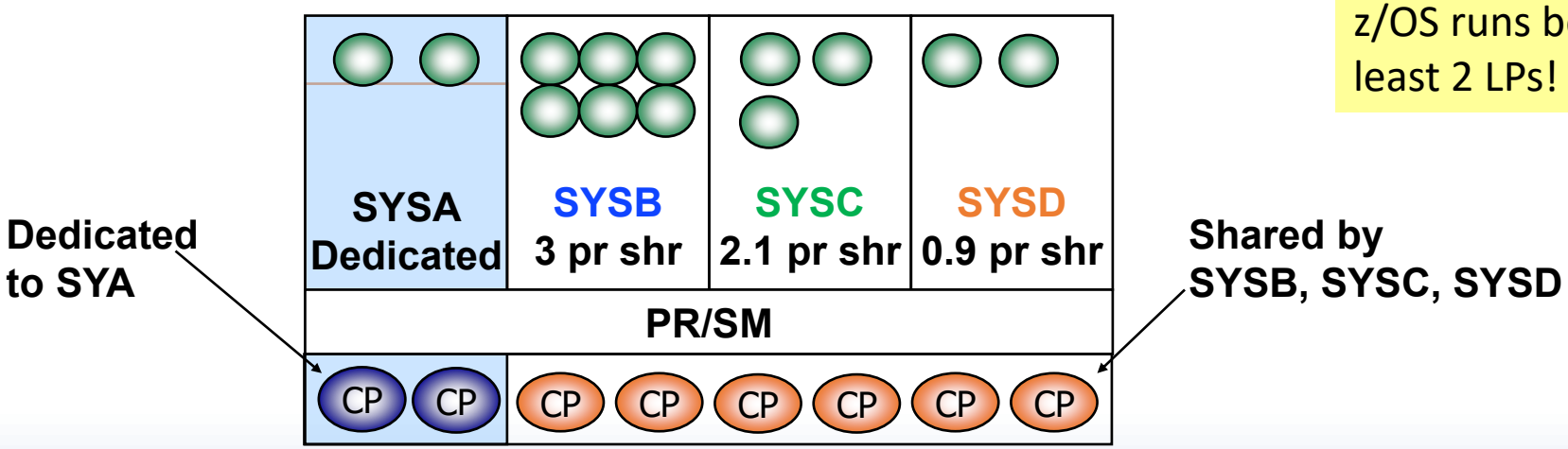


Horizontal CP Management

- Prior to HiperDispatch, PR/SM would split each logical CPU evenly based on its average share of a processor
  - SYSB gets 6 LPs, each effectively 50% of a physical (3 / 6)
  - SYSC gets 3 LPs, each effectively 70% of a physical (2.1 / 3)
  - SYSD gets 2 LPs, each effectively 45% of a physical (0.9 / 2)

Can lead to what's called "short CPs": note SYSB's logicals spend less time dispatched to a physical than SYSC's!

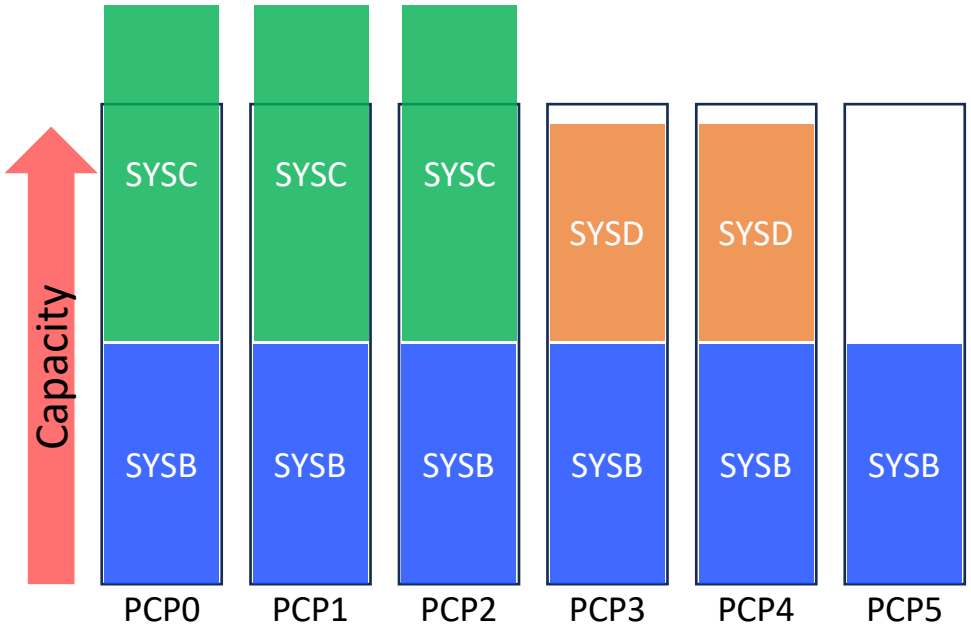
z/OS runs better with at least 2 LPs!

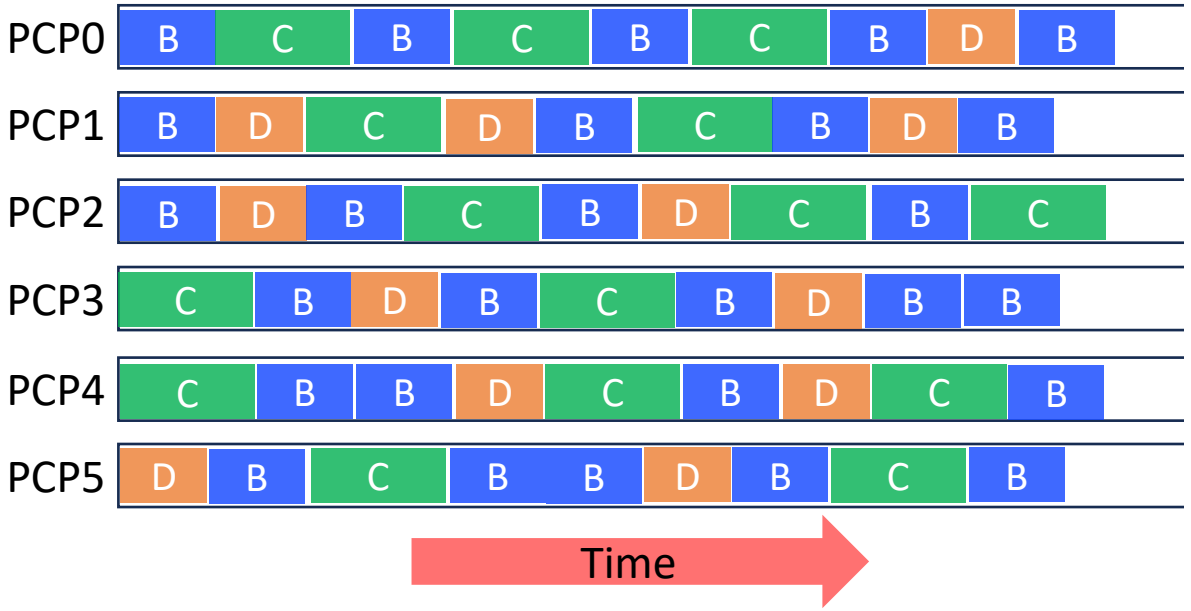


Vertical CP Management

- HiperDispatch manages CPs “vertically”, meaning it endeavors to make the logical CPs a larger percentage of a physical
- Logical processors classified as:
  - High – The processor is quasi-dedicated to the LPAR (100% share) (VH)
  - Medium – Share between 0% and 100% (VM)
  - Low – Unneeded to satisfy LPAR’s weight (VL)
- This processor classification is sometimes referred to as “vertical” or “polarity” or “pool”
  - E.G. Vertical High = VH = High Polarity = High Pool = HP
- Parked / Unparked
  - Initially, VL processors are “parked”: work is not dispatched to them
  - VL processors may become unparked (eligible for work) if there is demand and available capacity

Physical to Logical: Horizontal Mgt

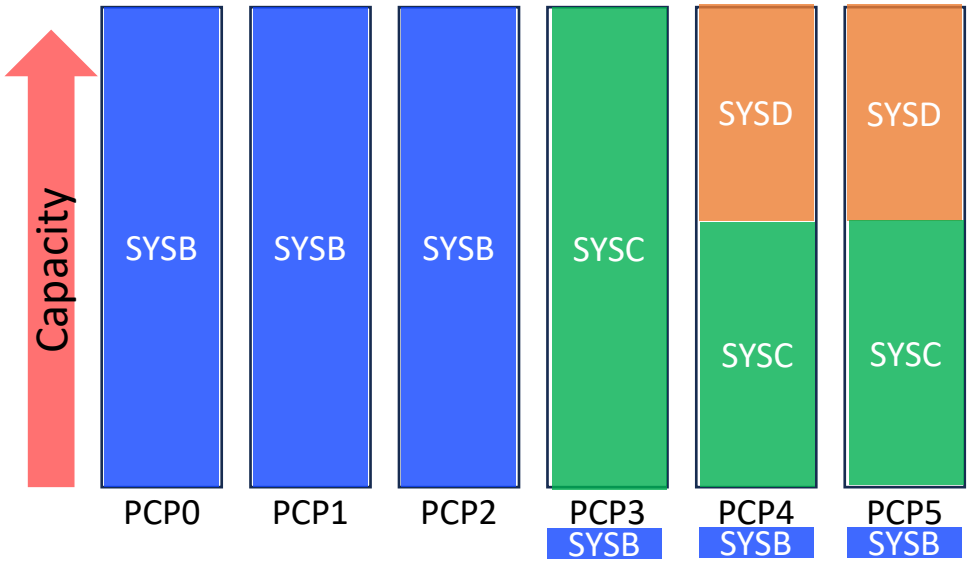




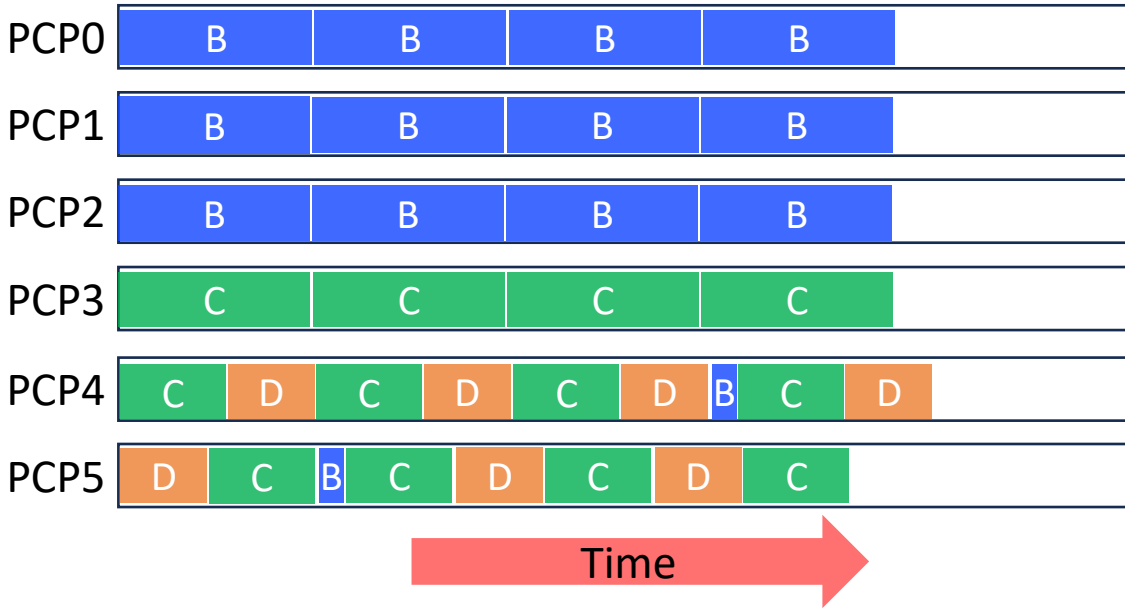
Note that capacity for the LPAR is spread horizontally across all the LCPs. Although in reality, PR/SM wouldn't (couldn't!) statically allocate logicals to physicals like this.

