LPAR Configurations to Avoid

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!)

PR/SM has been around for nearly 40 years now. PR/SM technology has advanced and evolved over these last 40 years. This also means that the most efficient configuration strategies haven also changed. What may have been a recommended configuration years ago may no longer be the best configuration today.

During this presentation, Scott Chapman will discuss PR/SM LPAR configurations to avoid. Scott will review these configurations and then explain why these configurations are not recommended. You will learn more about PR/SM, LPAR configurations, and processor measurements during this presentation.

Things we'll talk about

- Why do you care?
- Misc. Parameters
- Memory
- LPAR Weight
- CPs
- Summary

Why do you care about LPAR Configuration?

- Performance:
  - Improper configuration can impact performance
- Financial
  - Improper configuration can increase your CPU consumption, potentially increasing your software bill
- Understanding
  - Improper configuration can limit the data available for understanding and tuning your system

- In this presentation we'll call out some specific configuration problems that we've run into while helping customers optimize their environment

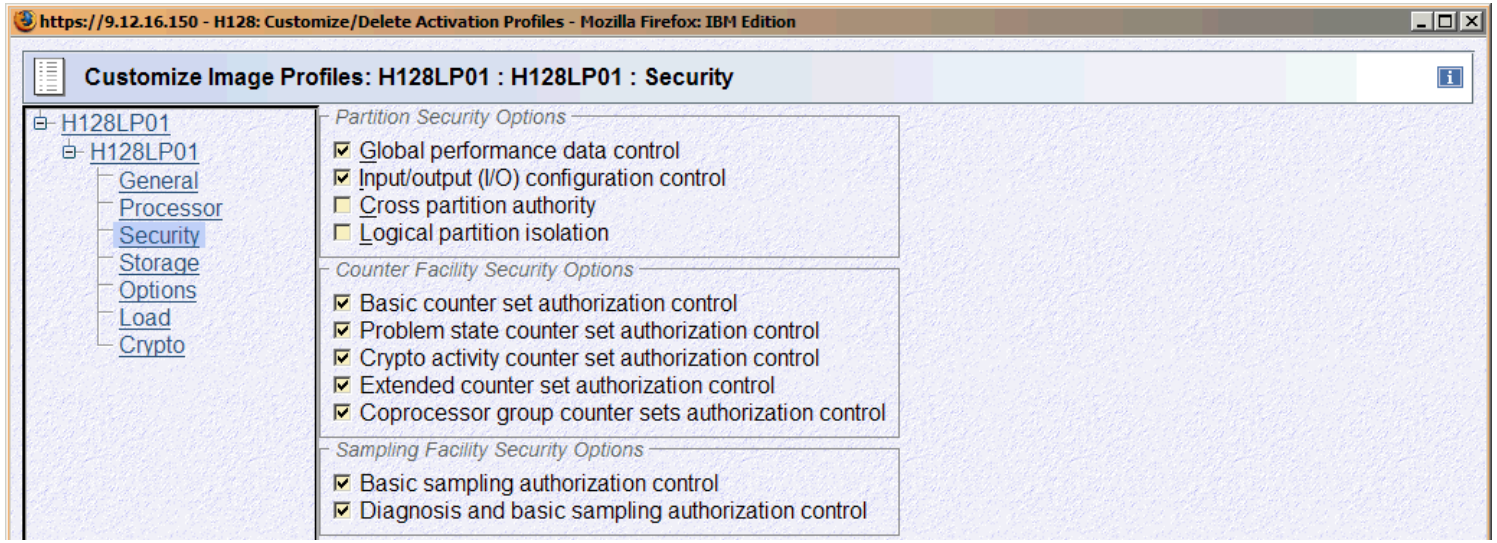
LPAR Configuration Parameters

Problem #1 (rare)

- Global Performance Data Control Authority disabled
  - This let's LPARs see all other LPARs' CPU utilization
  - Generally recommended and usually enabled, except sometimes in outsourced environments
- With this disabled an LPAR doesn't know how many other LPARs are on the machine or how much of the capacity that they are using
  - Can impact WLM's ability to manage low-pool processors as expected
  - Can make it difficult to determine the reason for certain performance problems
- Don't make your (or WLM's) job harder!

Problem #2 (unusual)

- CPU Measurement Facility Counter sets not enabled
  - Basic
  - Problem state
  - Crypto activity
  - Extended counters
  - Coprocessor group
  - SMT diagnostics



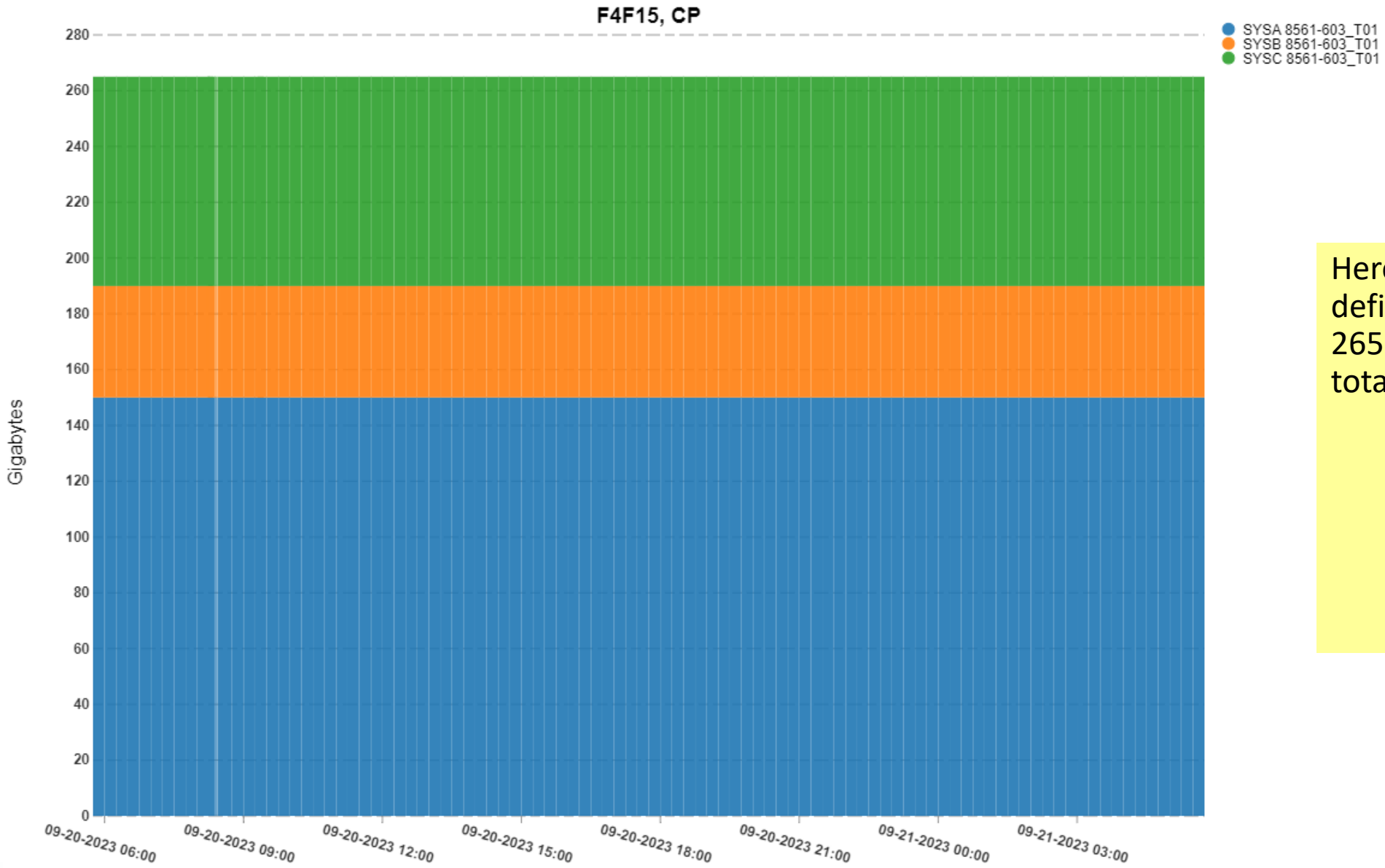
- Basically, enable them all to get all the SMF 113 measurements
  - Can provide detailed CPU hardware metrics that can be useful, especially around upgrades
  - No measurable impact by having these enabled
- Sampling more rarely used, but no harm in enabling it too

