Sharing CPUs: How z/OS and PR/SM Manage Logical and Physical Processors

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!)

Inside your mainframe there are physical CPUs, but in almost all cases z/OS gets logical CPUs to run work on. This distinction is sometimes confusing to people, especially in relation to how it can affect performance and available capacity. In this session we'll discuss logical CPUs, physical CPUs, and the importance of understanding LPAR configuration options that can impact performance. After attending this session you should have a clear mental model of logical versus physical CPUs as well some ideas about how to optimally configure your important LPARs.

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.epstrategies.com/content.html

z/OS Performance workshops available

During these workshops you will be analyzing your own data!

- Essential z/OS Performance Tuning
  - March 20-24, 2023
- Parallel Sysplex and z/OS Performance Tuning
  - May 2-3, 2023
- WLM Performance and Re-evaluating Goals
  - September 11-15, 2023
- Also… please make sure you are signed up for our free 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
PSP: z/OS Performance Tuning – Some Top Things You May Not Know	Peter Enrico Scott Chapman	Tue 13:15	Strand 12A
z/OS WLM – Revisiting Goals Over Time	Peter Enrico	Tue 16:00	Empire C
Sharing CPUs: How z/OS & PR/SM Manage Logical & Physical Processors	Scott Chapman	Wed 08:00	Empire C
Observability Shootout	Scott & other ISVs	Wed 16:00	Empire C
I/O, I/O It's Home to Memory We (Should) Go	Scott Chapman	Fri 09:15	Strand 12A

Agenda

- Overview of physical z16 CPUs
- Brief discussion of CPU characterizations
- LPARs and PR/SM and logical CPs
- HiperDispatch
- z/OS dispatching

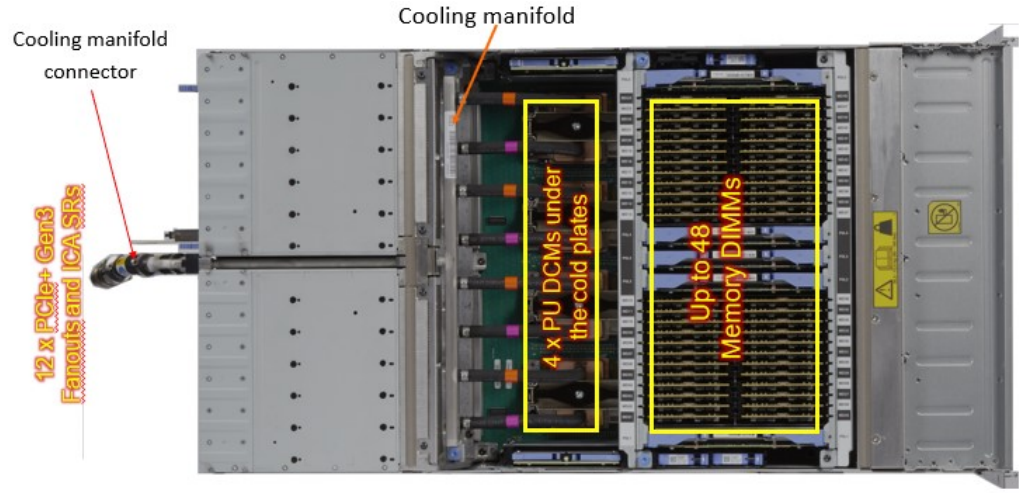
Defining "CPU"

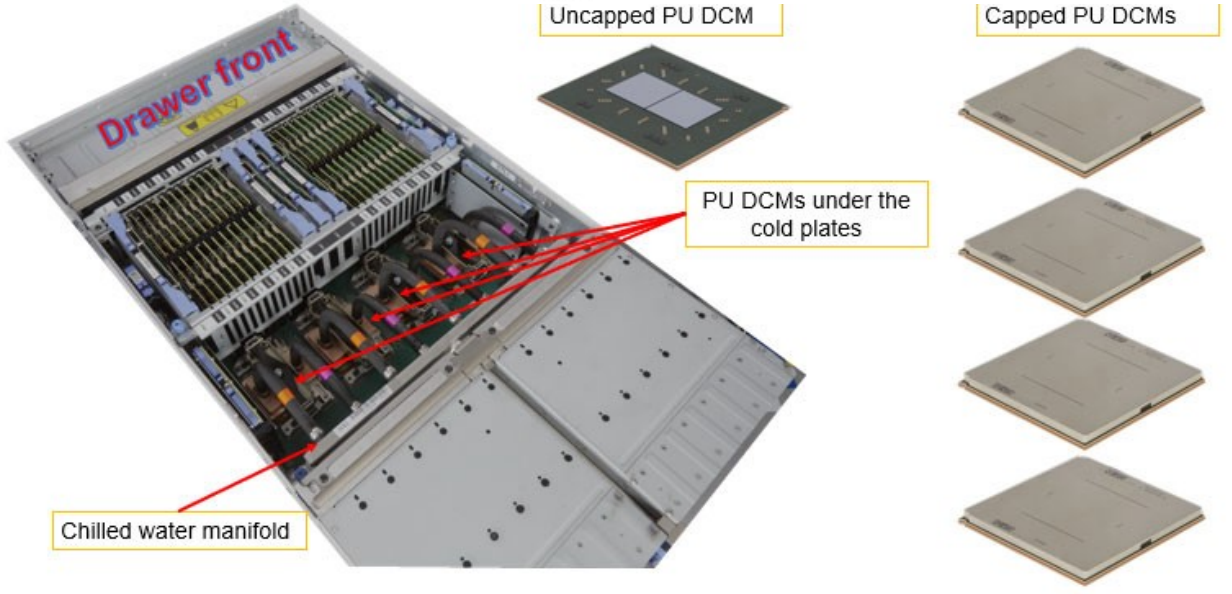
- CPU = Central Processing Unit
  - But that may still be a bit vague
- Sometimes people refer to the whole mainframe box as "the CPU"
  - While not entirely inaccurate, that's not what we're doing here
- Sometimes people refer to a microprocessor chip as a CPU
  - Again, not incorrect but not usually what we mean from an OS perspective
- From the perspective of the operating system, CPU = core on a chip
  - The (tiny) bit of hardware where a stream of instructions are executed
  - But in many cases, what the operating system sees as a CPU may in fact only be a "logical" CPU and not a specific physical core on a chip
  - The above is true of most (all?) operating systems across different architectures

Physical CPUs and Characterizing CPUs

z16 Drawer

- 1 to 4 drawers per z16
- Each drawer has 4 Dual Chip Modules
- Each DCM has 2 z16 Tellum Chips (PU)
- 8 cores per chip (not all may be active)
  - 48 active cores per drawer < Max200
  - 57 active cores per drawer Max200
  - Wafer yields improved by utilizing chips have some cores disabled