PR/SM time slices the physical processors dispatching logical to physicals for specific durations (or until the LPAR gives back the processor). With Horizontal CPU management, a logical CP might get redispatched to any physical CP.

Physical to Logical: Vertical Mgt



With vertical CPU management, this is closer to reality. Note that SYSB's VLs will only come into play when there's both demand from SYSB and the others' aren't using the capacity.

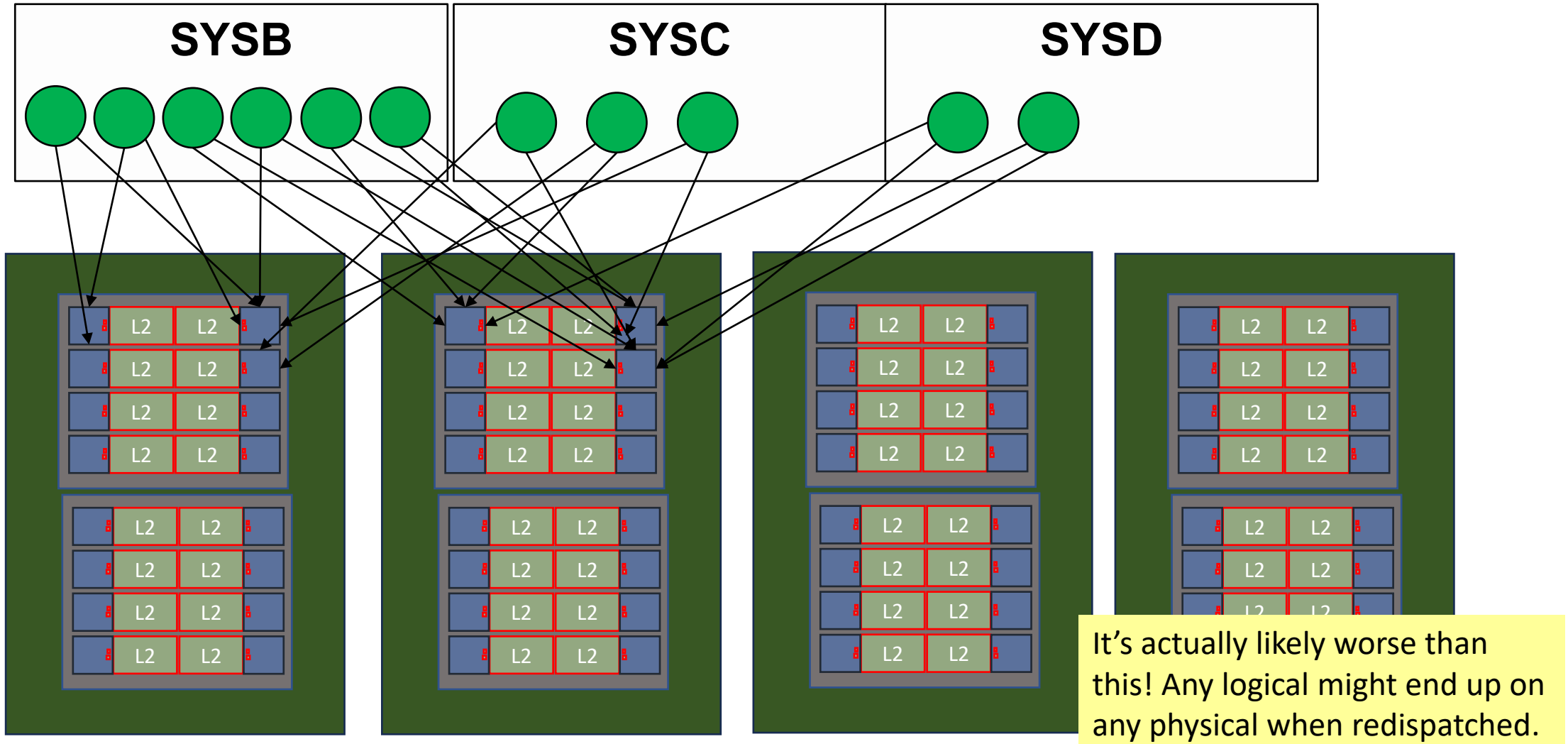


Note that while reality may be a bit messier, vertical CPU management does greatly reduce the movement of logicals to different physicals. Also note VH CPs get longer dispatch intervals.

A brief aside…

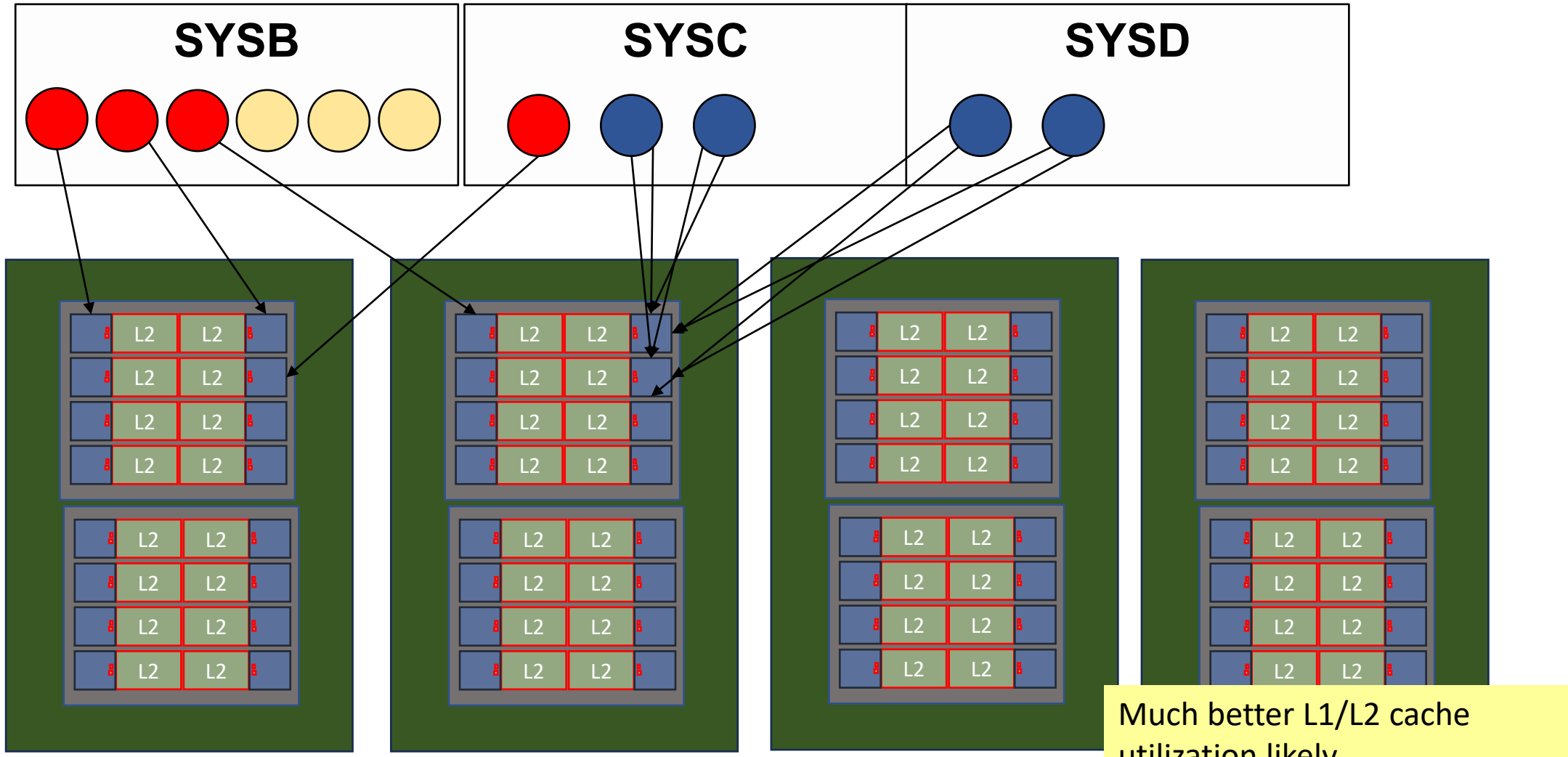
- The previous charts should be taken as being useful models
  - But all models are simplifications of reality!
- In reality, PR/SM's job is a bit more complicated
  - LPARs will give up their CPs when there's not work to do
  - VH CPs are not actually dedicated to their LPARs
    - But do get 100ms time slices vs 12.5-25ms for VM/VLs
    - And other LPARs are only going to get dispatched there if the VHs are not busy
  - PR/SM will issue warning track interrupts to let LPARs know their time slice is about to end
    - LPARs will usually yield back some time to avoid that stranding a logical without a physical
    - PR/SM then tries to make the LPARs "whole" over time
- But note that it is true that an LPAR may not have all (or even any!) of it's logicals dispatched to a physical at any one moment in time

HiperDispatch Off



It's actually likely worse than this! Any logical might end up on any physical when redispatched.

HiperDispatch On



Much better L1/L2 cache utilization likely.

Medium Pool Rules

Medium Pool Processors

- HiperDispatch prefers not to have VMs with low weight
  - Instead a VH will be taken as a second VM and the two VMs sharing the weight of those two engines
- E.G. an LPAR with weight giving it access to 2.4 CP's worth of capacity:
  - ~~2 VH (100% each) + 1 VM (40%)~~ <- PR/SM will not do this
  - 1 VH (100%) + 2 VM (70% each) <- PR/SM will do this
- Basically PR/SM wants a single medium pool CP to get at least a 50% share of a physical CP
  - If the weight of an LPAR is just under n.5 CPs of capacity getting it to n.5 should result in an extra VH

Potential VM Confusion

- z13 has different rules for when the weight is between 1.5 and 2.0 CPs
  - Instead of 1 VH and 1 VM, gets 2 VMs
- Can only have a VM if it's weight would be at least 0.5 CPs
  - Otherwise a VH is demoted and the combined weight is divided between the two VMs
  - E.G. and LPAR with a weight of 2.1 CPs would have 1 VH, and 2 VMs at 0.55 each
- So if there's 2 VMs, they will always have a weight between 0.5 and 0.75
  - I.E. ((1 + .01) / 2) <= x <= ((1 + .49) / 2)
  - Except the z13 scenario above, where both will be > 0.75
- But if the VM would have had a weight > 0.5 it can stand on its own
  - And such a solo VM could have a weight approaching 1

HiperDispatch Expectations

High-pool love, Low-pool hate

- Common belief / expectation:
  - VH processors perform better 😻
  - VL processors perform worse 😠
  - HiperDispatch is geared towards machines with many processors
- It is common to hear recommendations to tweak LPAR weights to get an extra VH processor for a loved LPAR
- Also common is the recommendation to not use low-pool processors
  - IBM recommendation to not have more than 2 VL processors
  - Note we're only talking about z/OS running under PR/SM in this presentation: impacts to z/VM and z/Linux may be different

How can we measure efficiency?