Memory

Old Thinking -> New Thinking

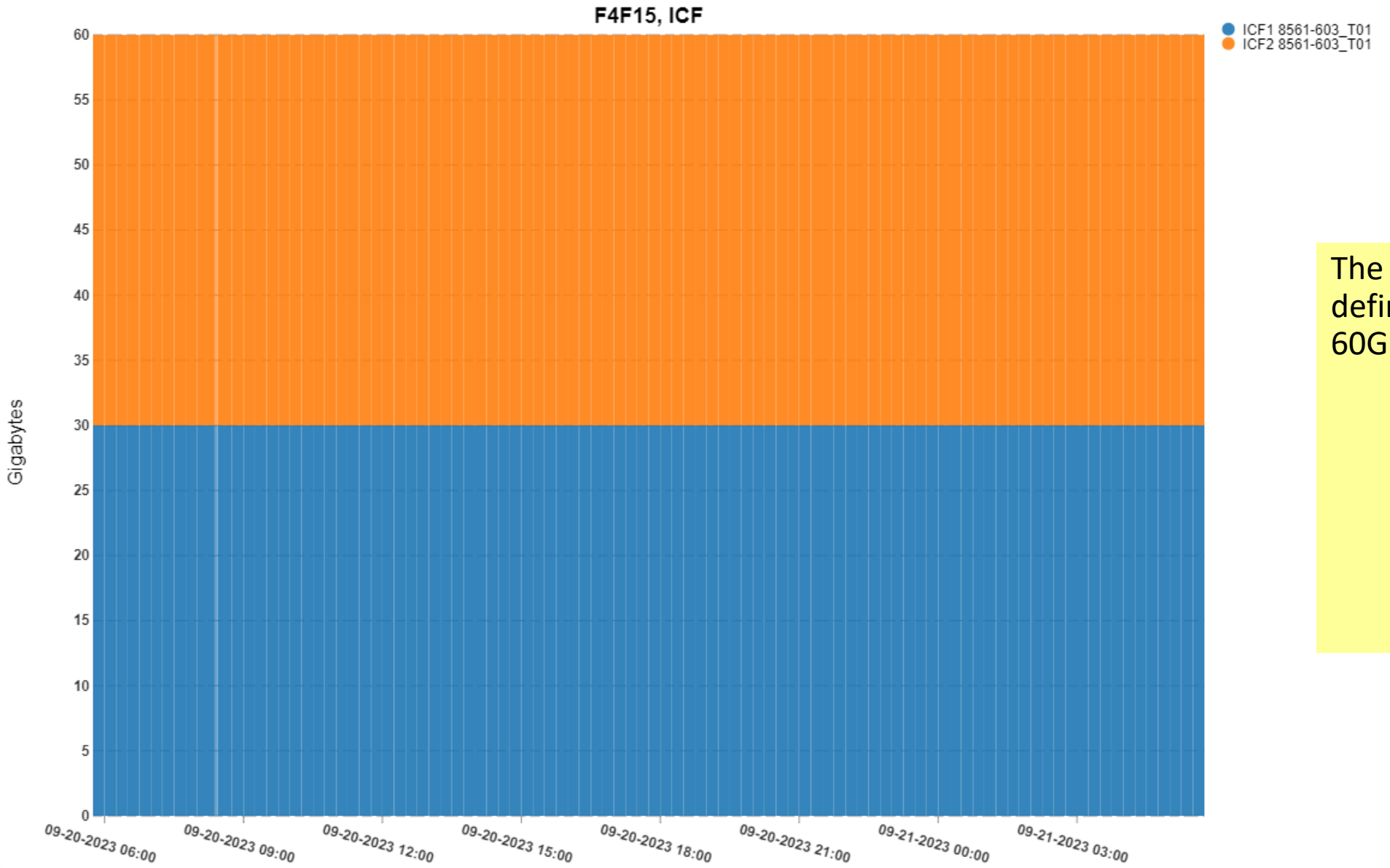
- Old:
  - Memory is expensive, we need to be careful how we hand it out
  - Hold back any memory that's not actively needed by the LPARs in case we have an emergency need for more memory
  - Be stingy in what we allow the address spaces to use
  - We have to precisely set LFAREA
- New:
  - Only hold back minimal amount of memory to account for growth
  - The LPARs have generous cushions and we monitor for growth to avoid surprises
  - Memory is a lot cheaper than CPU, can we improve performance and potentially reduce CPU consumption by being generous with memory?
  - LFAREA can be set much more generously because of z/OS 2.3 changes
- Still:
  - Don't be reckless: IEFUSI (and/or SMFLIMxx) are still a good idea!

CEC Average LPAR Storage Online



Here the z/OS LPARs are defined with about 265GB of memory in total.

CEC Average LPAR Storage Online



The CF LPARs are defined with another 60GB of memory

Problem #3

- Some sites have held back way more memory than likely makes sense
  - We can't always (even usually?) tell this because the total installed memory is not in the SMF data, but…
- The minimal orderable memory on the z15 T01, z16 A01 machines is 512GB
  - So when we total up the LPARs and come to 325GB we do wonder if the rest of that memory could be put to effective use!
- If you have a plan for using this (e.g. you're planning on standing up some additional LPARs), then fine. But otherwise: use what you have!
- Scott's ROT re. reserved memory:
  - Old: keep 10% of installed memory in reserve
  - New: keep 10% (or less) of largest LPAR in reserve
  - Subject of course to site-specific situations