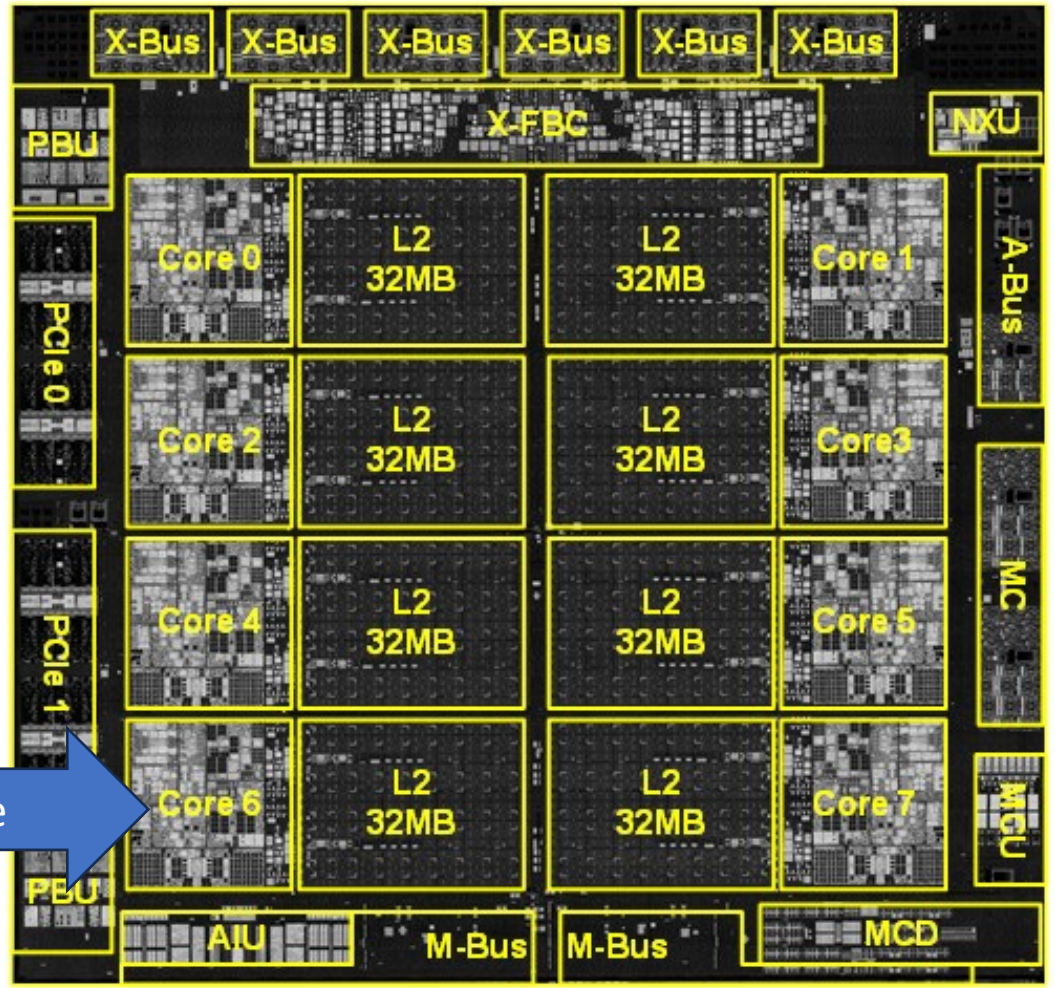


© IBM

© IBM

z16 PU Chip - Telum

- This is one z16 PU (Processor Unit) Chip
  - A bit under 1” square (530 mm^2)
  - 22.5B transistors
- Note large L2 and no specific L3/L4
  - Virtual L3/L4 from sharing L2 between cores
  - Having the data close to the core is important!
  - For active cores, a portion (normally half?) of the L2 for the core is dedicated to the core
- For today, from the physical perspective, lets agree that 1 CPU = 1 Core



Code runs here

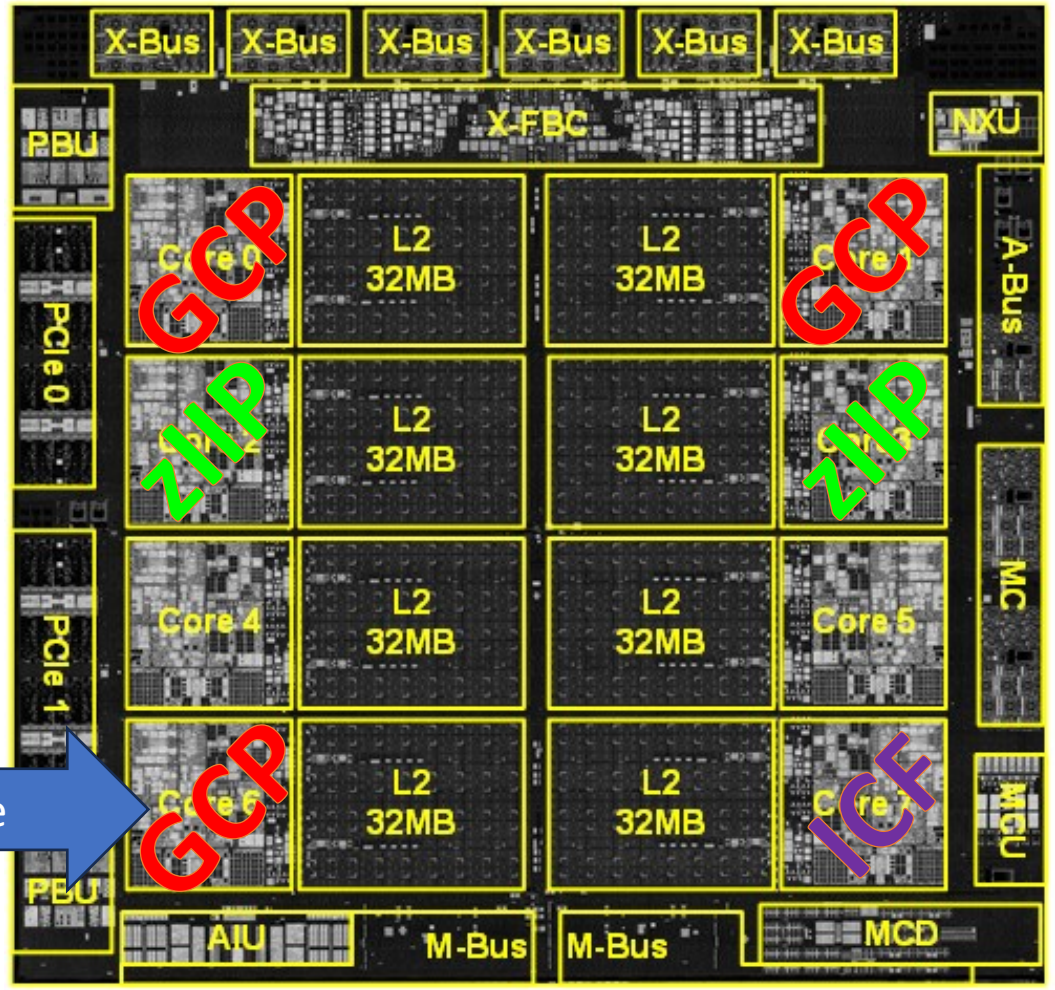
© IBM

CPU Characterization

- Mainframe CPUs (Cores) can be characterized as different “types” of CPUs, but in reality they’re all the same hardware
- This is done for pricing purposes
  - GP / GCP / CP / CPU
  - ICF
  - zIIP
  - IFL
  - SAP
  - IFP
  - Spares (everything not characterized)
- Type set by microcode
  - Enforced by microcode and OS

In this presentation, “CPU” or “CP” is meant to generically refer to any of these characterizations