- Commonly cited:
  - CPI – Cycles Per Instruction – lower is better
    - Can be broken down into
      - Instruction Complexity CPI – CPI influenced by the instruction mix
      - Finite Cache CPI – CPI influenced by cache contention (because caches are finite)
  - RNI – Relative Nest Intensity – lower is better
    - Calculates a number that is workload-related and should remain somewhat stable when moving between processor generations
    - Can be useful for showing the relative impact of cache misses at each level
- More directly: if you make a change and the CPU consumption for the workload goes down, that was a good change
  - Note you can't take single measurements though—you have to look over multiple executions to account for normal variations and cross-workload contentions

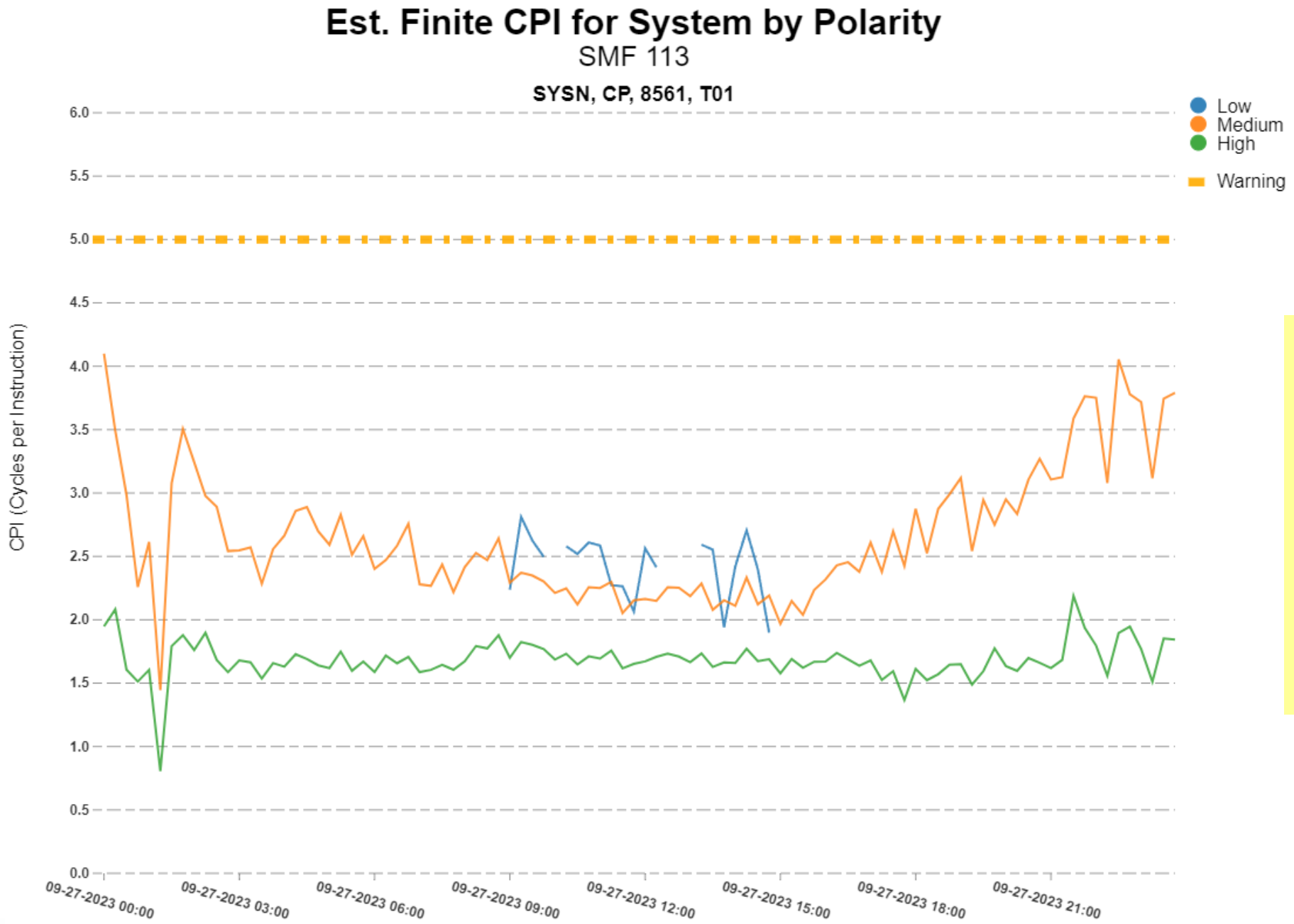
Can we justify the love/hate?

- Probably the easiest way to show this is to look at the Estimated Finite CPI for each processor, with the expectation:
  - VH will show lower Est Finite CPI
  - VL will show higher Est Finite CPI
  - VM will be in the middle

- But do we see this?

Maybe
Sometimes
It Depends

Real life measurements

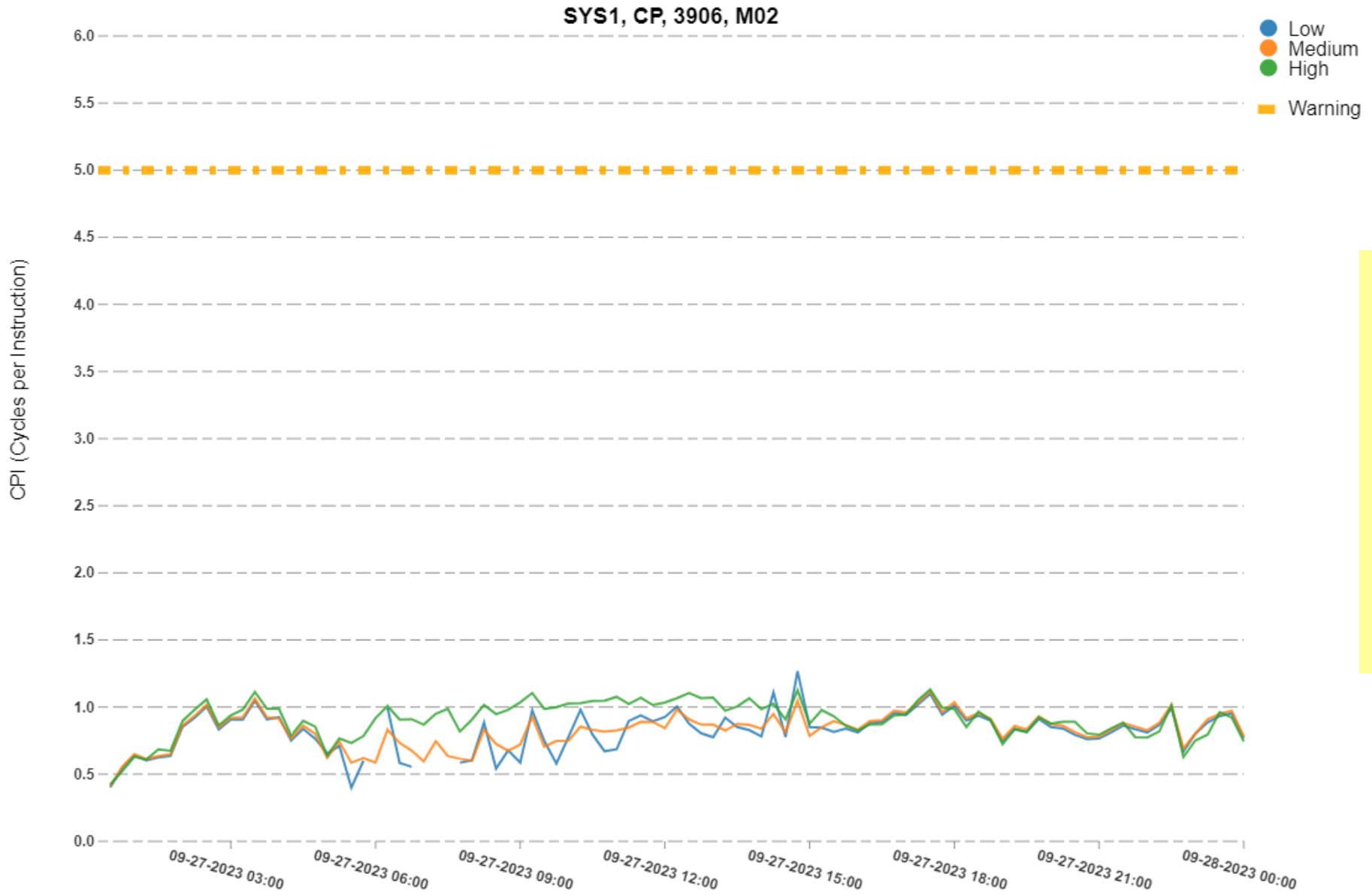


So the assumption is that if we looked at estimated finite CPI by engine polarity, we'd see patterns that usually looked like this.

Reality is rarely this nice!

Est. Finite CPI for System by Polarity

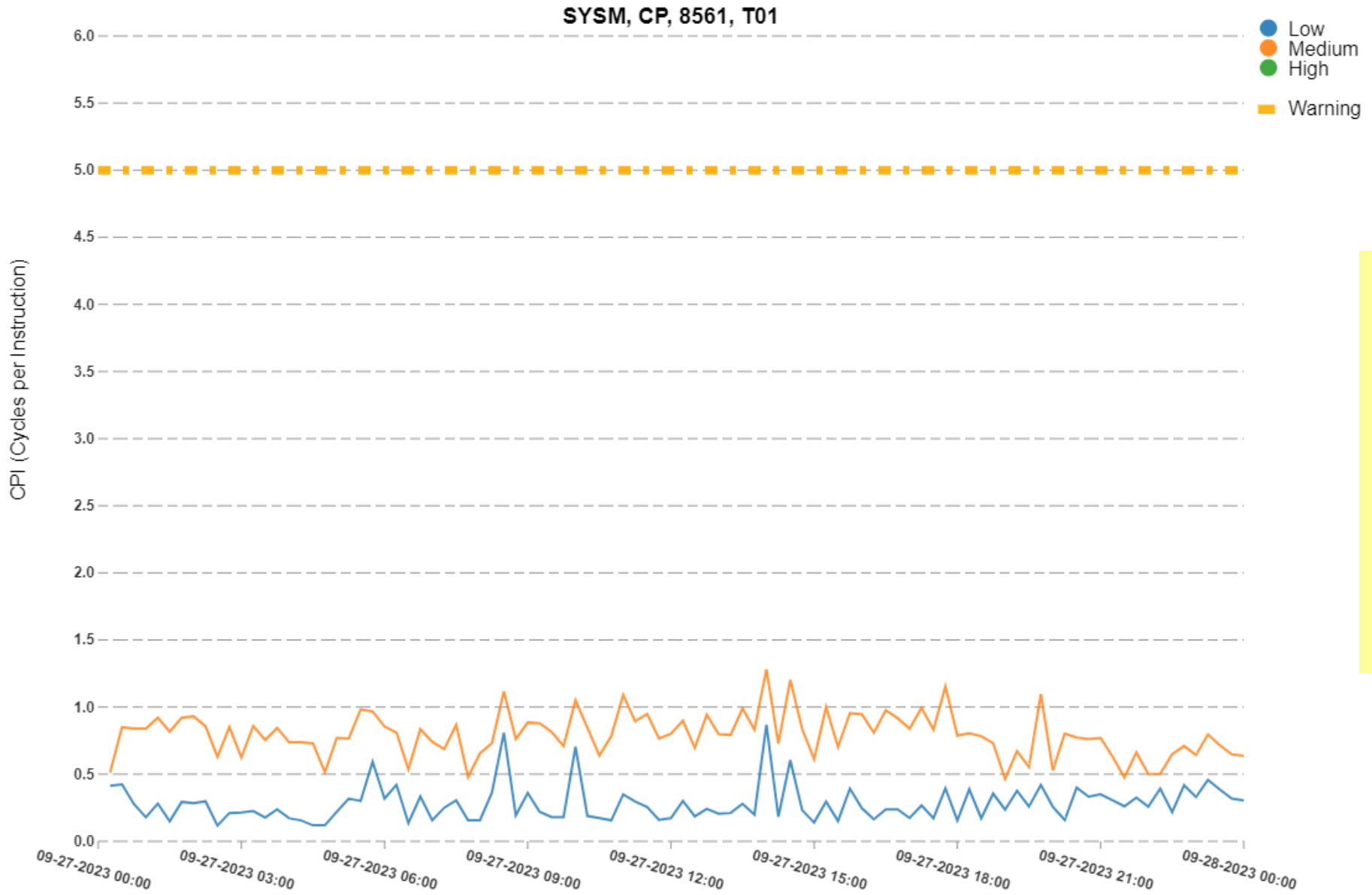
SMF 113



Instead we see systems like this where there is either no difference or maybe even the high pool processor shows running worse than the medium/low!