z/OS Configuration

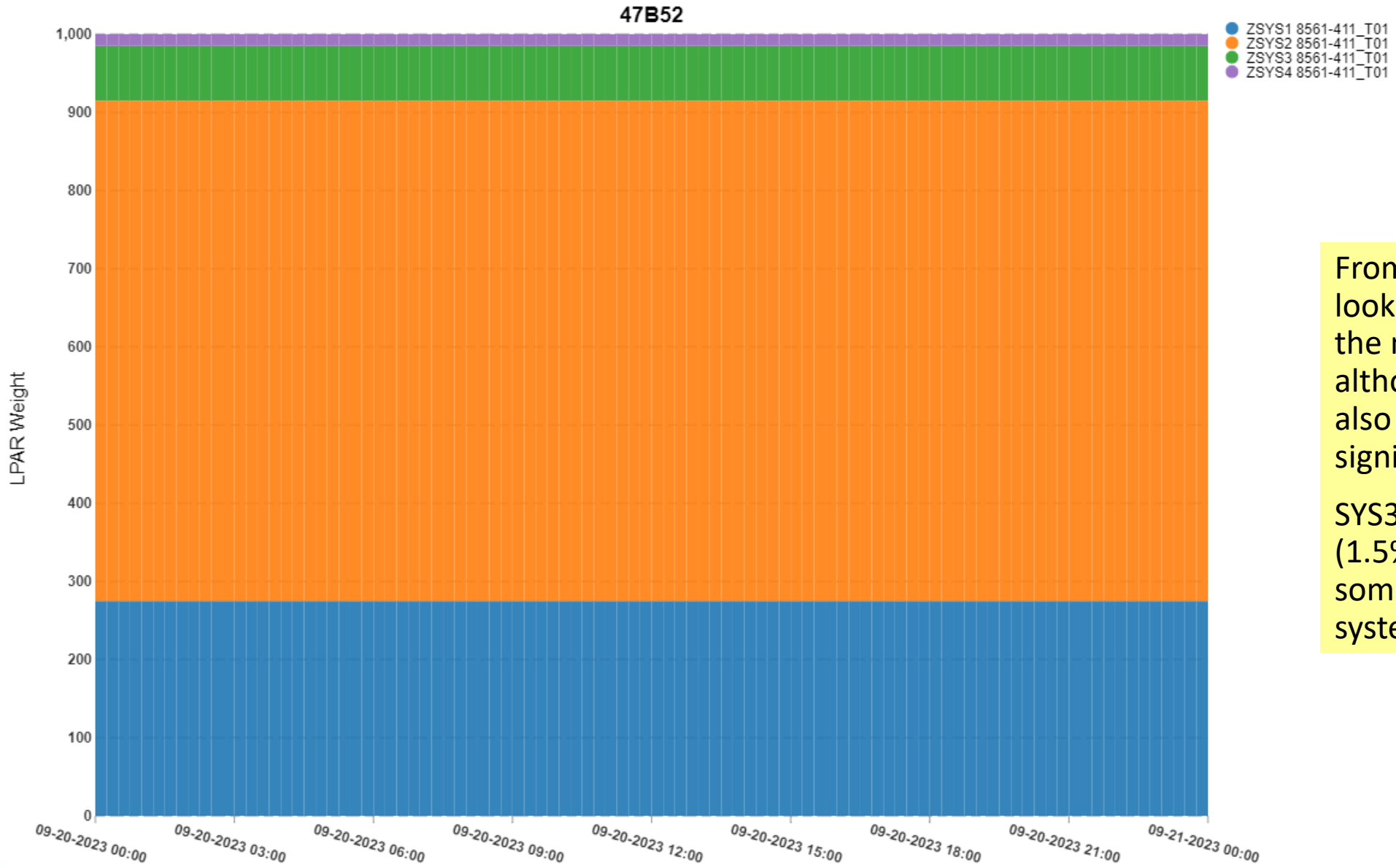
- Once the LPARs have been given the memory: allow the address spaces to use it!
- Be generous in your LFA-area size for fixed 1MB pages
  - With z/OS 2.3 there were changes in how z/OS manages the storage areas and the fixed 1MB limit is now treated strictly as a limit: a block of fixed 1MB pages is no longer set aside in the same manner as prior releases
  - 2GB pages more rarely used and still should be more closely managed (they do get a set-aside block of memory that can't be decomposed to smaller pages)
- See also:
  - https://www.pivotor.com/library/content/Chapman_MemoryMgtEvolutionWebinar.pdf

LPAR Weight

Problem #4

- We often see LPAR with sub-optimal weights
  - LPARs should have weight to match their capacity requirements
    - While most sites allow LPARs to borrow unused capacity from other LPARs, this is not optimal
  - Some LPARs may benefit from small adjustments to get an additional high-pool CP
  - Sometimes this is because LPARs have been added or removed
    - LPAR's fair share = LPAR's weight / total weight of all *activated* LPARs
    - Sometimes DR or test LPARs are shutdown but not deactivated meaning their weight is still influencing the active LPAR's share
- LPARs consuming more than their weights are at risk of being stolen from
  - This could have a significant negative impact on the work
- Weight concerns also relate to processor efficiency under HiperDispatch

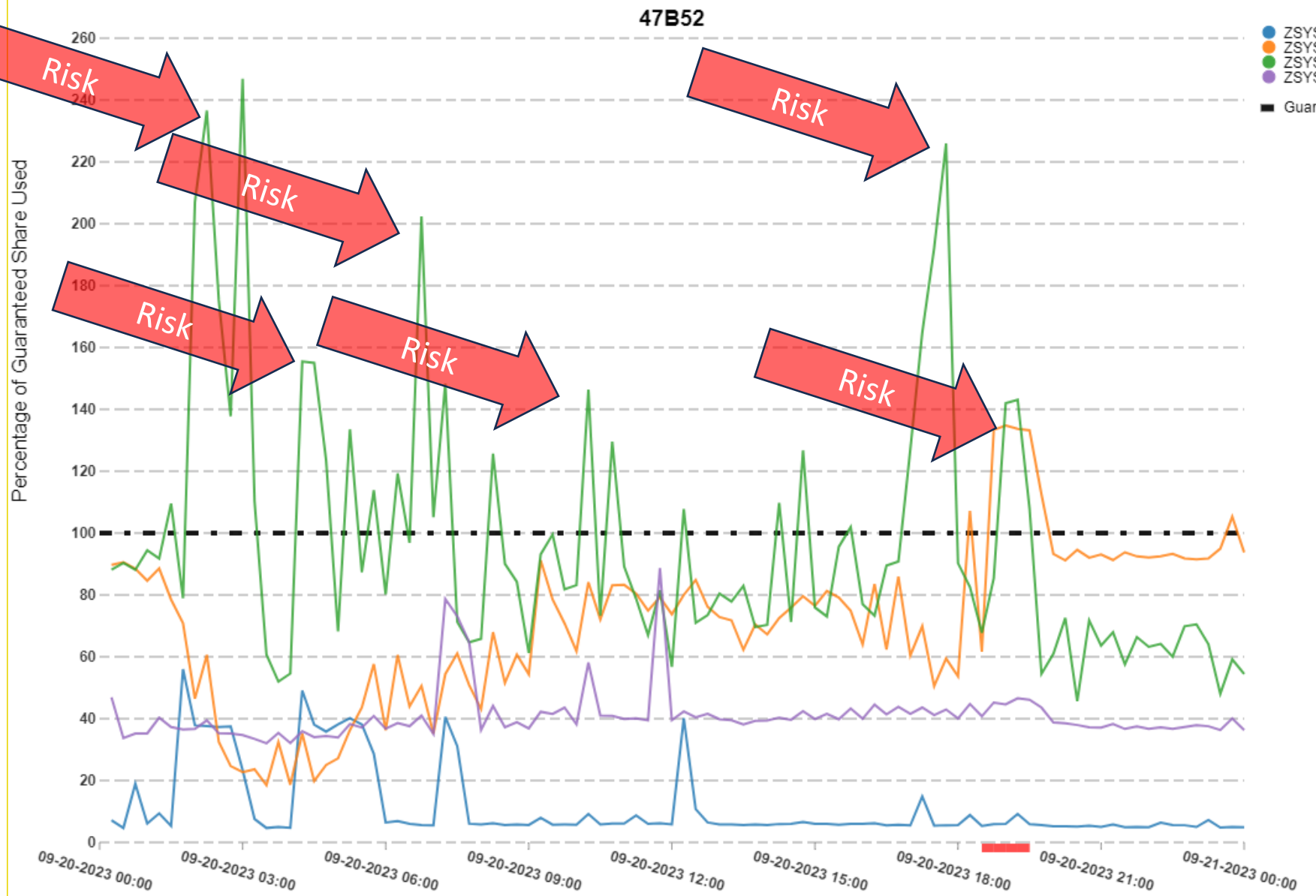
CEC Assigned CP LPAR Weights



From the LPAR weights it looks like SYS2 (64%) has the most work to do, although SYS1 (27.5%) also seems quite significant.