Code runs here



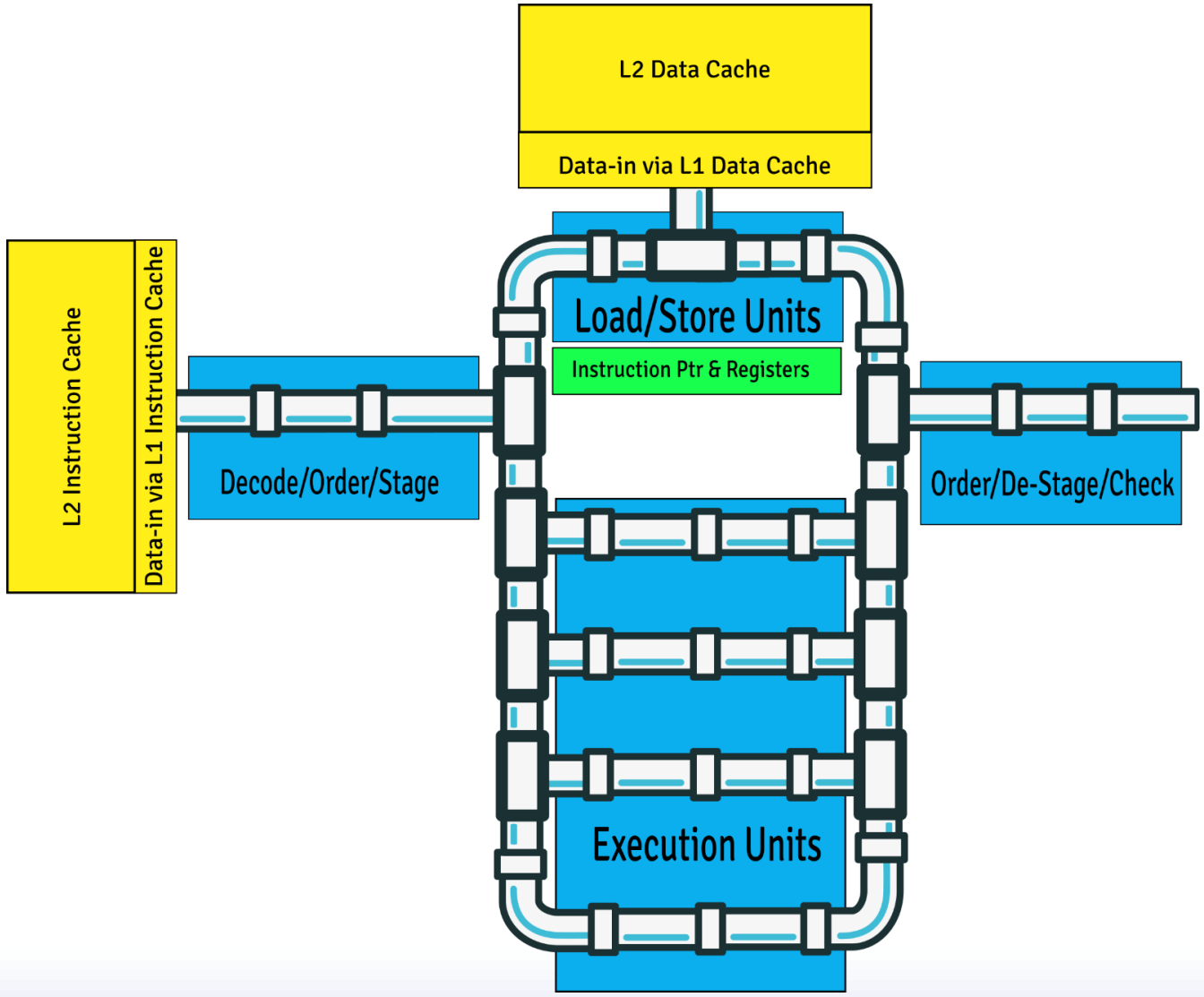
© IBM

CPU Concurrency

- How many things can a CPU be doing concurrently?
- Depends on what a “thing” is!
  - If “thing” is an instruction, then many
  - If “thing” is “executing a program”, then 1 (or maybe 2)
- All programs effectively become a stream of instructions
  - A CPU can only execute 1 (or maybe 2) streams of instructions at a time
- Of course a program might launch another task/thread and that task would then be executing its own stream of instructions as well
  - But that’s a second stream of instructions, so for those two threads to be running concurrently you need 2 CPUs because 1 CPU can only execute 1 stream of instructions at a time
  - Unless… “or maybe 2”
- “Or maybe 2” = Simultaneous Multi-Threading (SMT)
  - Make 1 CPU execute (sort of) 2 threads simultaneously

Modern Superscalar Processors

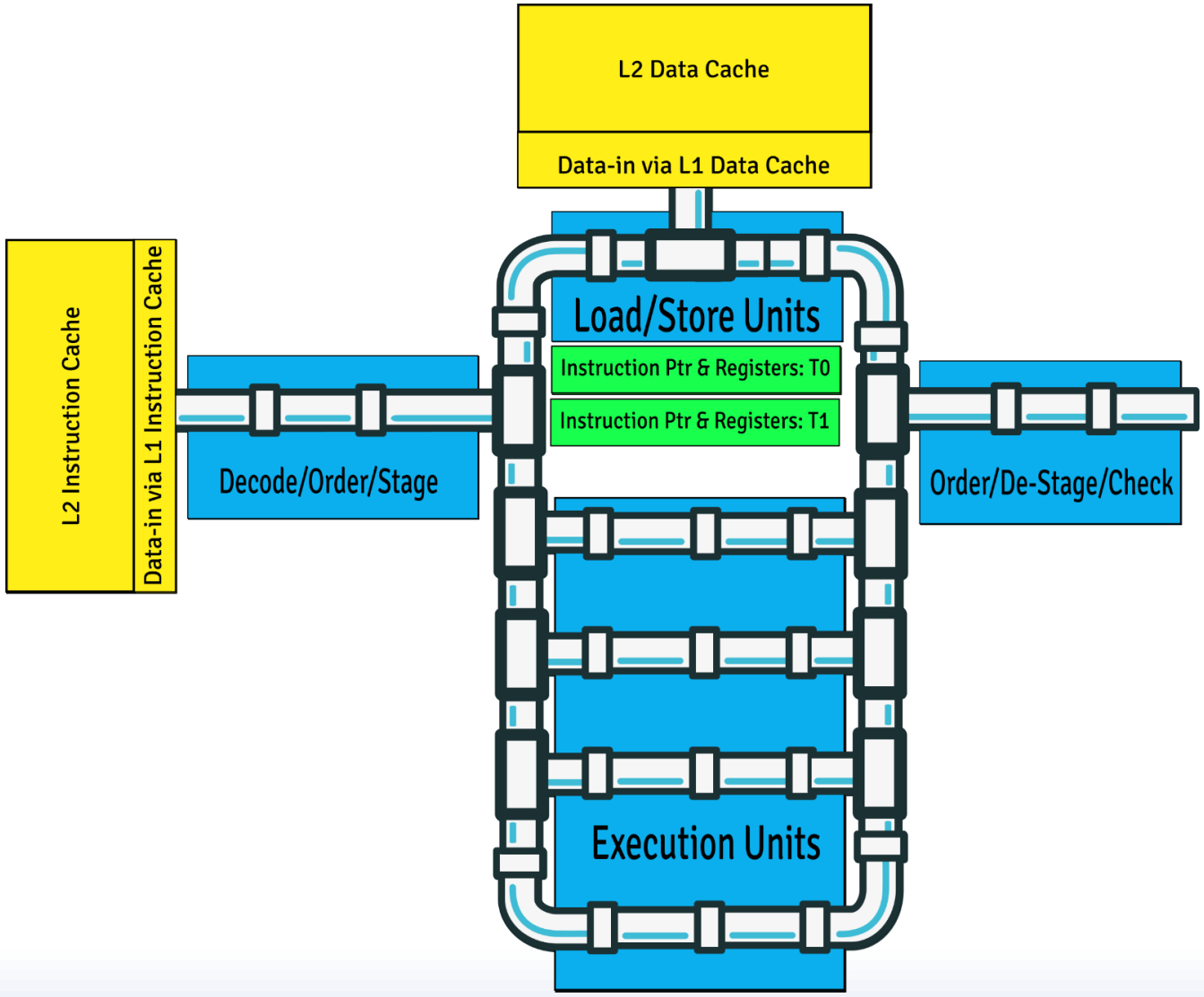
- Modern processors are complicated
- Multiple instructions in-flight (being executed) in the pipeline at any given time
  - But all these instructions come from the same instruction stream
- Aim: 1+ instruction finished / cycle
- Lots of things can cause pipeline stalls
  - L1 / TLB misses
  - Branch prediction misses
  - Data dependencies
  - Long instructions
- When stalled, there are parts of the chip idle, not doing work
  - Out-of-order execution help avoid pipeline stalls



Modern Superscalar Processors w/ SMT

- Simultaneous MultiThreading (SMT) is a way to further improve efficiency
- Second instruction stream (thread) with separate registers and instruction pointer
- Each thread processes independently and can be processing while other thread is waiting on L1 miss (e.g.)
- But the threads will contend for common resources (everything but the registers)
- So individual threads will run slower due to contention from the other thread
  - But more work done in total (hopefully)

- Note that a CPU with SMT enabled is not 2 physical CPUs, it's 1 CPU that can be executing 2 instruction streams simultaneously



Where can SMT be used?