Est. Finite CPI for System by Polarity

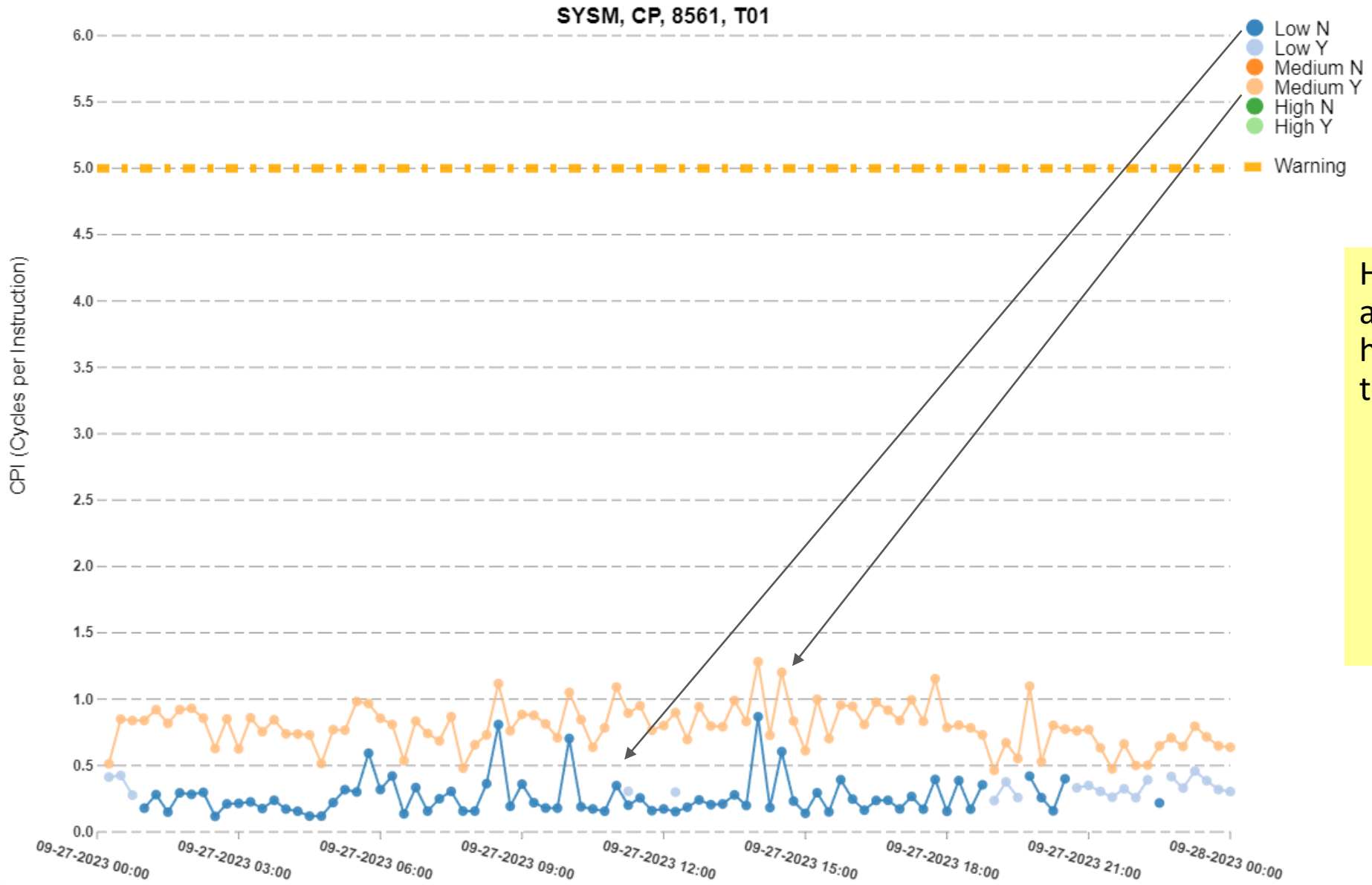
SMF 113



Or how about this, where there is no high, but the low(s) always seem to be more efficient than the medium(s)??

Est. Finite CPI for System by Polarity and I/O Interrupts

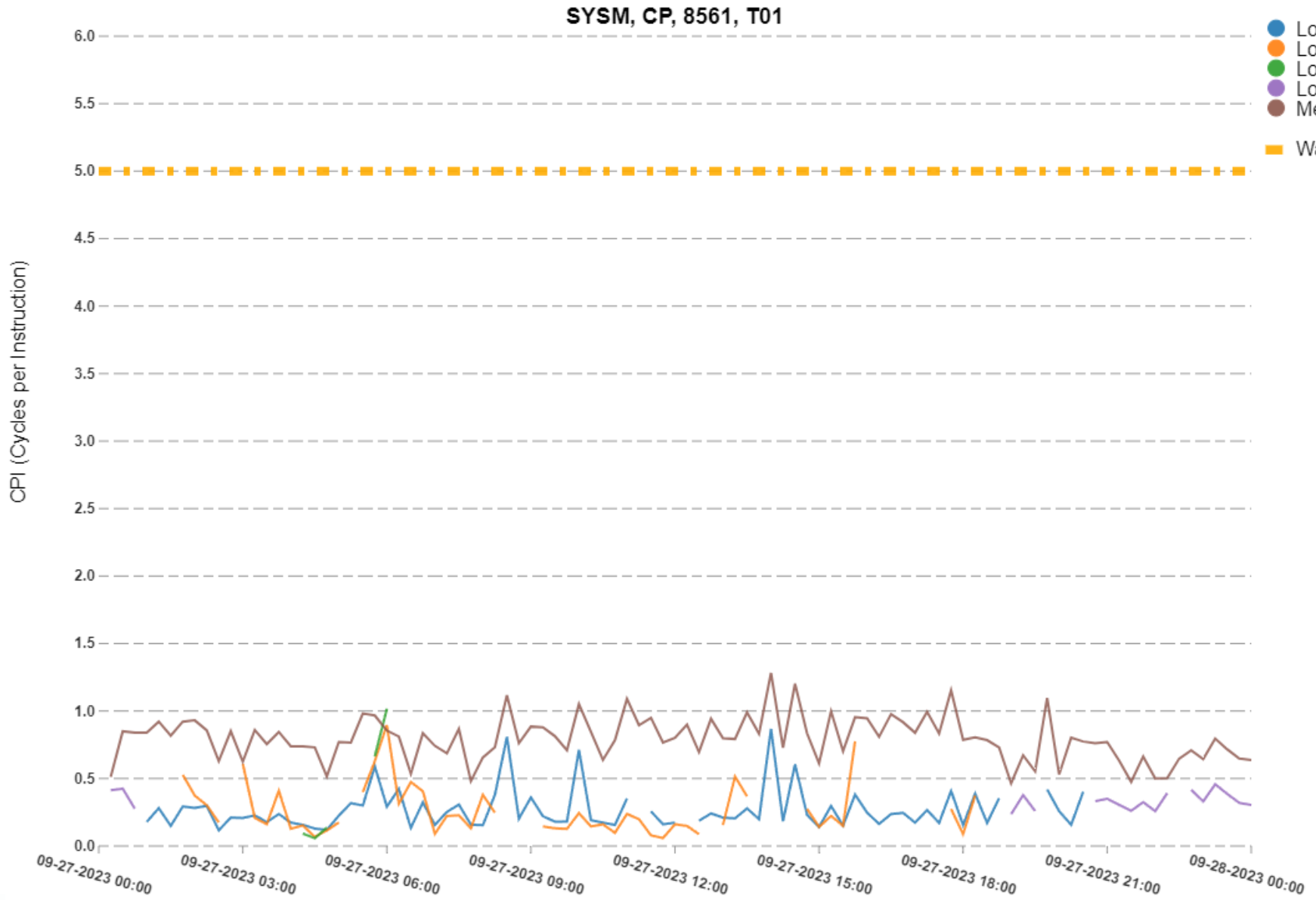
SMF 113



Here's part of the answer: CPs which handle I/O interrupts tend to be less efficient.

Est. Finite CPI by Specific CPU

With Polarity and I/O Interrupt



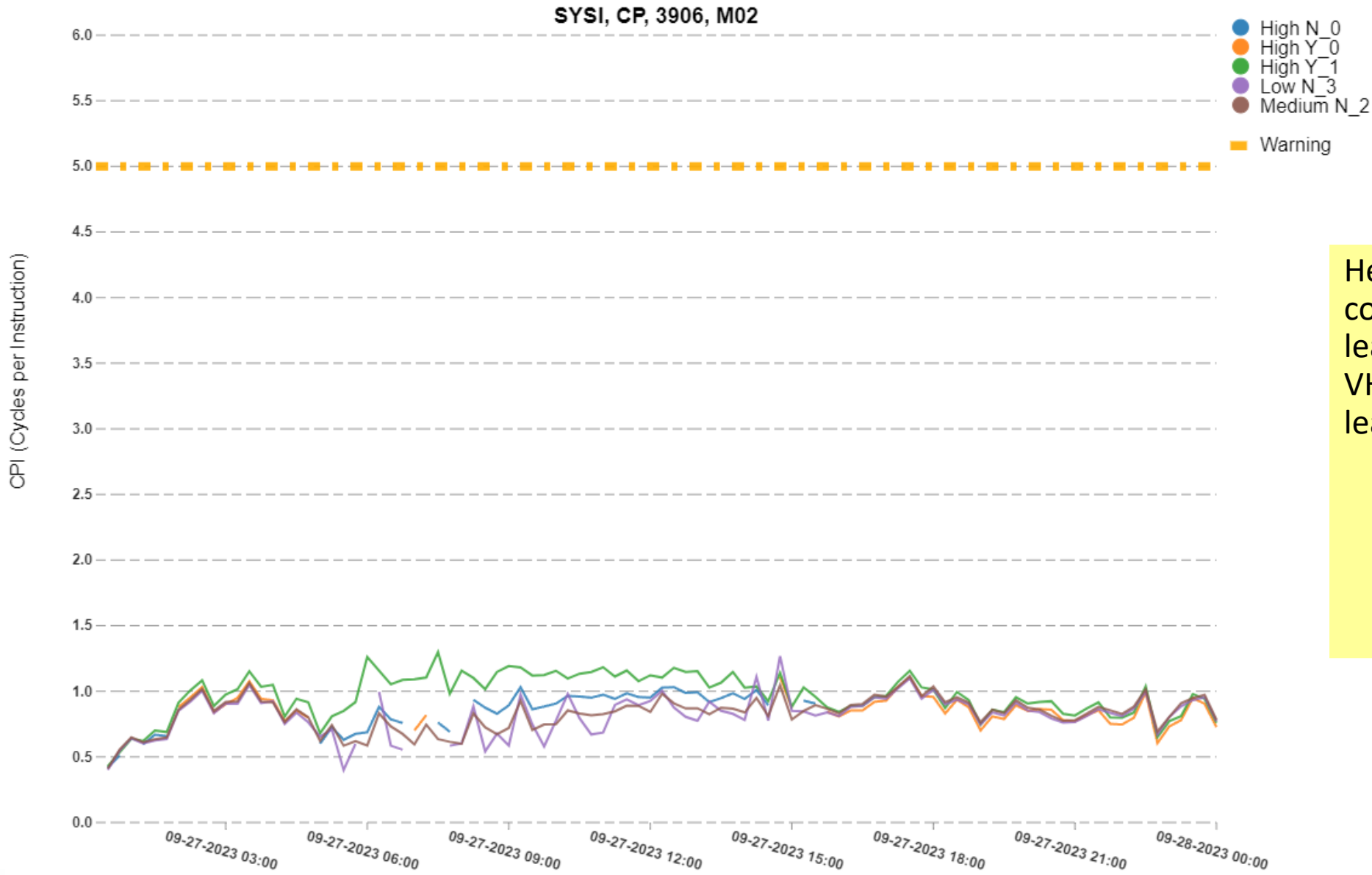
In this case, the LPAR only has 1 VM and 3 VL, so one VL will always be unparked.