SYS3 (7%) and SYS4 (1.5%) are probably some sort of dev/test systems.

CEC Percent CP Weight Used



This shows how much capacity each LPAR is consuming relative to its weight.

Note SYS1 rarely even consumes 50% of its weight, while SYS2 in the early evening needs much more than its weight.

SYS3 regularly well above its weight.

Risk mitigation

- What happens if SYS1 has an increase in demand in the early evening?
  - SYS2 (and SYS3) are going to lose access to that weight they're borrowing from SYS1
  - This will add CPU delay to SYS2 and SYS3, impacting work running there
- It may be better to give some of SYS1's weight to SYS2 and (maybe) SYS3
  - Unless SYS1 is in fact really important, and we want it to guarantee access to that capacity
- In some cases it may make sense to move weight between LPARs at different times of the day
  - This can be done with BCPii via some REXX (or C or Assembler) code
  - See also: https://github.com/IBM/zOS-BCPii
  - In theory, just-in-time weight movement might be an interesting ML opportunity
    - Actually: probably don't need ML at all

HiperDispatch

- 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 essentially dedicated to the LPAR (100% share)
  - Medium – Share between 0% and 100%
  - Low – Unneeded to satisfy LPAR’s weight
- 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

Highs and Lows

- In the prior example, it's possible that SYS2 or SYS3 might be able to get an additional high pool processor based on getting additional weight from SYS1
- Generally speaking:
  - High pool processors are more efficient
  - Low pool processors are less efficient
- "Efficient" = less CPU consumption required to do a given amount of work
  - Efficient = better performance
  - Efficient = less capacity consumption
  - Less capacity consumption may mean lower software costs
- In the next webinar we'll explore how different this really is

CPs

Problem #5

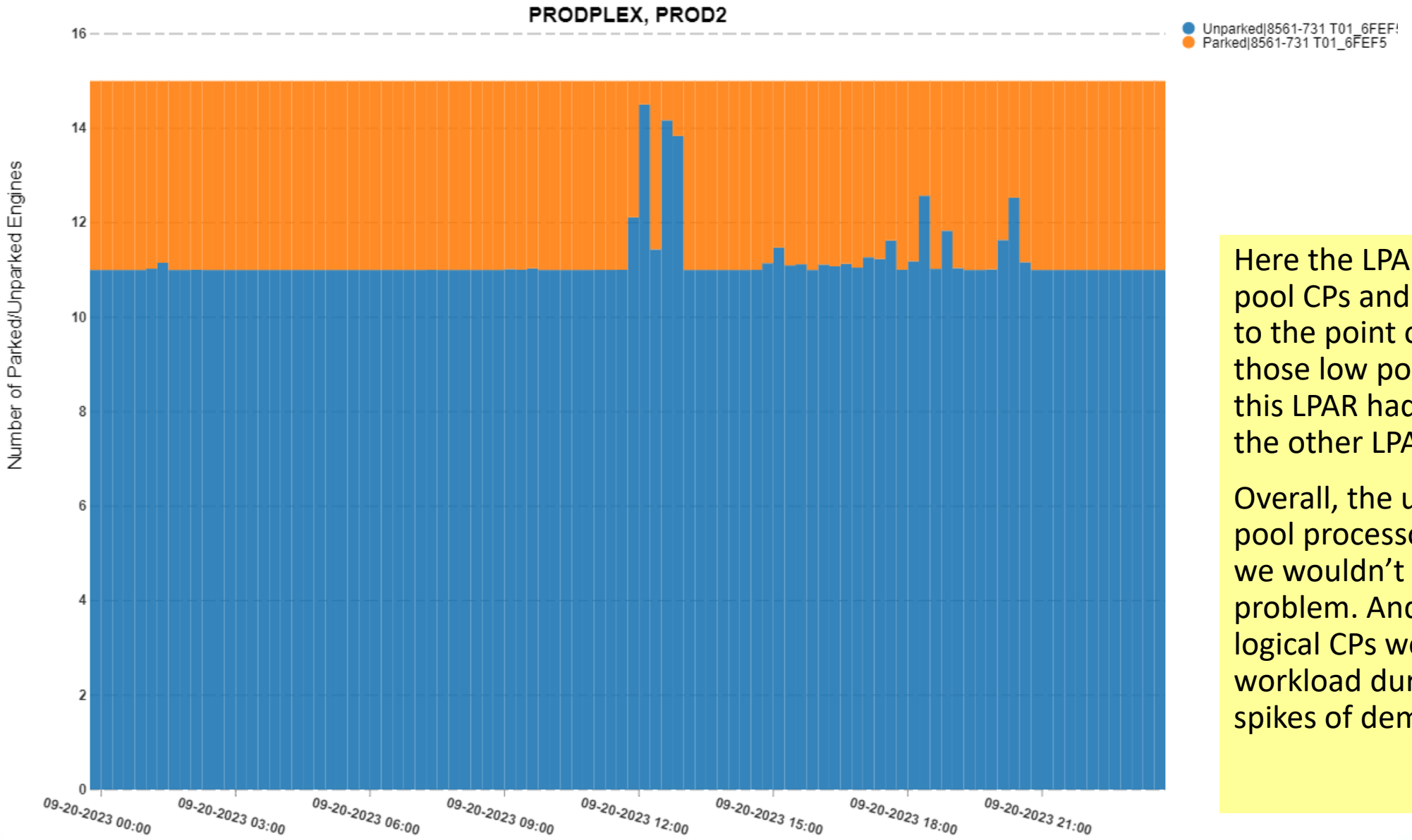
- Sometimes we find LPARs that have too many or too few logical CPs
  - "CPs" here can mean either GCPs or zIIPs
- Too many implies too many low-pool processors
  - IBM recommendation: have no more than 2 low-pool processors
  - Scott's recommendation: it depends…
    - Unused low pool processors don't hurt
    - Used low pool processors imply a need for weight adjustments
    - Occasional brief use of low pool processors is probably fine
- Too few implies possible limitations imposed on the work
  - Not enough CPs to dispatch on
  - Also: avoid single-engine z/OS LPARs
- Because too few is more impactful than too many, better to over define if possible
  - This was not true prior to HiperDispatch

For large systems: limit CPs/zIIPs to less than the number in a drawer

Logical vs. Physical CPs

- LPARs use “logical” CPs which are allocated some fraction of a physical CP
  - Highs: 100%
  - Lows: 0%
  - Medium: >0%, <100%
  - Especially for medium and lows, there will be times when the logical CP is not dispatched by PR/SM to a physical CP
- LPAR cannot have more logicals defined than physicals enabled
  - But can have reserve CPs that can be brought online during CoD/CBU events
- If all the logical CPs are busy, it doesn’t matter how much additional physical capacity is available, the LPAR won’t be able to get to it
  - This can be used as an intentional limiting factor (although there may be better ways)