- z13 and later machines
- IFLs (Optional)
- zIIPs (Optional)
- SAPs (Not Optional, z14 and later)

- Whether you should use it is a larger question!
  - https://www.pivotor.com/library/content/Chapman_SMT_MeasureDecide_201811_GSEUK.pdf

Moving to “logical” CPUs…

Hypervisors

- A Hypervisor is software or firmware that allows multiple instances of operating systems to share a given host machine
  - Pioneered by IBM in the late 60s/early 70s
  - Provides a virtual hardware abstraction layer
  - Today available on all platforms and most server systems run in a Hypervisor
    - Mainframe examples: PR/SM, z/VM, KVM
    - Other examples: KVM, Xen, Hyper-V, Vmware, VirtualBox, Parallels, MS Virtual Server
- All z/OS systems today run under some sort of HperVisor
  - Almost exclusively PR/SM
  - Rarely, multiple levels: PR/SM -> z/VM -> z/OS
- What the operating system sees as CPUs are really “logical” or “virtual” CPUs

LPARs

- PR/SM (Processor Resource/Systems Manager) enables and manages LPARs (Logical Partitions) on the mainframe
- Modern machines require LPARs (as opposed to old “Basic” mode)
- PR/SM works with cores, but the operating system sees CPUs
  - When z/OS has zIIP SMT active it will see two zIIPs per zIIP assigned to the LPAR
- LPARs can use dedicated or shared cores/CPUs
  - Dedicated CPs are assigned to a physical CPs for the exclusive use of that LPAR
    - Total number of dedicated CPs assigned to all activated LPARs can’t be greater than the number of CPs you’ve purchased
  - Shared CPs does not get exclusive use of the physical CP to a specific LPAR
    - Each LPAR can only have active CPs <= purchased number of CPs (less any dedicated ones)
    - Total shared CPs across all LPARs can be greater than purchased CPs

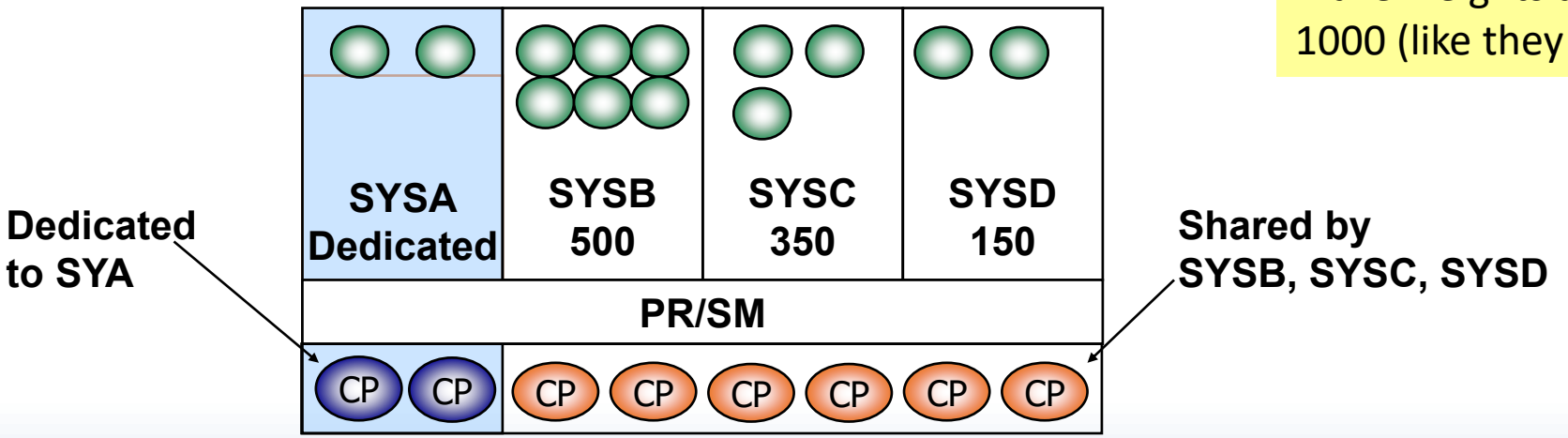
Defining LPARs

- On the HMC, you define for each LPAR a number of configuration values such as:
  - Number of processors (for each type of processor assigned to the LPAR)
    - Reserved processors can be defined to allow for future non-disruptive addition of processors
  - LPAR weight settings (for each type of processor assigned to the LPAR)
  - Capping setting
  - Central Storage (memory)
  - Various other controls that we'll be less concerned about today

LPAR Weight Enforcement

- When there is contention for the shared physical processors, PR/SM enforces the weights assigned to each partition
- Each LPAR with shared CPs is guaranteed to get at least its share of the shared CPs
  - $LPAR\ Share = 100\ * \frac{LPAR\ Weight}{\sum Weight\ of\ activated\ LPARS}$
- In below example:
  - SYSB – guaranteed 50% of shared physical processors
  - SYSC – guaranteed 35% of shared physical processors
  - SYSD – guaranteed 15% of shared physical processors

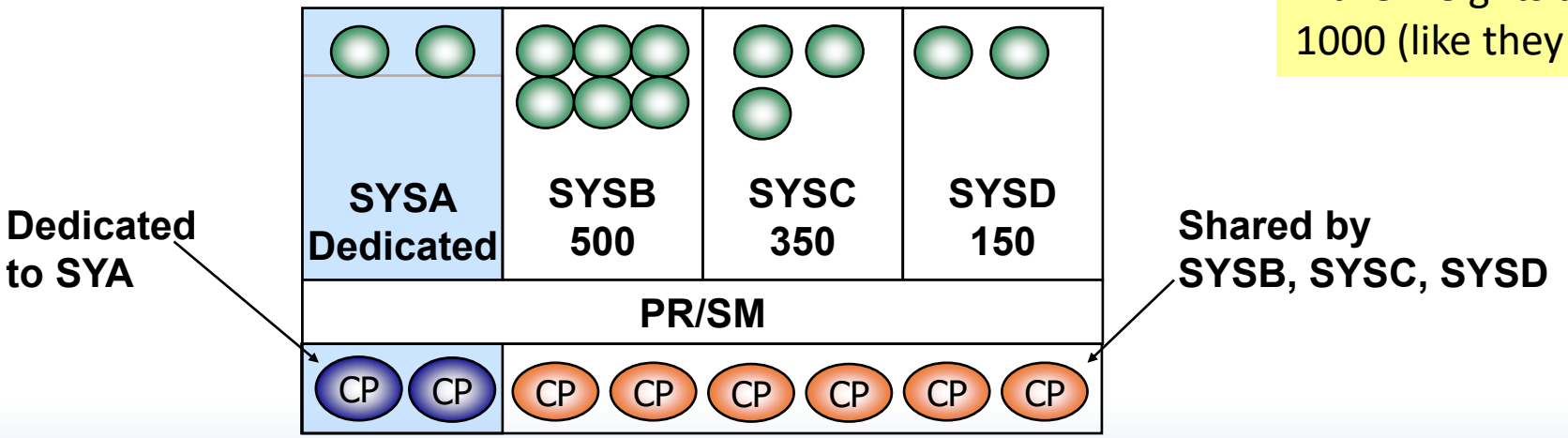
For ease of use, try to make weights add up to 1000 (like they do here).