The VM is usually the only CP enabled for interrupts, and it so it runs less efficiently.

Also note in this case even when the VL did handle interrupts, it didn't handle many.

Est. Finite CPI by Specific CPU

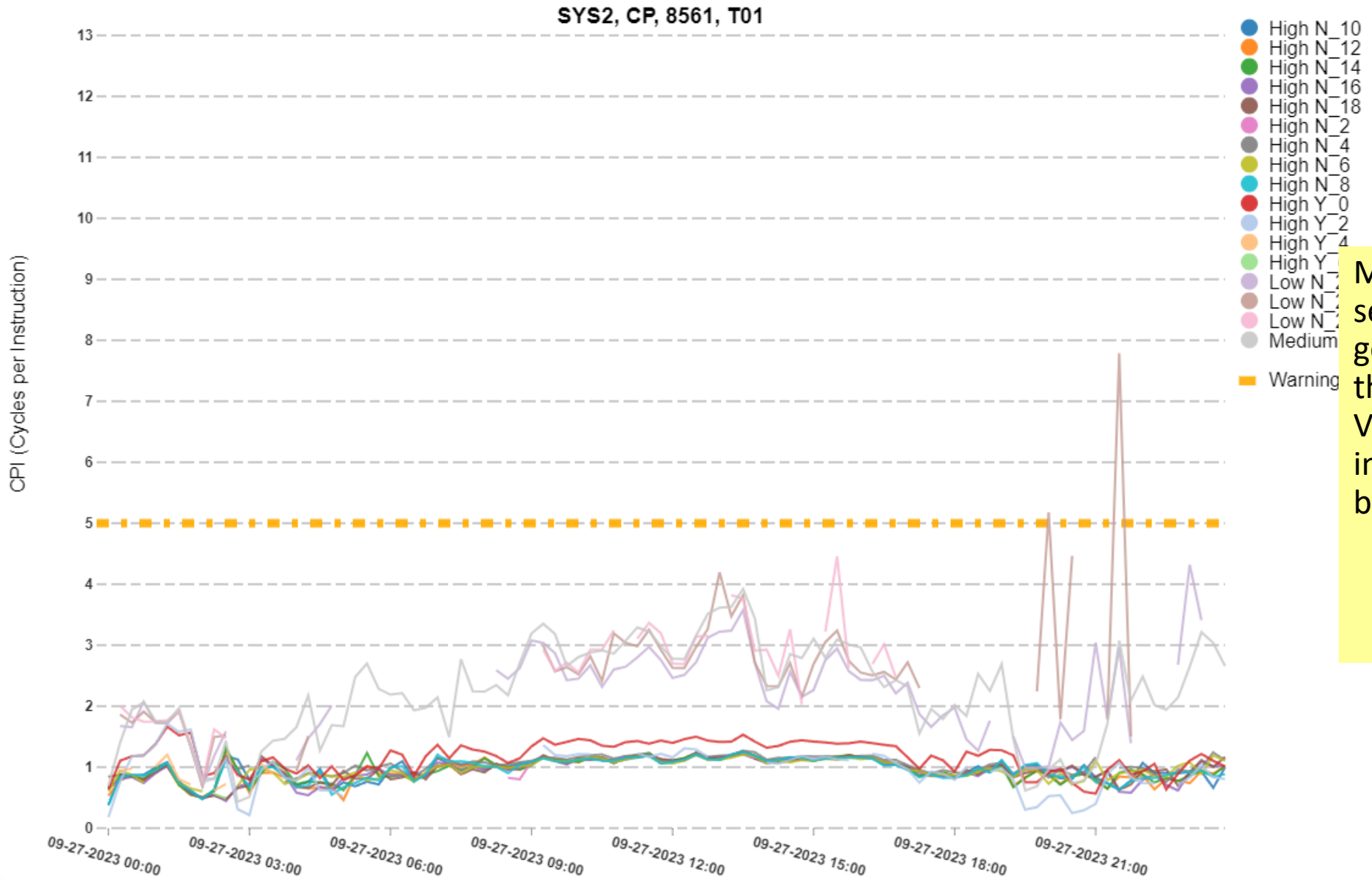
With Polarity and I/O Interrupt



Here's a system with a couple of VH and for at least part of the day, the VH that does I/O is the least efficient CP.

Est. Finite CPI by Specific CPU

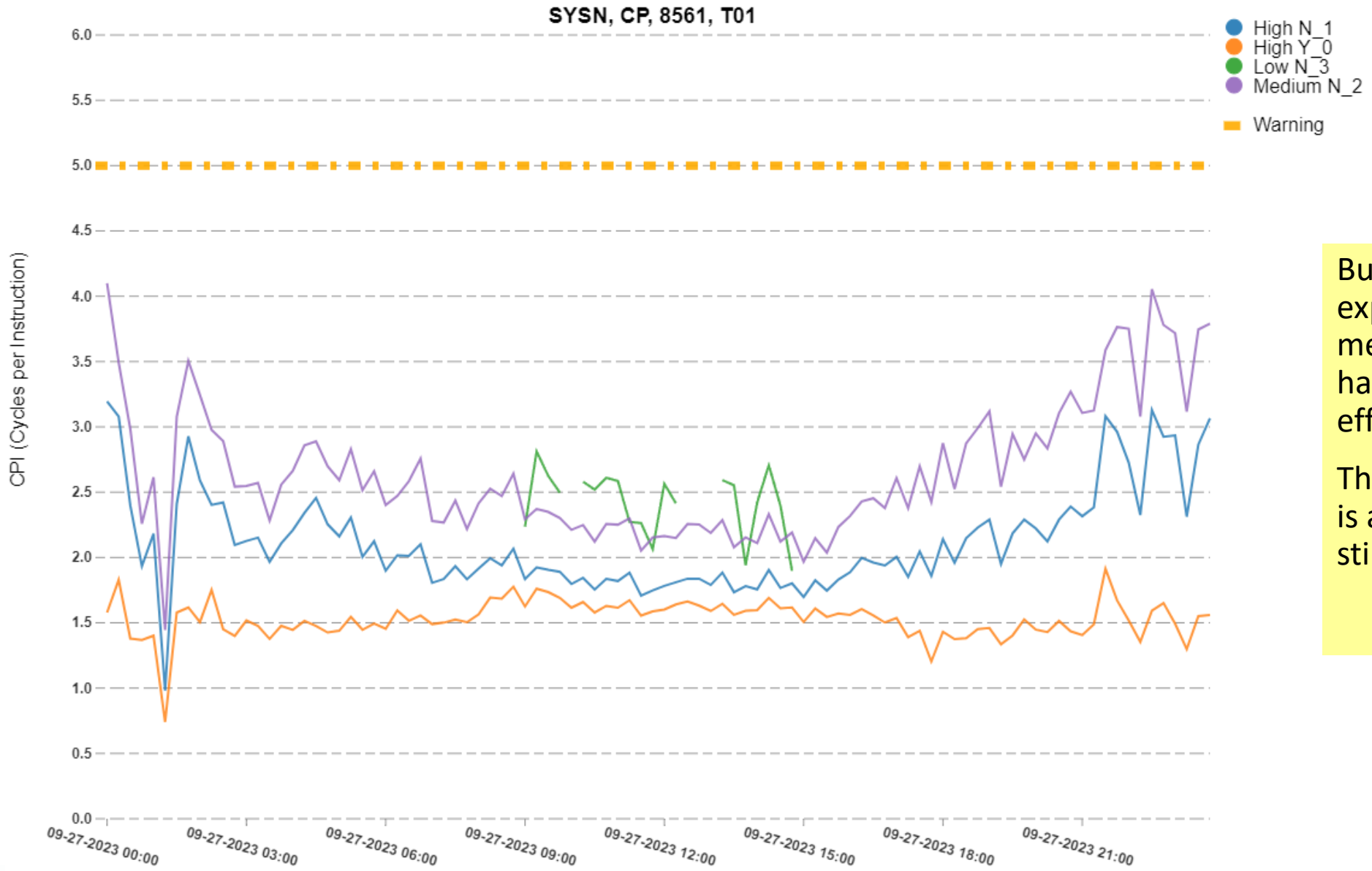
With Polarity and I/O Interrupt



Much larger system with several VH, all of which generally outperform the VM and VLs. And the VH handling the I/O interrupts is generally a bit less efficient.

Est. Finite CPI by Specific CPU

With Polarity and I/O Interrupt

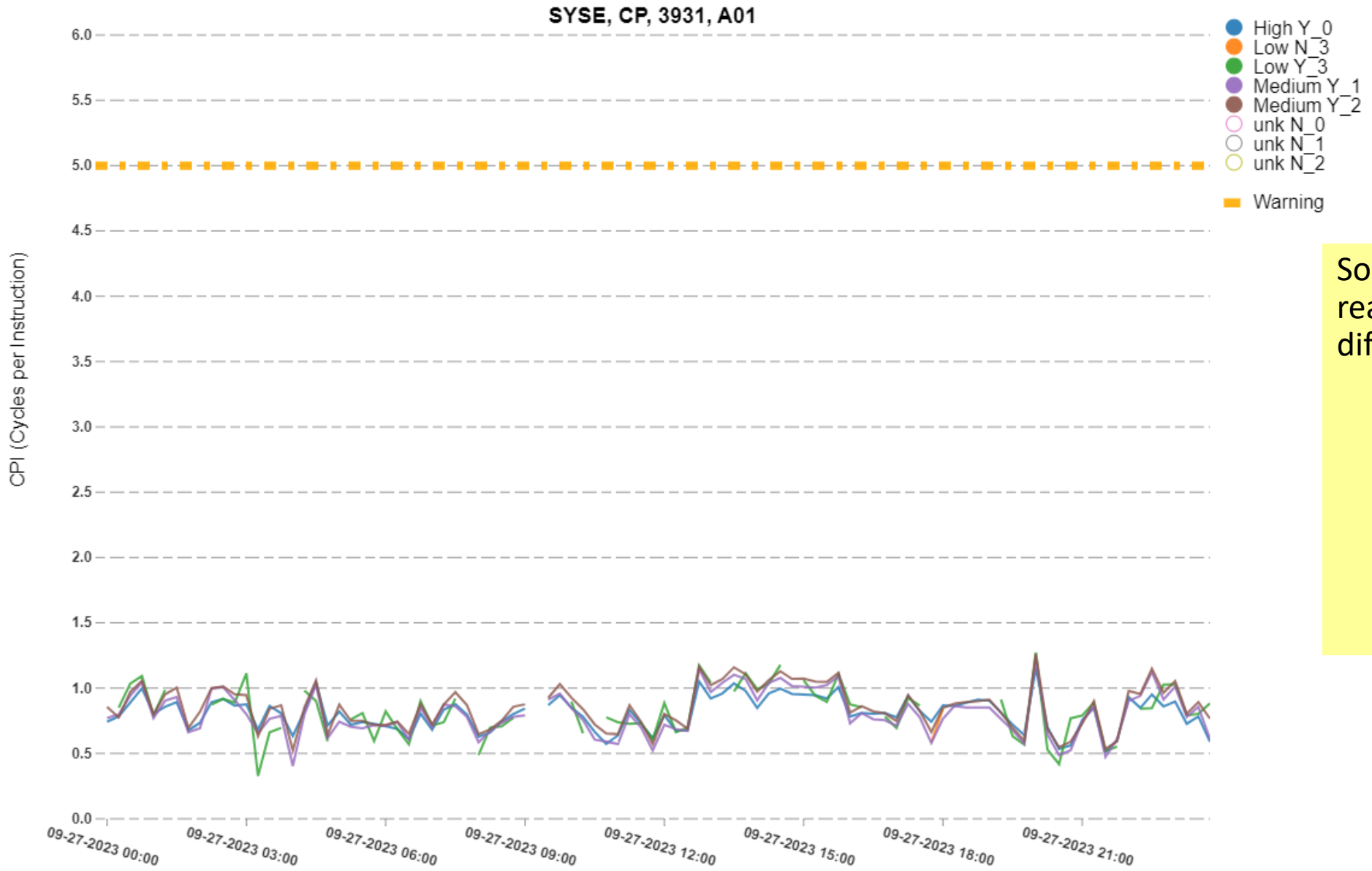


But sometimes our expectations are not met. Here the VH handling I/O is the more efficient.