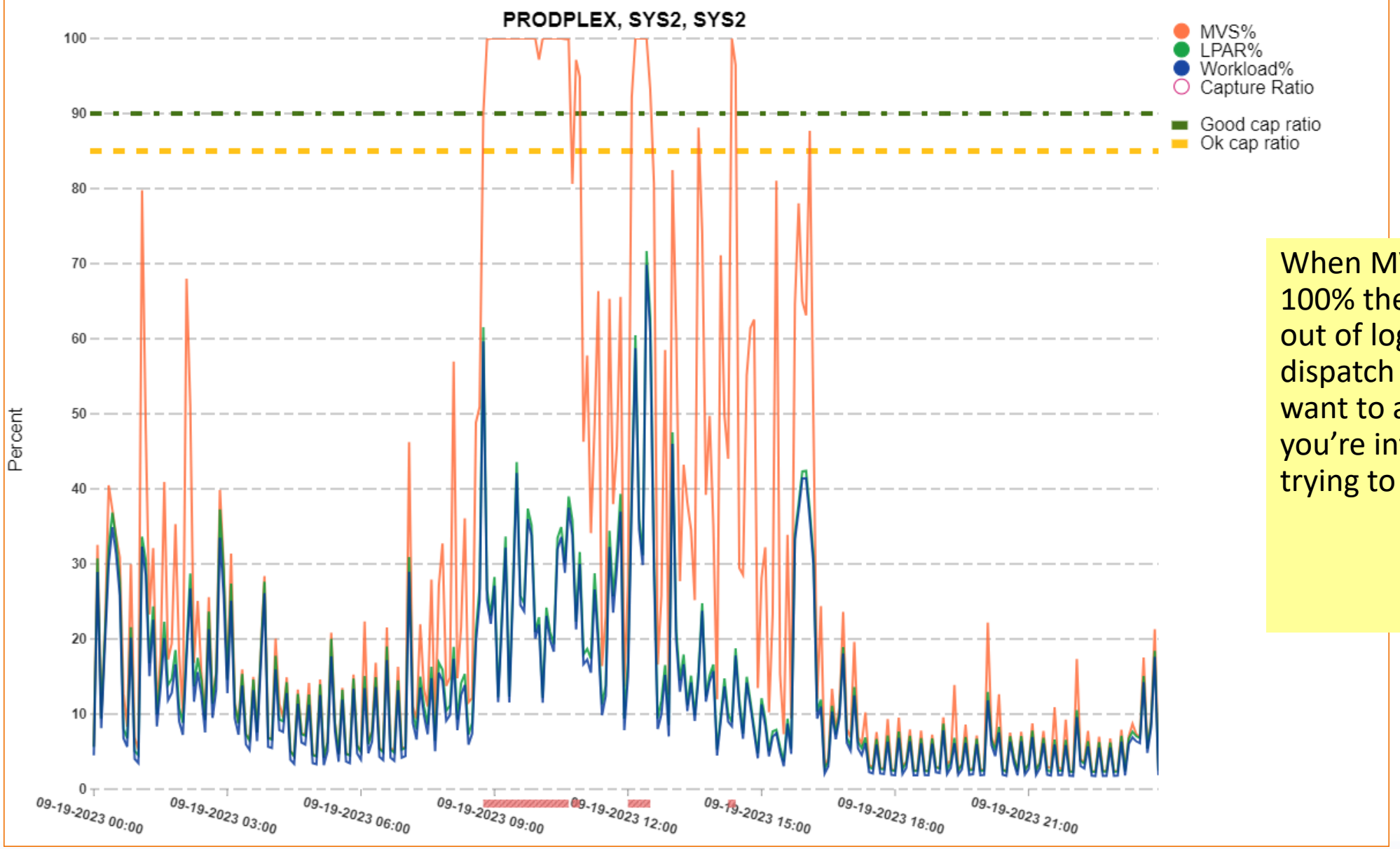
HiperDispatch - Parked / Unparked CPs



Here the LPAR has 4 low pool CPs and did (rarely) get to the point of using all 4 of those low pool CPs when this LPAR had demand and the other LPARs did not.

Overall, the usage of low pool processors is limited, so we wouldn't consider this a problem. And taking away logical CPs would limit the workload during those spikes of demand.

LPAR, MVS, and Workload CP Busy% with Capture Ratio



When MVS Busy goes to 100% the LPAR has run out of logicals to dispatch on. Generally want to avoid this unless you're intentionally trying to limit the LPAR.

Bonus Problem #6

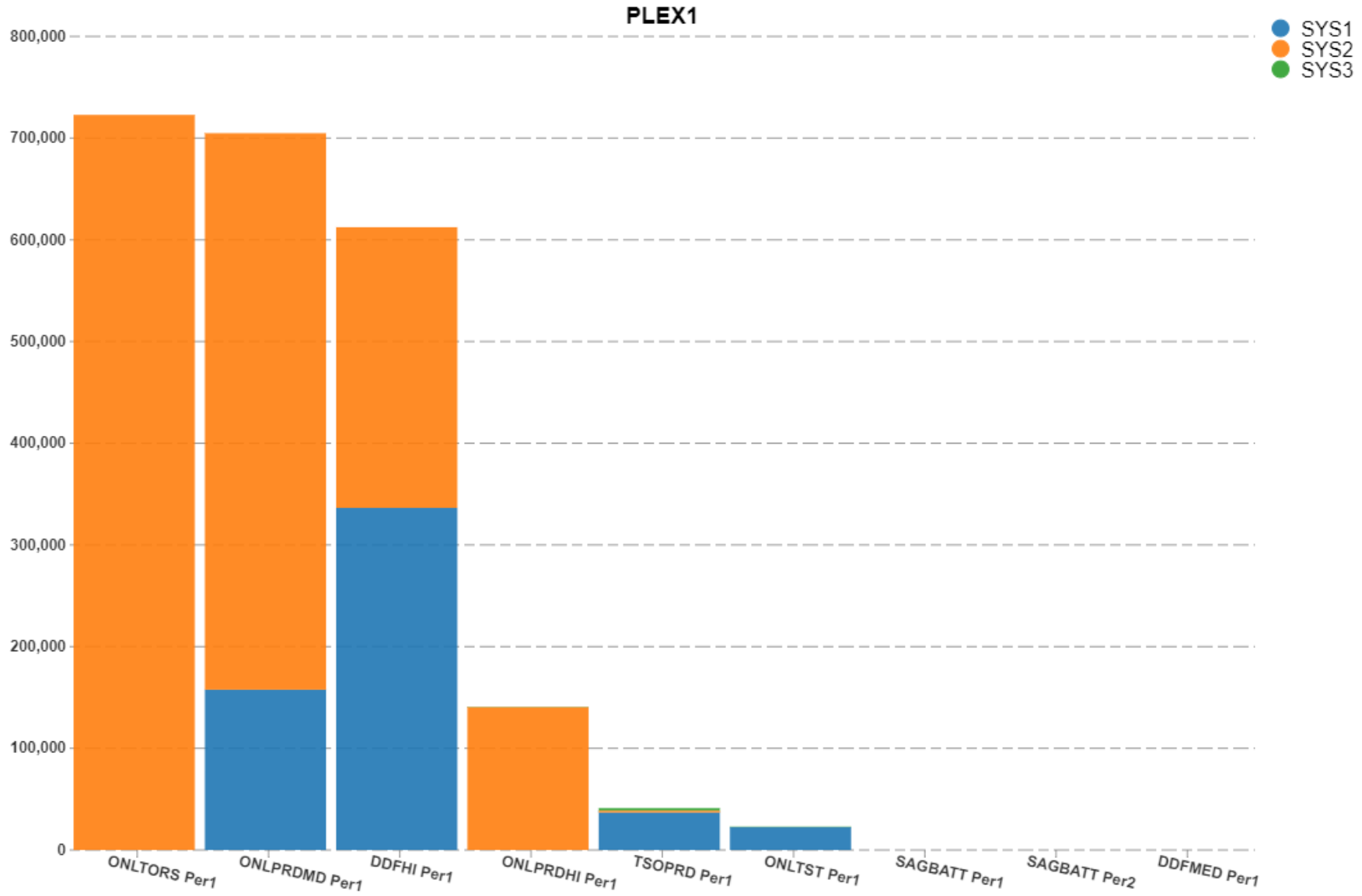
- **You need enough physical CPs to dispatch on**
  - We've seen multiple cases where customers have had issues after going to fewer/faster CPs
- In most cases, more/slower is better than fewer/faster CPs
  - Most systems have multiple LPARs share those physical CPs
  - Most LPARs have many tasks, often trying to run at the same time
- Sometimes do have single-TCB task issues
  - Most often CICS regions that are constrained by QR TCB
  - If you have this situation, you should look at application changes to resolve it
    - Thread-safe
    - Splitting the work across multiple AORs
    - Application tuning