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

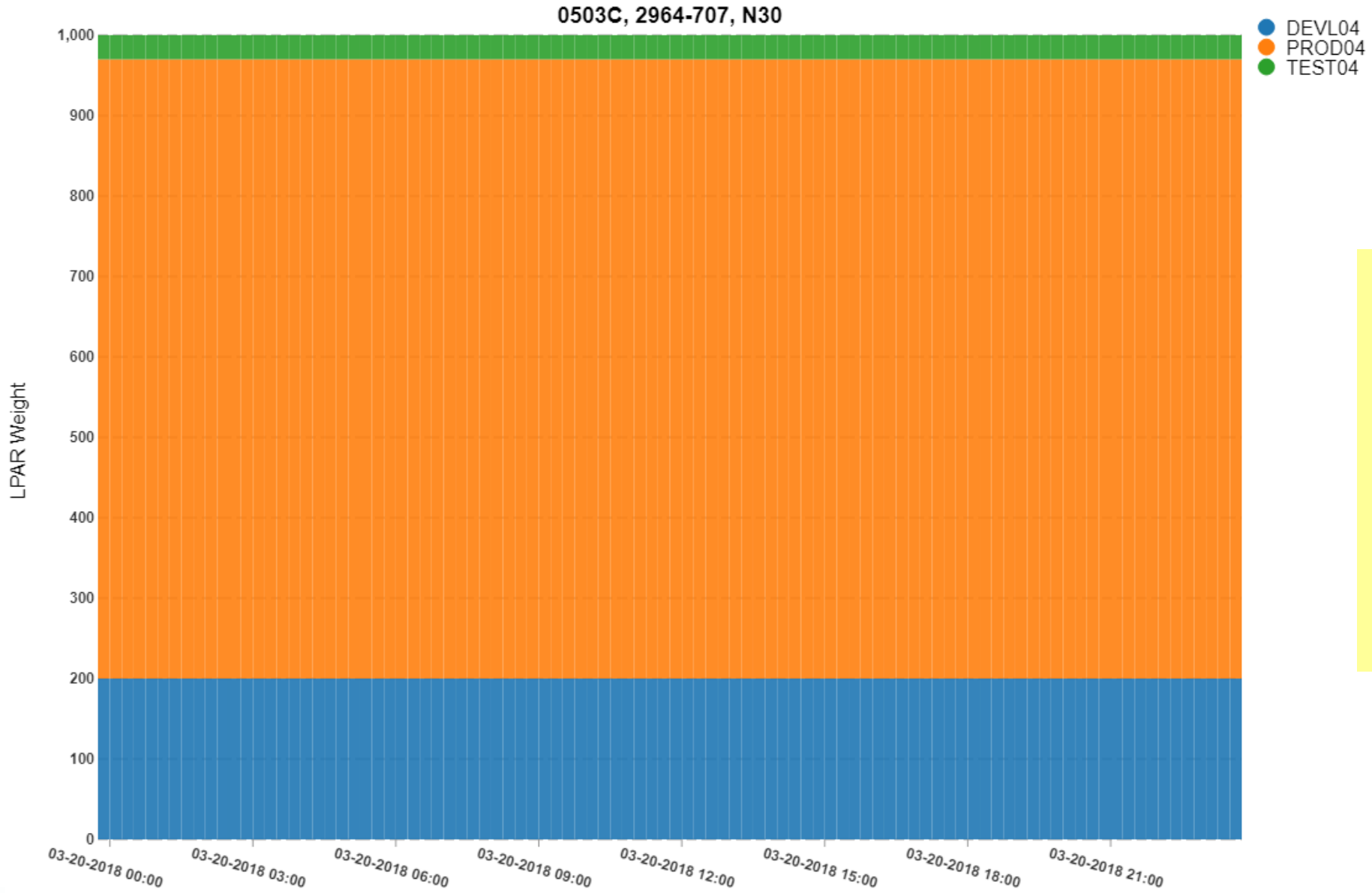
For ease of use, try to make weights add up to 1000 (like they do here).



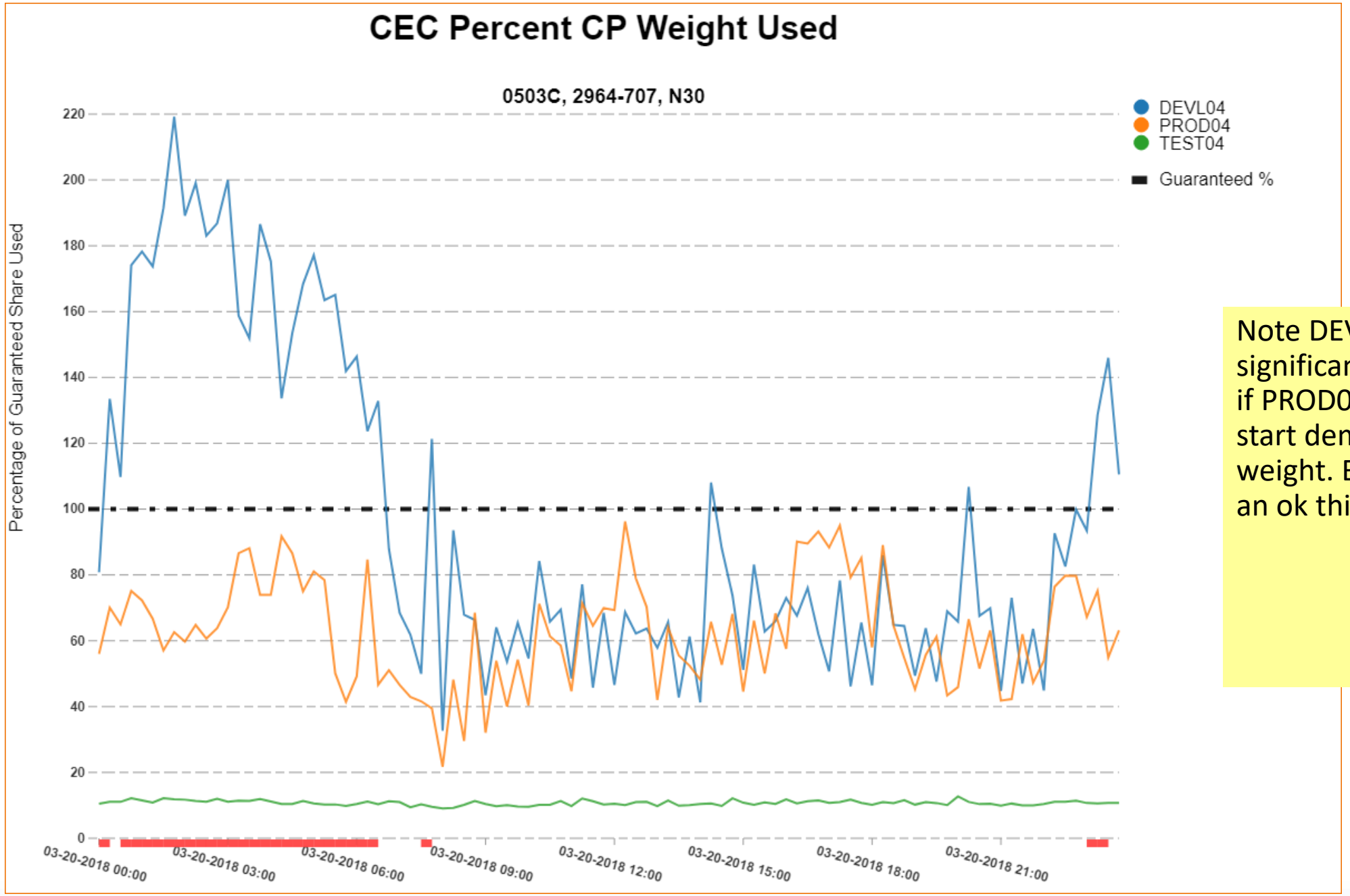
Weighty Issues

- Generally speaking, you should give the LPAR a weight that reflects the capacity that the LPAR needs to get its work done
  - Giving too little weight runs the risk that the LPAR won't get enough capacity when it needs it
  - Giving too much weight runs the risk some other LPAR won't be able to get to the capacity that it needs
- If an LPAR is regularly using more than its weight, understand that it is at risk of being constrained if the other LPARs demand their weight
- Assign no more than a couple of CPs more than the number of CPs guaranteed by the weight
- Usually unimportant trivia: PR/SM weight management generally accurate to +/- 3.6% of the weight

CEC Assigned CP LPAR Weights



DEVL04 has 20% of the machine, PROD04 77% and TEST04 3%. Note the weights nicely add to 1000 here.



Note DEVL04 is at significant risk overnight if PROD04 and TEST04 start demanding their weight. But maybe that's an ok thing in this case.