This may be because this is a less busy system, but still does significant I/O.

Est. Finite CPI by Specific CPU

With Polarity and I/O Interrupt



Sometimes there just really isn't that much difference at all!

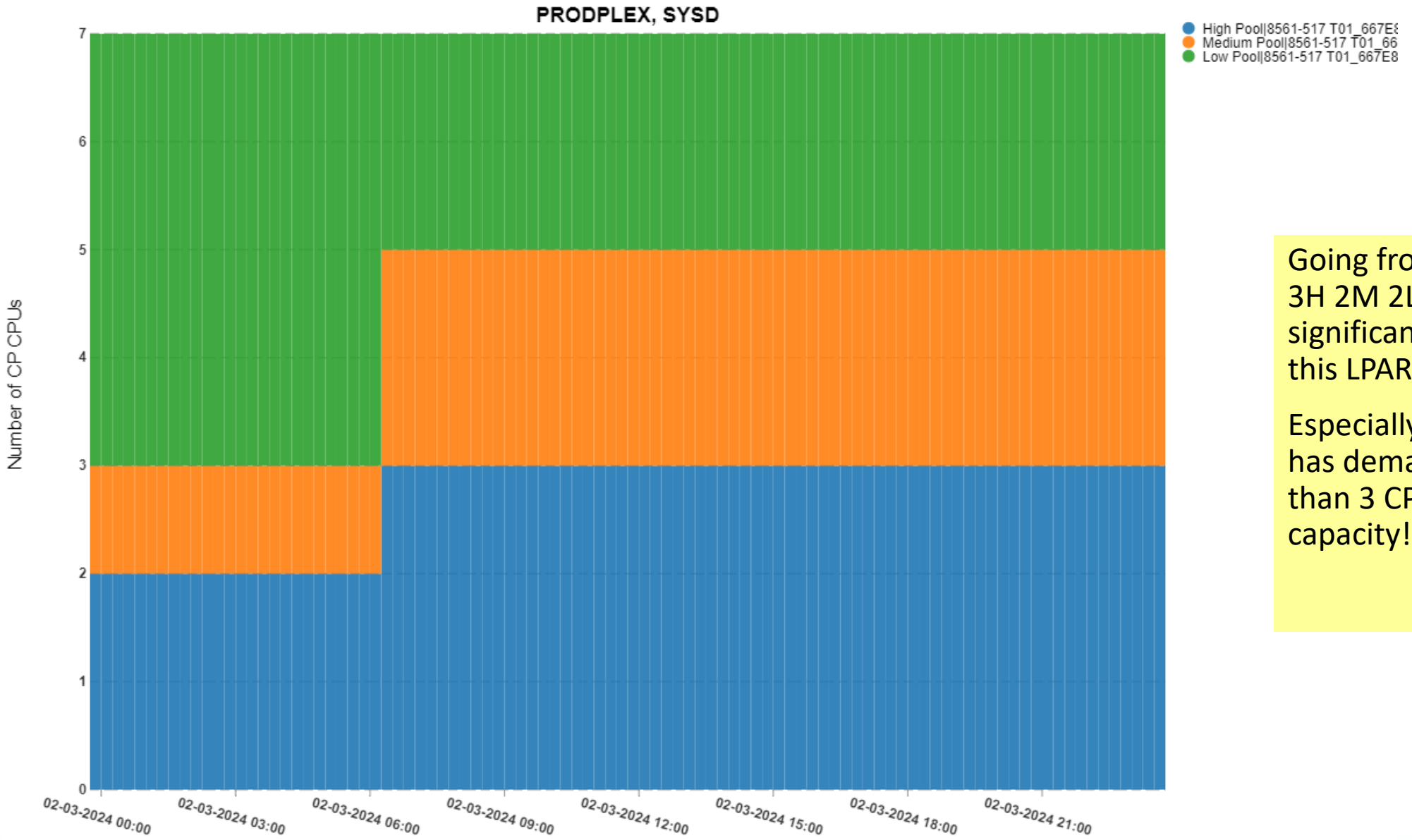
So should we try to convert VM to VH?

- The situation under consideration is generally:
  x VH & 2 VM → x+1 VH & 1 VM
  - The idea being that if we add a little weight to the LPAR it will convert a VM to VH
- This means you have to take weight from another LPAR(s)
  - This could be an issue if those LPARs need that weight!
- The remaining VM will have less weight than the two VMs did before
  - So the new VH *may* be more efficient, but the remaining VM *may* be less efficient
- If the LPAR already has lots of VHs, one more is probably not a big deal
- If the LPAR has 0 (or 1) VHs, the new VH will likely be handling I/O interrupts, meaning it might not be substantially more efficient
  - But… you might see less I/O dispatch delay

Note on weights

- Make sure the LPAR has sufficient weight for its work
  - This can be more important than tweaking to convert a VH to VM!
  - E.G. this may involve multiple VH/VM processors not just one
- Can be done via automation and REXX code (since z/OS 2.1!)
  - In SYS1.SAMPLIB see HWIXMRJL and HWIXMRS1, HWIXMRS2
  - https://www.ibm.com/docs/en/zos/2.5.0?topic=bcpii-setup-installation
- May want to change twice a day if overnight needs are different
- Could change weights based on expected loads and day of week
  - E.G. if Fridays are generally less busy in production, maybe give dev/test more?
- Note too that there's Intelligent Resource Director
  - But only works within a sysplex, and can be slow to react: you may be smarter!

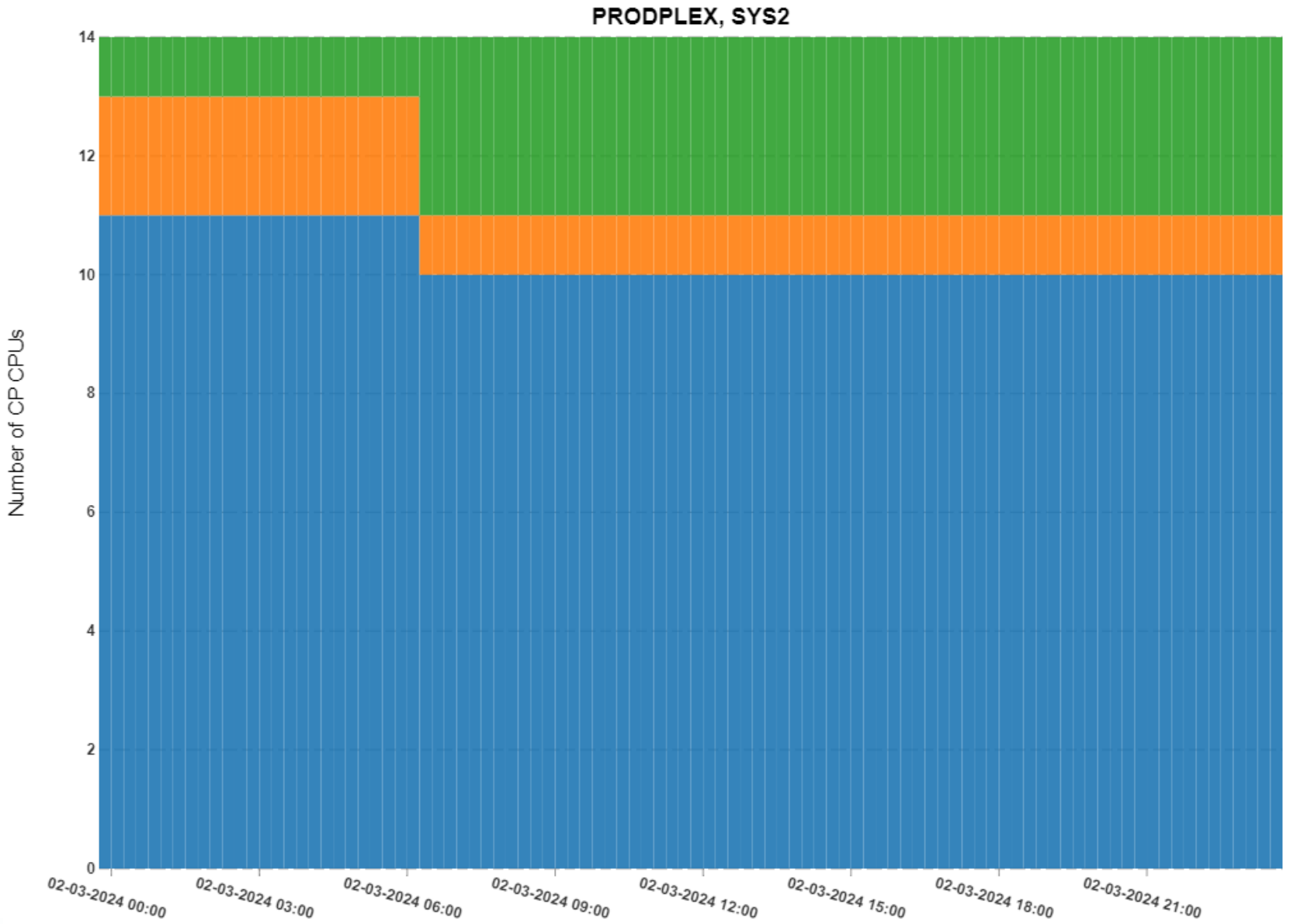
HiperDispatch CP CPU Pooling at End of Interval



Going from 2H 1M 4L to 3H 2M 2L is a pretty significant change for this LPAR!

Especially if it regularly has demand for more than 3 CPs worth of capacity!

HiperDispatch CP CPU Pooling at End of Interval



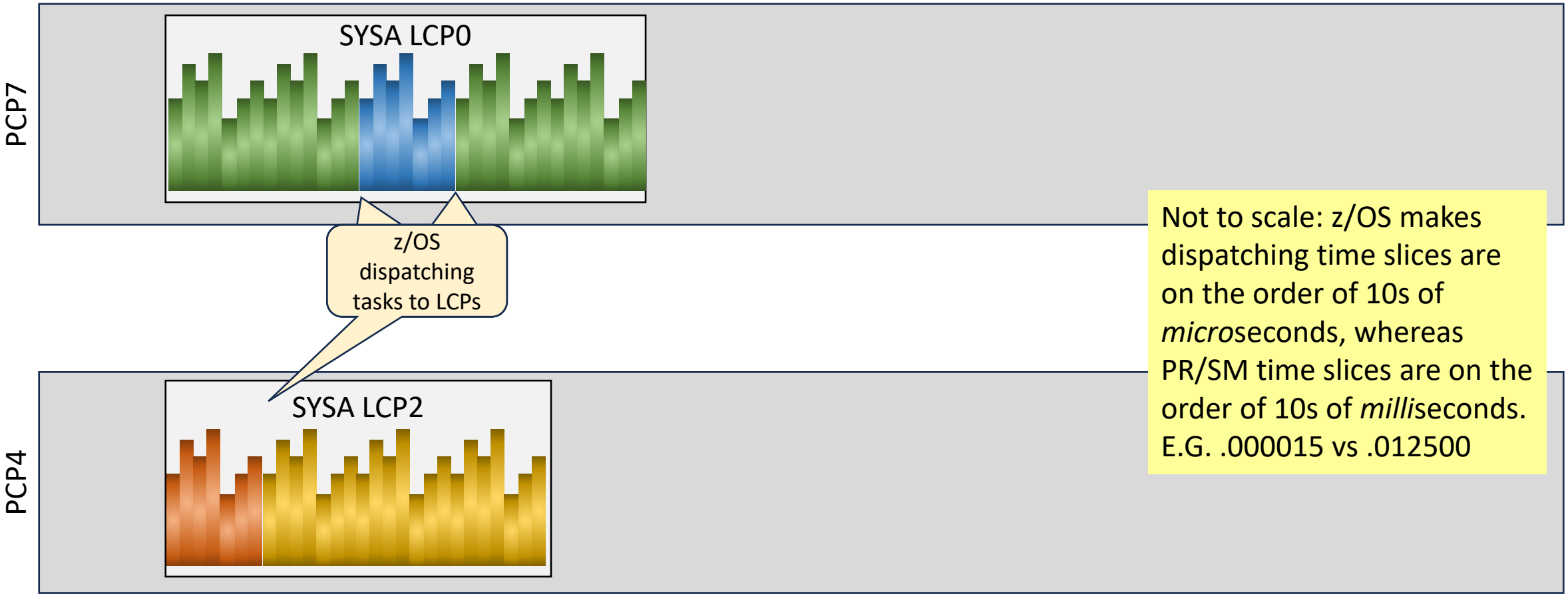
Moving weights is a zero-sum game: there has to be a loser to support the winner.