Recent case in point

- Customer upgraded from z14 to z16
  - From: z14 3907-T05, 5 CPs, 450 MSUs
  - To: z16 3932-W03, 3 CPs, 455 MSUs
- Reported “some issues” after upgrade
- Comparison of similar days follows…

WLM RT Goal - Top Ended Transactions

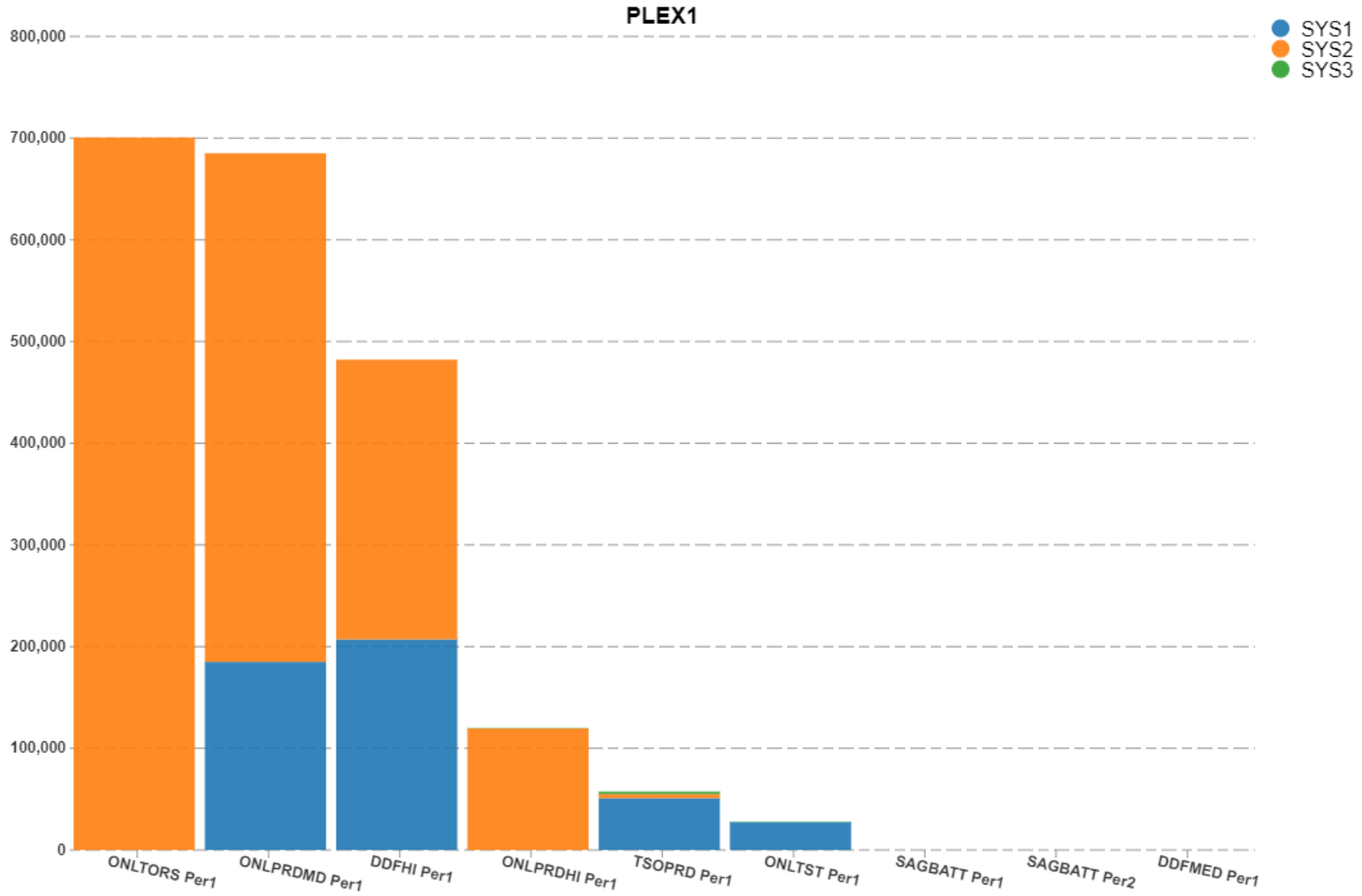
(Period of Study for RT Goal Periods)



Before

WLM RT Goal - Top Ended Transactions

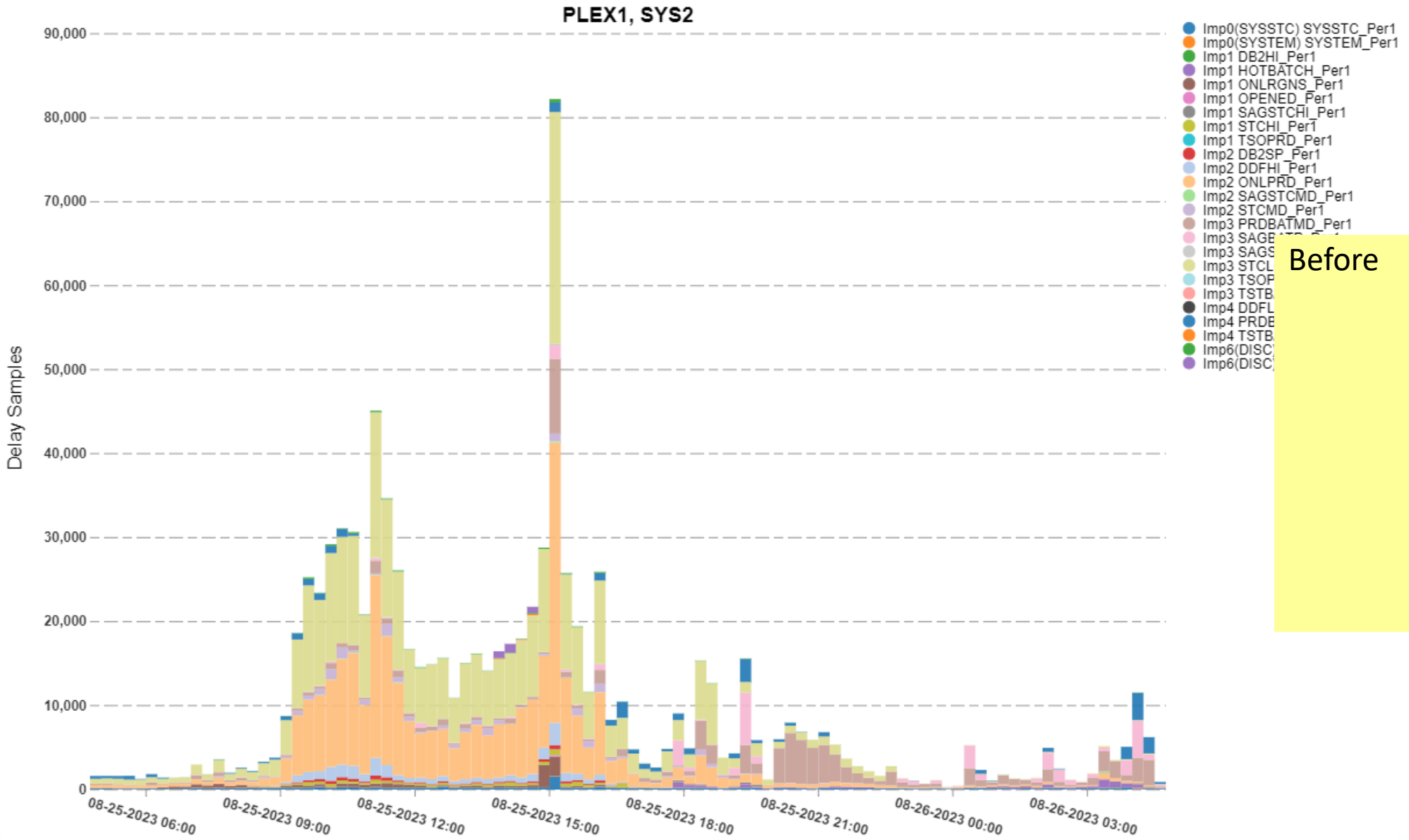
(Period of Study for RT Goal Periods)