PR/SM Dispatching

- PR/SM is responsible for time-slicing physical CPs amongst the LPARs
- Vertical High CPs will generally get a time slice of 100ms
  - And get quasi-dedicated physical CP
  - Unless the LPAR doesn't have demand for it's full time slice, in which case PR/SM may use the physical CP for some other LPAR's needs
- Vertical Medium/Low CPs will get a time slice of 12.5-25ms (usually 12.5)
- While a physical CP is dispatched to an LPAR's logical CP, no other LPAR can use the physical CP
- How many time slices PR/SM gives to the LPARs determined by the LPARs' relative weights
  - And most sites allow LPARs to borrow unused capacity from other LPARs

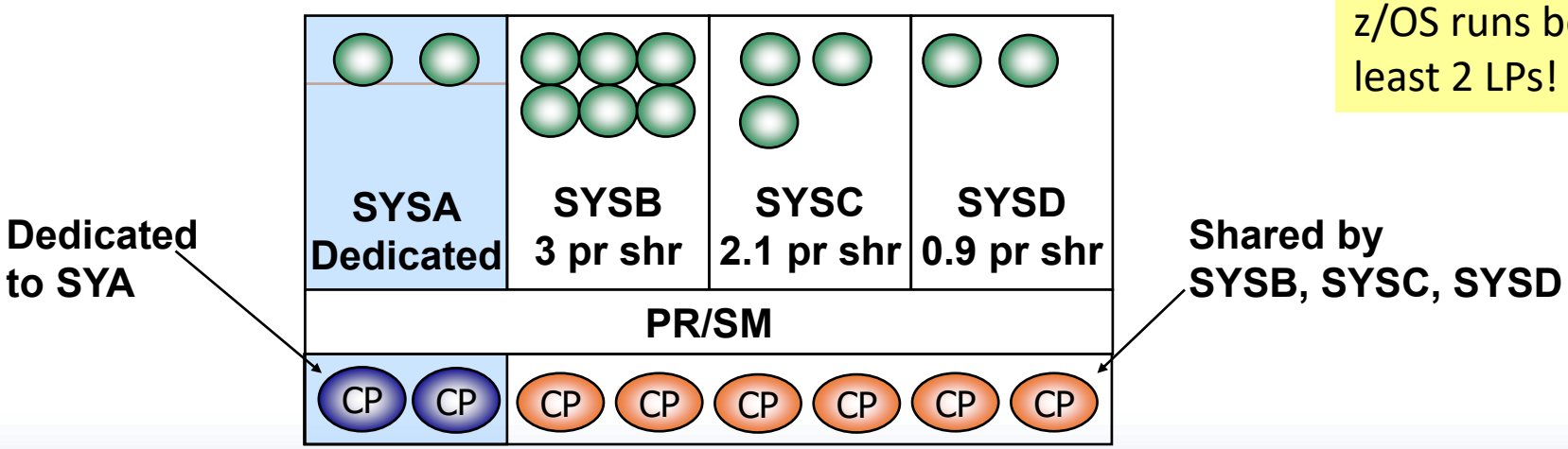
HiperDispatch

Horizontal CP Management (old)

- Cache effectiveness will be better when a unit of work is redispatched on the same physical CPU that it was last on
- But 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"

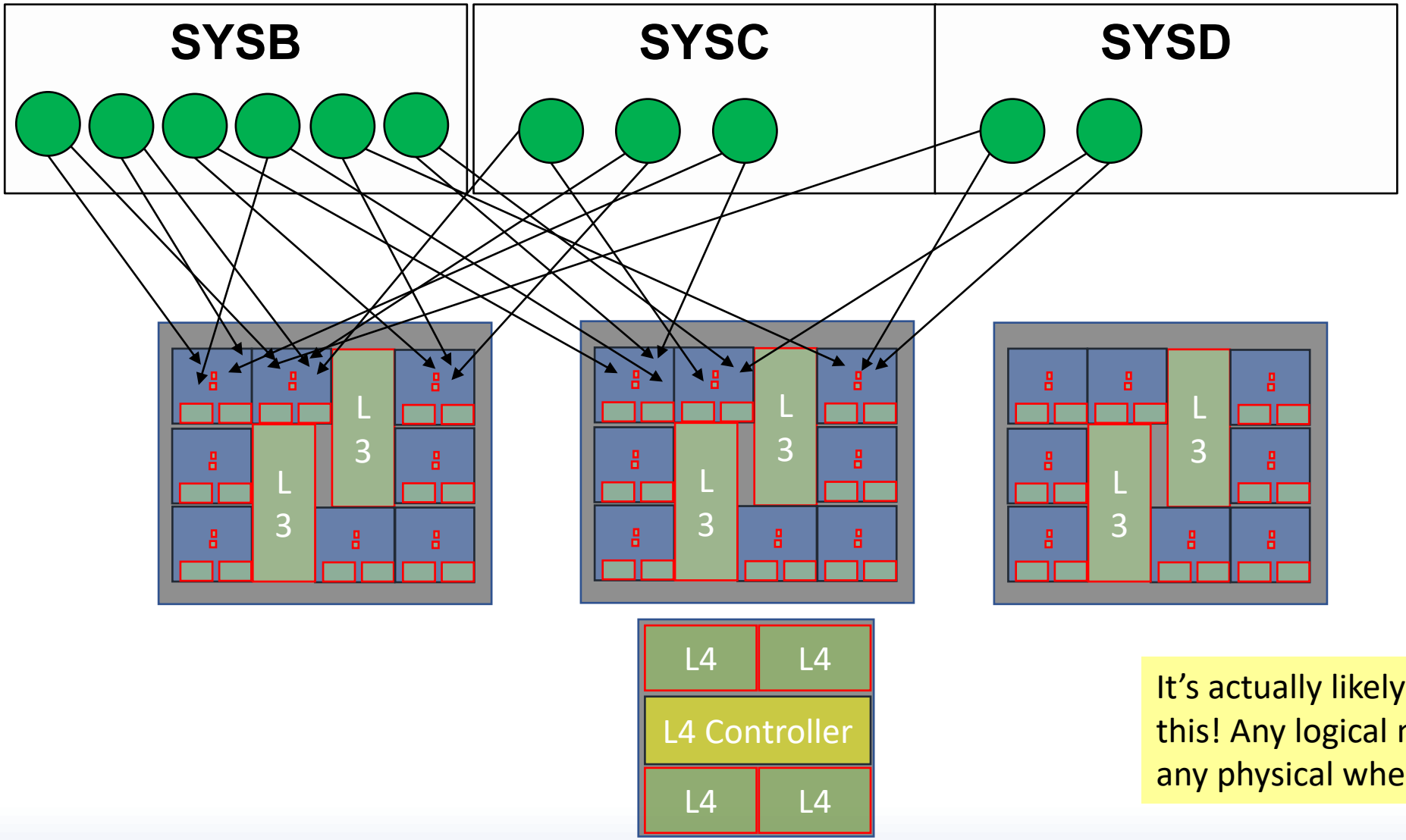
z/OS runs better with at least 2 LPs!



Vertical CP Management (not so old)