In this case SYS2 lost 1 of its 11 VHs and 1 of 2 VMs. This certainly could have a negative impact on SYS2, but the loss is relatively minor compared to the relative gain on SYSD.

Whether this was a good change depends on the work running on those systems.

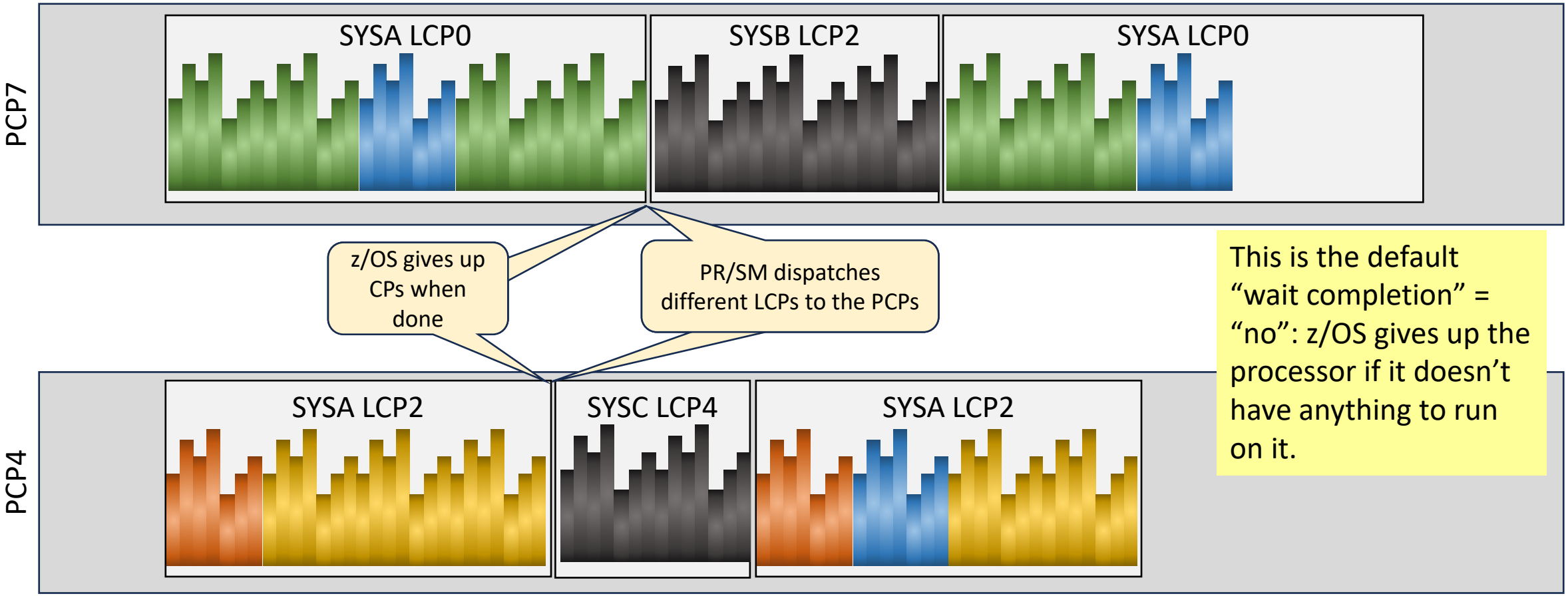
More Vertical Medium & Low Issues

PR/SM Dispatching LCPs



Not to scale: z/OS makes dispatching time slices are on the order of 10s of *micro*seconds, whereas PR/SM time slices are on the order of 10s of *milli*seconds. E.G. .000015 vs .012500

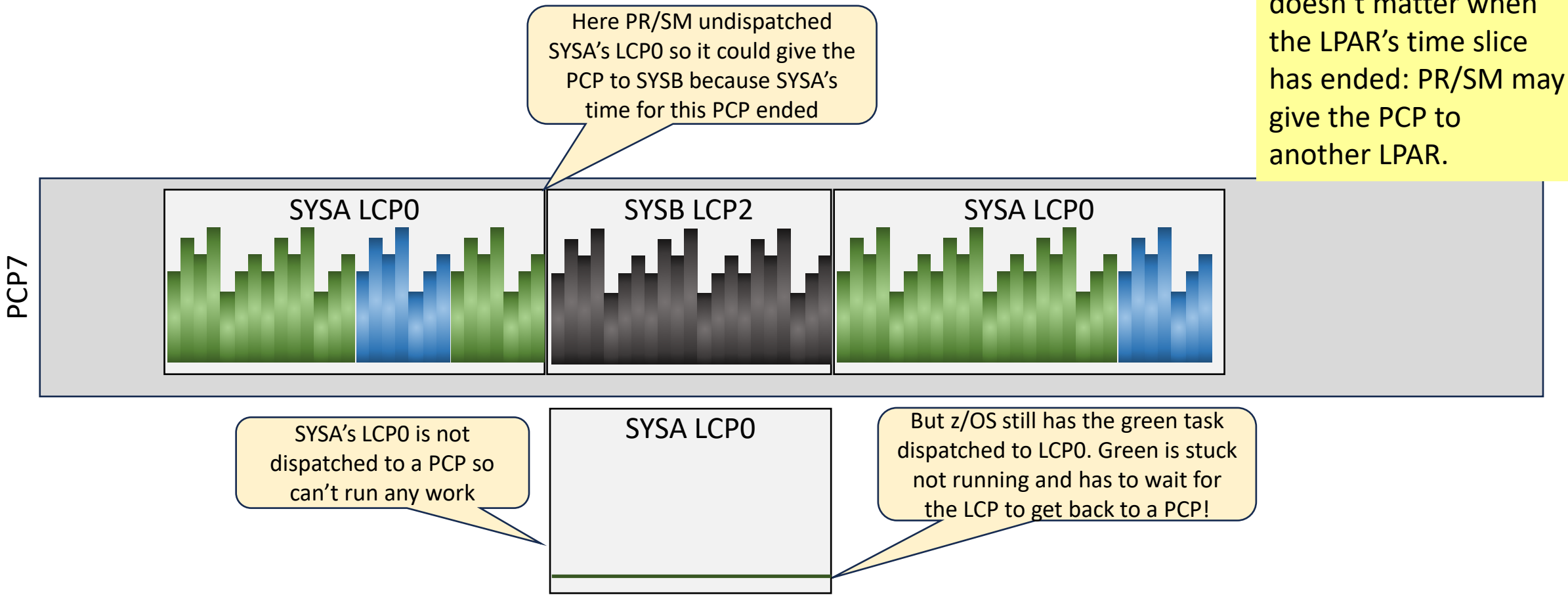
PR/SM Dispatching LCPs



This is the default "wait completion" = "no": z/OS gives up the processor if it doesn't have anything to run on it.

What if z/OS task wasn't done?

Wait completion doesn't matter when the LPAR's time slice has ended: PR/SM may give the PCP to another LPAR.



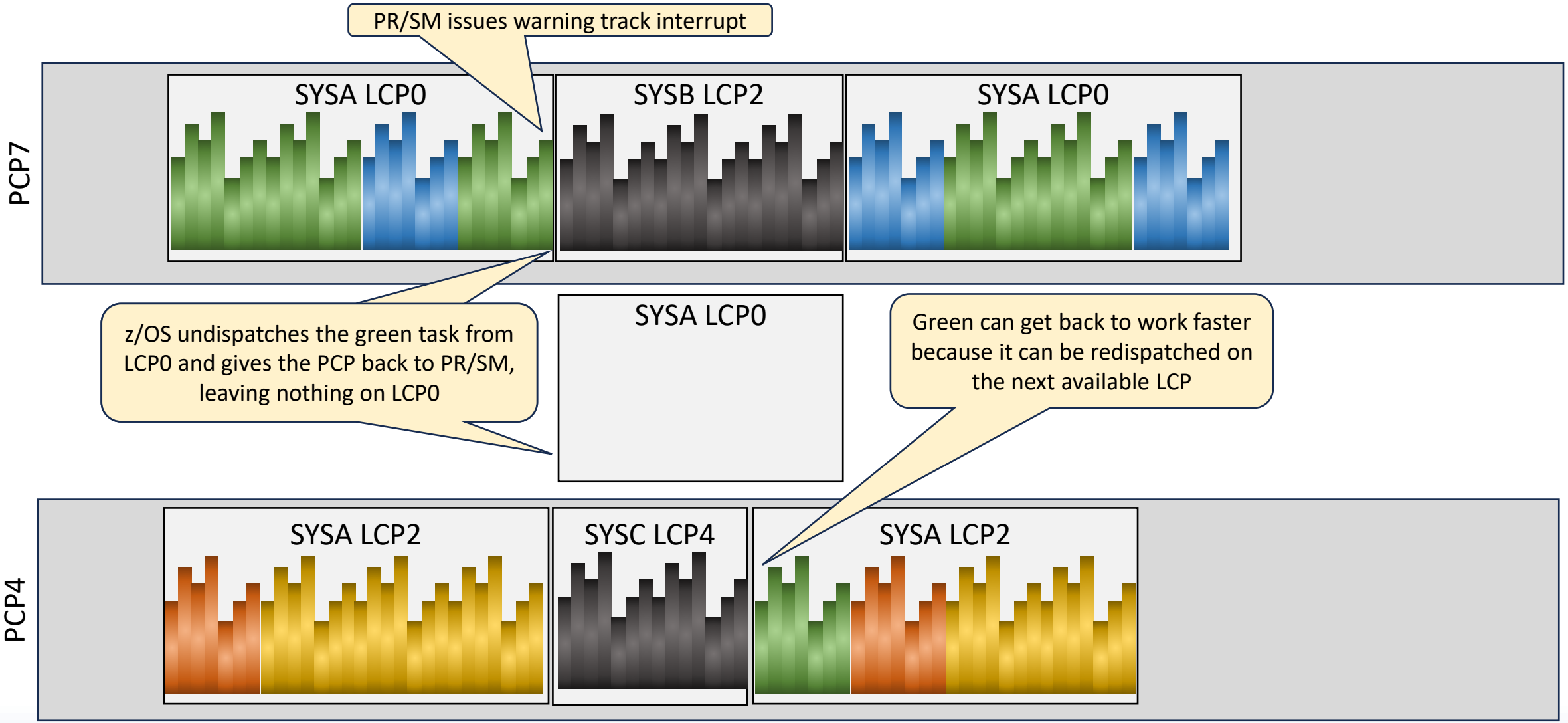
Involuntary Wait Issue

- When PR/SM steals an PCP from a z/OS LPAR when z/OS is still actively using it, the active task remains dispatched to the logical processor but is effectively suspended because it has no hardware to run on
- Note that with HyperDispatch this generally would only be expected to happen for Vertical Medium and Low processors
  - Vertical Highs are quasi-dedicated to the LPAR so if the LPAR's time slice ended but still has demand PR/SM would be expected to give the PCP back to the LPAR
- For Vertical Medium and Low processors, the PR/SM dispatch interval is generally expected to be between 12.5 and 25ms
  - This can be a long time for a task to be involuntarily stranded
  - Worse: VLs might not come back for seconds (or longer) if they get parked
  - Can be especially painful if an important task gets stuck this way!