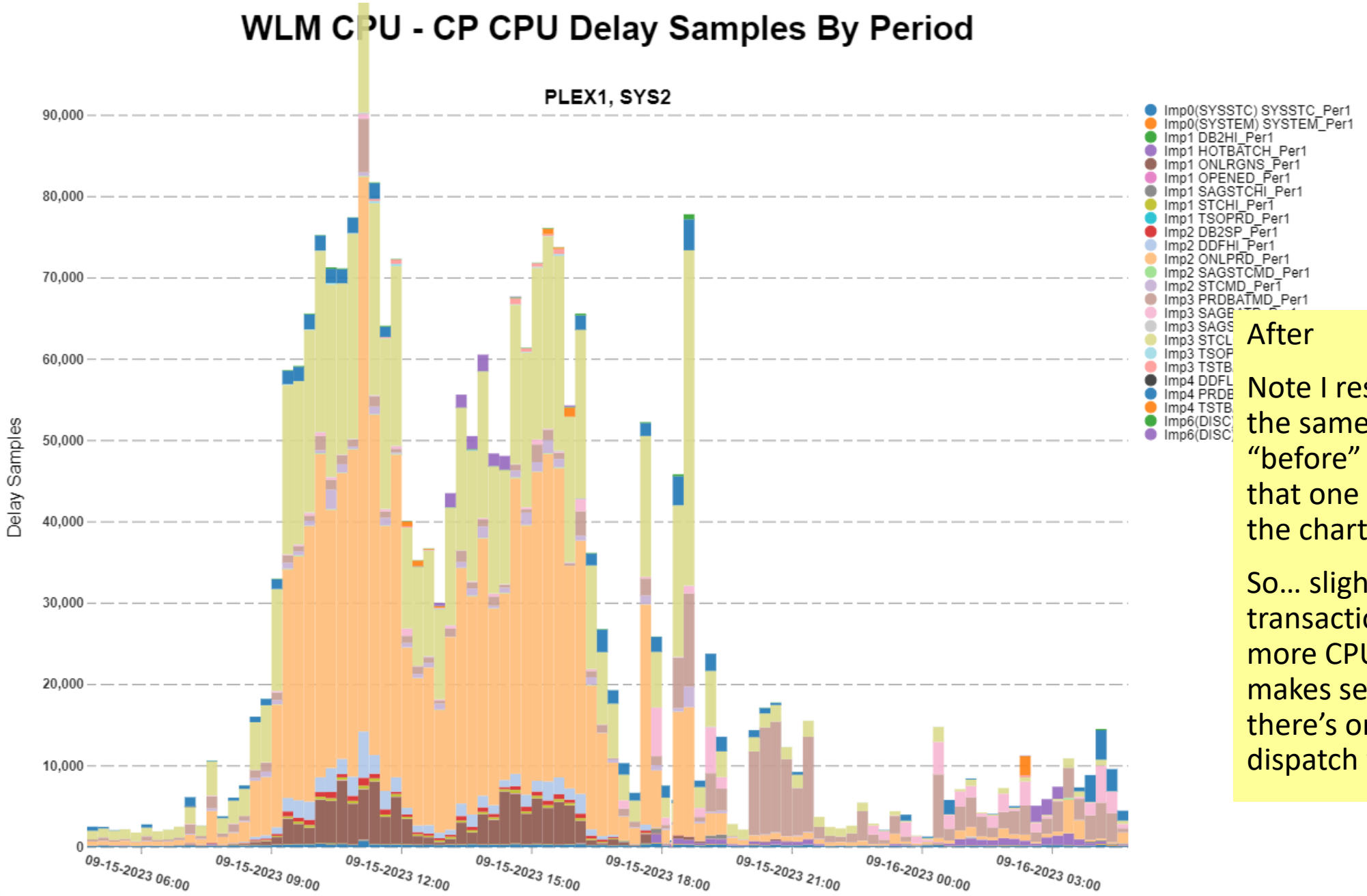
After

Note very similar (but slightly lower) transaction volumes for the day

WLM CPU - CP CPU Delay Samples By Period



Before

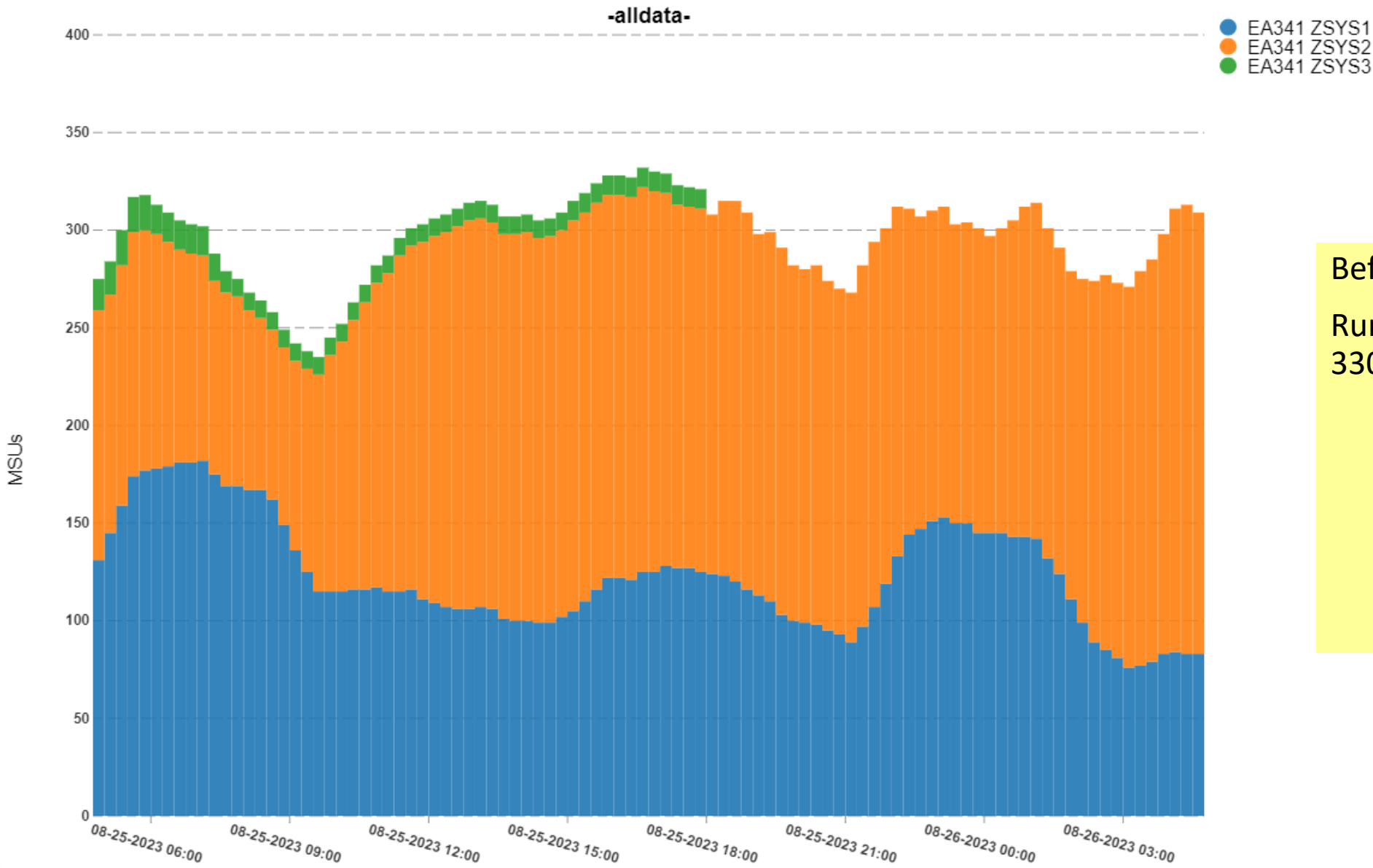


After

Note I rescaled this to the same scale as “before” which is why that one interval is off the chart.

So… slightly lower transaction load, but more CPU delay. This makes sense given that there’s only 3 CPs to dispatch work on.

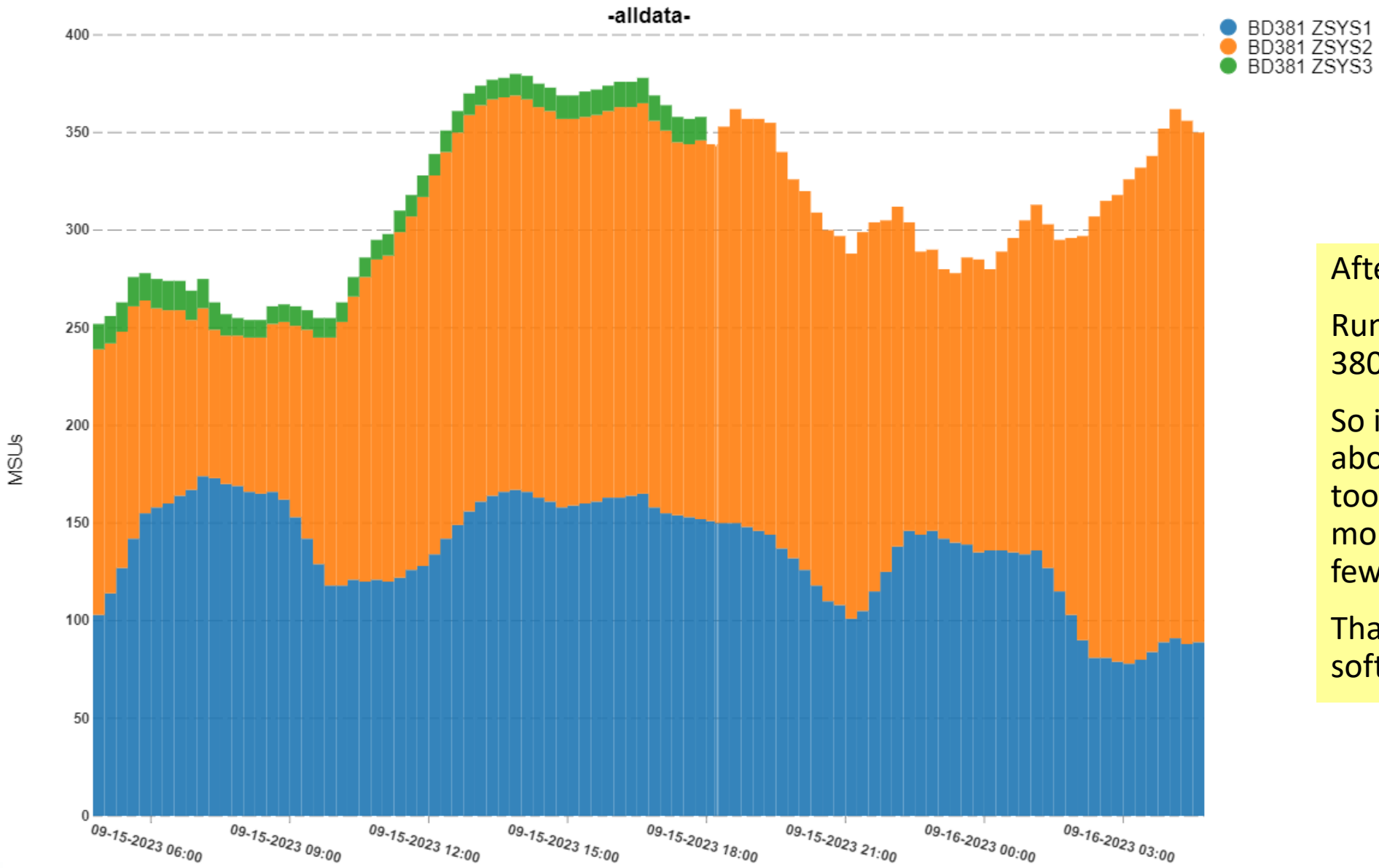
Long-term (R4HA) Average MSUs for All LPARs



Before

Running below about 330 MSUs

Long-term (R4HA) Average MSUs for All LPARs



After

Running around 370 – 380 MSUs

So it appears that to do about the same work it took upwards of 15% more MSUs on fewer/faster engines.

That could impact their software bill.

Important notes about this example

- I didn't try to examine all the workloads to identify which ones exactly went up vs. down
- The chosen before/after days were a couple of weeks apart, so there might have been application changes
  - But very similar results for comparing days just 1 week apart
- WLM did mostly maintain response times for most important workloads
- **I don't believe this is a z14 to z16 issue**
  - In fact, I would have assumed the z16 could have faired better going to fewer/faster
  - Unfortunately, not all counters were enabled on the z16
- **I do believe this is a more/slower to fewer/faster issue**
  - Largest LPAR went from 2H, 2M, 1L to 1H, 1M, 1L
    - Much more sharing of CPs both between and within LPARs

Summary

- Don’t forget to enable all CPU MF Counter sets
- Don’t starve your LPARs or address spaces for memory
- Don’t use weights that leave important LPARs at risk from being stolen from
- Don’t under-define the number of logical CPs: with HiperDispatch too many logical CPs is much less of a problem than too few
- Don’t assume similar sized machine with fewer/faster engines is going to perform better than one with more/slower engines