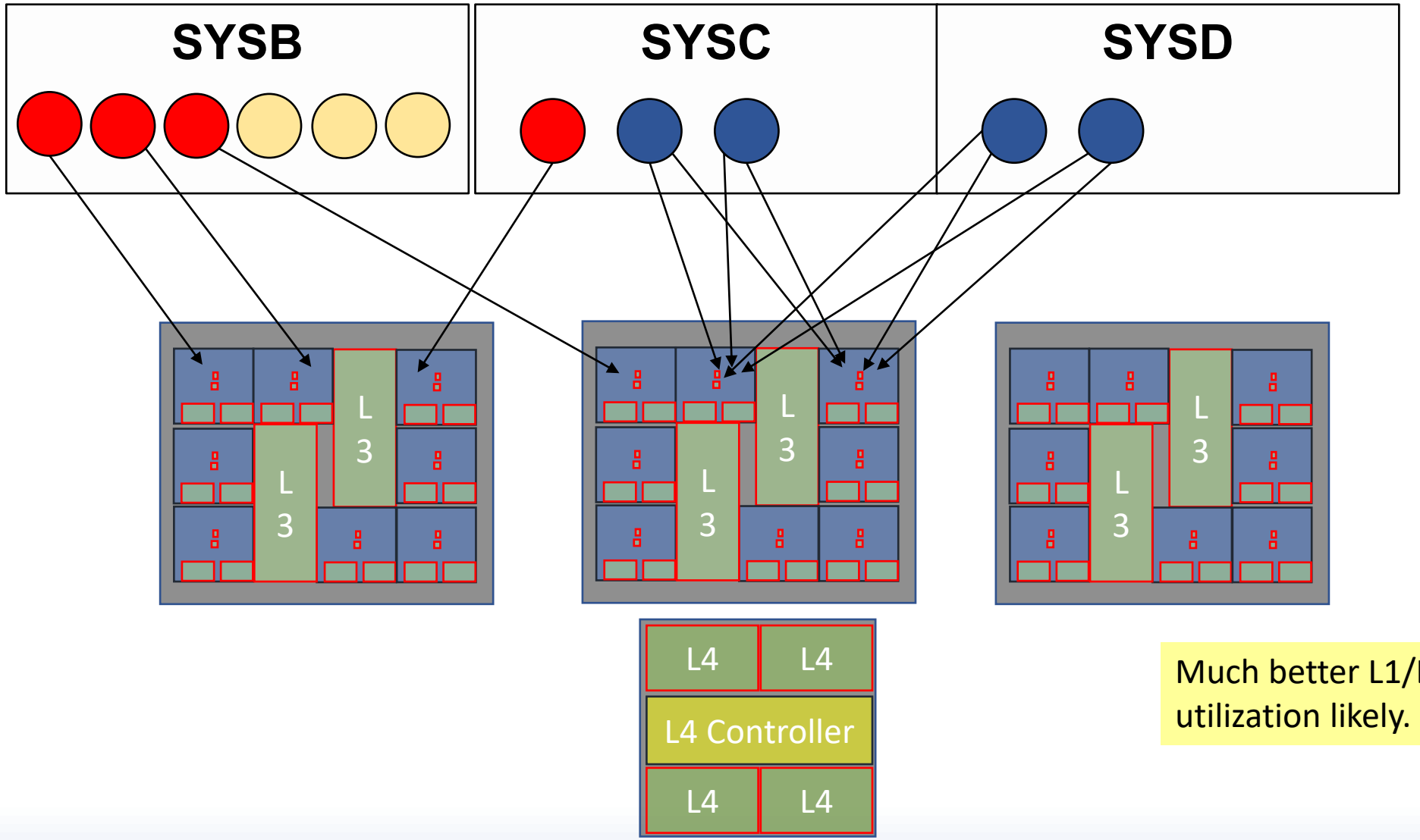
- HiperDispatch manages CPs “vertically”, meaning it endeavors to make the logical CPs a larger percentage of a physical
- Logical processors classified as:
  - High – The processor is essentially dedicated to the LPAR (100% share)
  - Medium – Share between 0% and 100%
  - Low – Unneeded to satisfy 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

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.

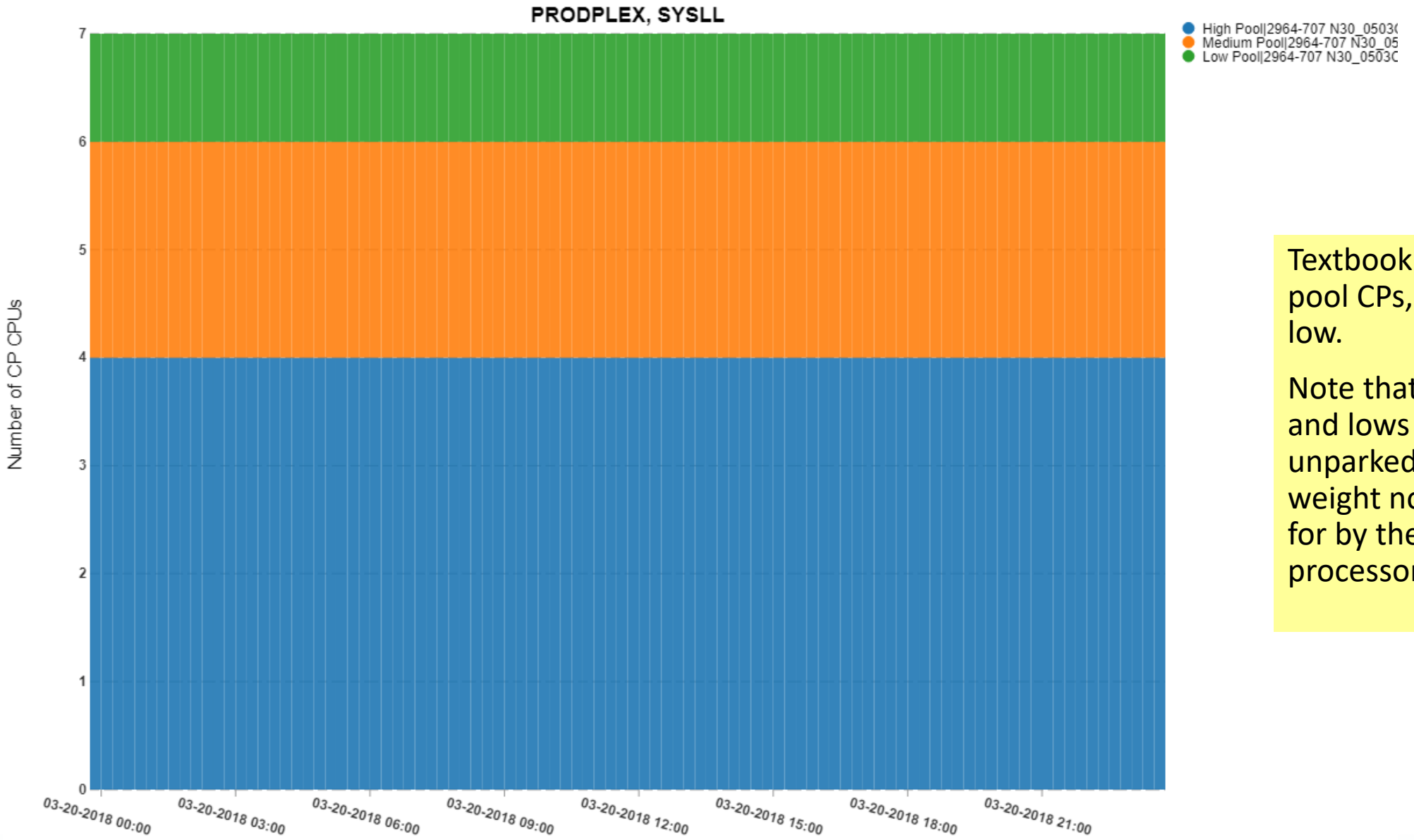
HiperDispatch: optional?

- IEAOPTxx HIPERDISPATCH option controls whether or not HiperDispatch is enabled
- Generally (95%+) it is a very good thing to have enabled
- HiperDispatch defaults to “YES” with z/OS 1.13 and a z196 or later processor
- An LPAR has 64 or more logical processors automatically forces YES
  - Even if you specify “NO”, LPAR will be managed in HD mode
  - You probably don’t want an LPAR to have 64 logical processors though!
- Use of SMT requires YES
- Fundamentally, HiperDispatch reduces MP overhead (contention) by:
  - Essentially dedicating high-pool CPs to LPARs
  - Reducing the number of CPs work is spread across by parking low-pool CPs

How many highs?

- How many CPs worth of capacity is the LPAR?
  - E.g. 2.4, 3.8
- Whole number is number of highs
  - Unless the decimal is < 0.7, in which case take away 1 high to make 2 mediums
- For example:
  - 2.4 = 1 High, 2 mediums each at a 70% share
  - 3.8 = 3 Highs, 1 medium at 80% share
- Generally High pool processors are the most efficient
- Sometimes can get an extra high by adding weight to convert 1 of 2 mediums to a high because the remainder will be >70%
  - E.G. if an LPAR had a weight giving it 3.65 bumping it to 3.71 might be beneficial