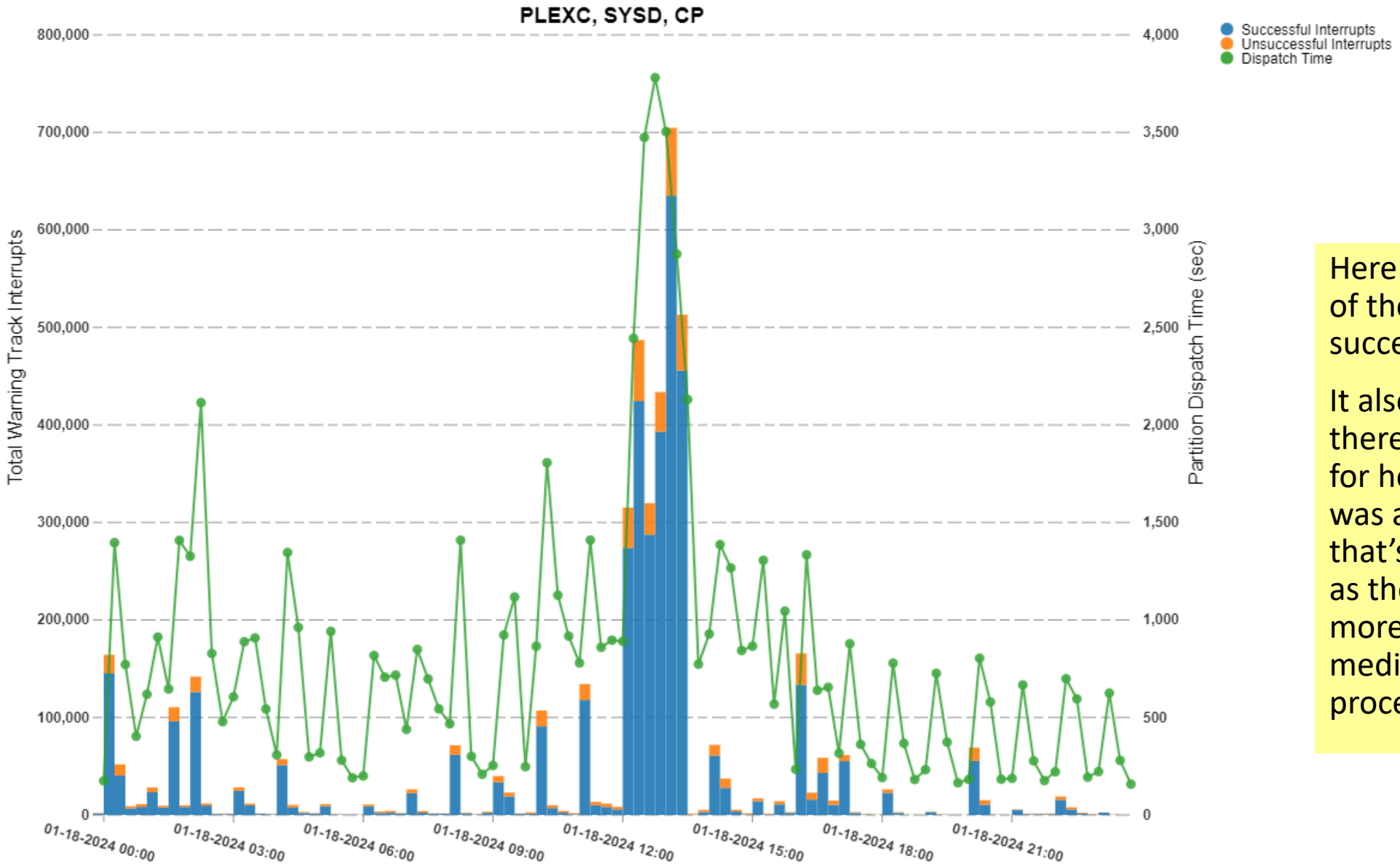
Warning Track Solution

- Starting with the z/EC 12, PR/SM issues a warning track interrupt (WTI) to z/OS that it's about to take away the processor
- z/OS gets a grace period to undispatch the running task from the LCP and return the PCP to PR/SM
  - z/OS can then redispatch the task to an active LCP
- If z/OS doesn't return the processor in time, PR/SM takes it anyways
  - "Unsuccessful" Warning Track Interrupt
- Should help avoid having work hung on an LCP that's not going to get redispatched for some time

What if z/OS task wasn't done?



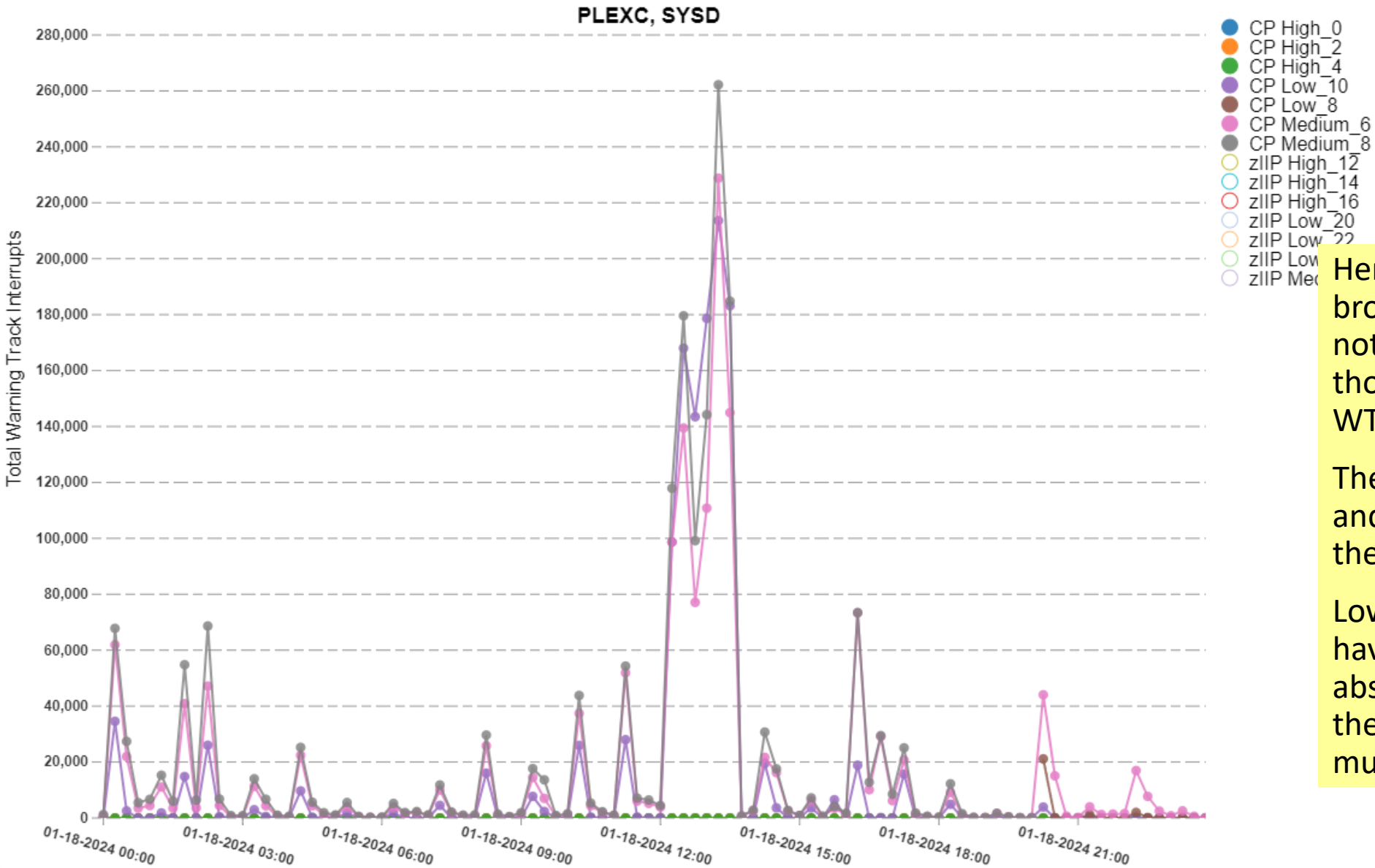
Warning Track Interrupts vs. Dispatch Time



Here we see that most of the WTIs were successful.

It also appears that there was a correlation for how busy the LPAR was and the WTIs. But that's probably because as the LPAR got busier more work was run the medium and low pool processors.

Warning Track Interrupts Per CPU



Here we have the WTIs broken down by LCP and note the polarity of those CPs. VHs took no WTIs, as expected.

The WTIs for the VMs and VLs did increase as the LPAR got busier.

Low Polarity CPs may have fewer WTIs in absolute terms because they may be parked for much of the interval.

Summary

General Conclusions

- Effects of HiperDispatch most obvious on LPARs with several CPs
  - VH's benefits more commonly seen in that situation
  - But still has value on LPARs with fewer CPs too
- Efficiency by polarity can be confusing
  - Especially when there's relatively few CPs
- On LPARs with a VM and VL, the unparked VL is effectively a VM
- On larger LPARs, VLs that are regularly used may be similar to VMs
  - But as the CEC gets busier, they will suffer more and become less efficient
- *Usually* the CPs handling I/O interrupts will be a bit less efficient
  - The VH handling I/O interrupts may be less efficient than the VM that's not
  - But if the CP has little to do other than I/O, it might appear more efficient
- VMs and VLs will take WTIs which may reduce their dispatch interval

Less I/O = more efficient CPU

Summary: How much should you care?

- Probably not much: set your LPAR weights appropriately and enable HiperDispatch and you probably don't need to worry much beyond that
- I/O interrupts being handled by the processors with more assigned weight is a good thing because it helps ensure interrupts aren't delayed
  - Avoiding I/O means fewer I/O interrupts which can impact CPU efficiency
- Warning track is a good thing, but the number of WTIs is a bit surprising
  - Some WTIs may still be unsuccessful so still possible to strand work on an LCP
- A VM -> VH conversion might not result in any significant improvement
  - I'll even say: probably won't in *most* cases
- Correcting weights to avoid using VL is still good & beneficial practice
  - Some minor sporadic use of VL may be fine, but "flapping" VLs will likely be less efficient
  - Avoids even small possibility of stranding work on a VL LCP that gets parked
  - Using automation to change weights may be a very good idea

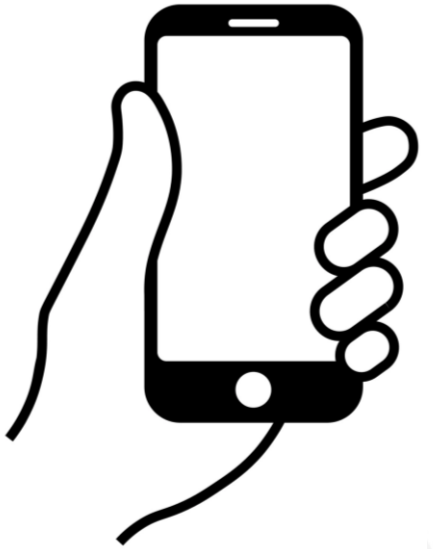
Your feedback is important!

SHARE
EDUCATE • NETWORK • INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation

SHARE mobile app



Questions?