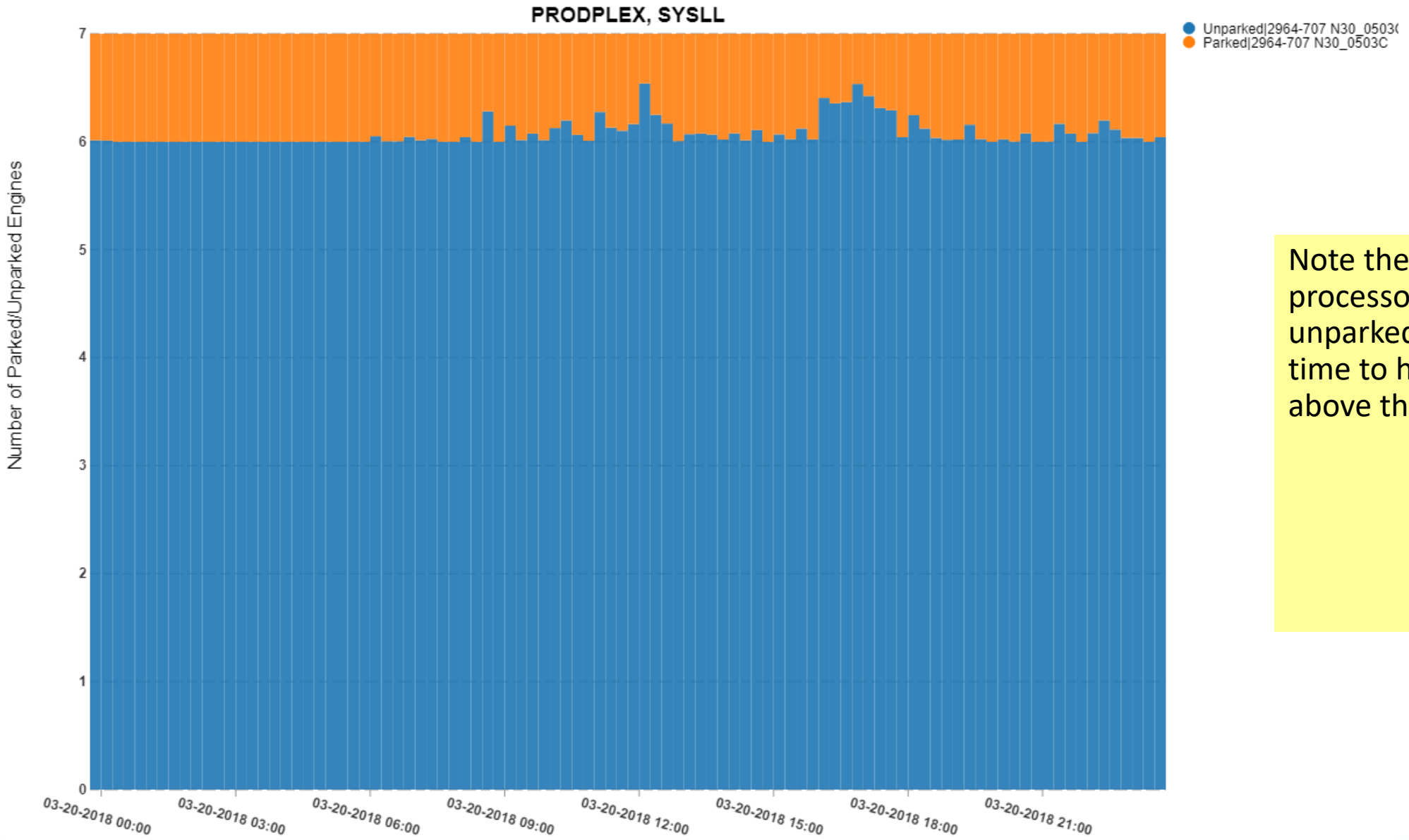
HiperDispatch CP CPU Pooling at End of Interval



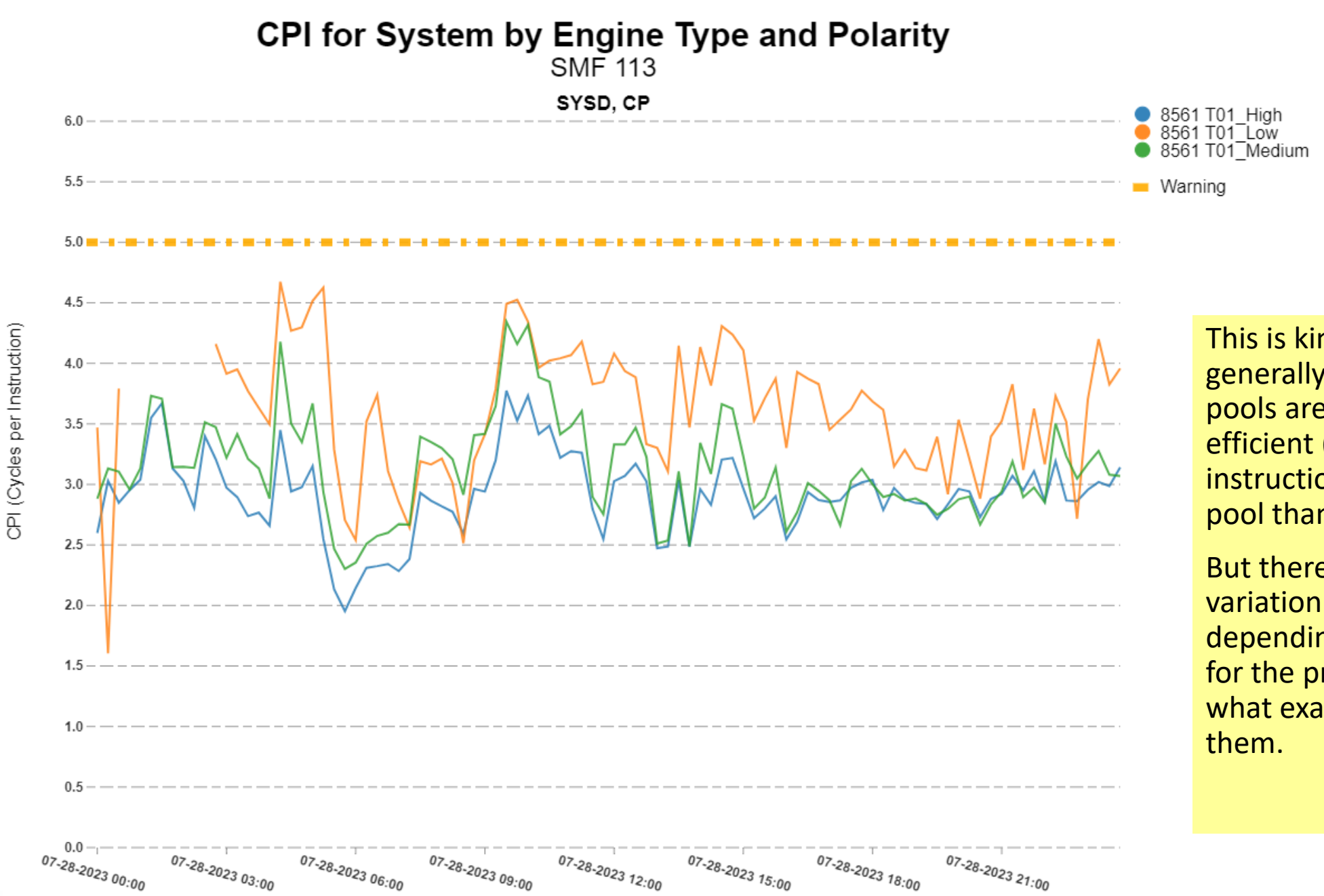
Textbook case: 4 high pool CPs, 2 medium, 1 low.

Note that the mediums and lows (when unparked) will share the weight not accounted for by the 4 high pool processors.

HiperDispatch - Parked / Unparked CPs



Note the low pool processor getting unparked for periods of time to handle work above the LPAR's weight.



This is kind of what we generally expect: High pools are a bit more efficient (fewer cycles per instruction) than Medium pool than Low pool.

But there's a lot of variation in here depending on the demand for the processors and what exactly is running on them.

z/OS Dispatching

Two Important z/OS Dispatching Concepts

- Reduced Preemption
  - When a work unit is given the processor, it will be allowed to keep it for some minimal amount of time, even if a higher-priority piece of work comes ready
  - Improves CPU cache efficiency, especially for lower-importance work
  - This is the minor time slice
- Fair-share dispatching
  - A work unit is only allowed to run on a processor for a period of time before it is forced off the processor and goes to the back of the queue for its dispatching priority
  - Allows all work at a given dispatching priority to have access to the CPU
  - This is the major time slice

Responsibilities

- PR/SM is responsible for dispatching physical CPUs to LPARs in relation to the LPARs' weights

- z/OS is responsible for dispatching work to logical CPs based on the work units' dispatching priorities which are set based on the goals and relative importance defined for the work

Note that during the first 50µs LPAR 2 had no CPs available to it and only 1 CP during the second 50µs

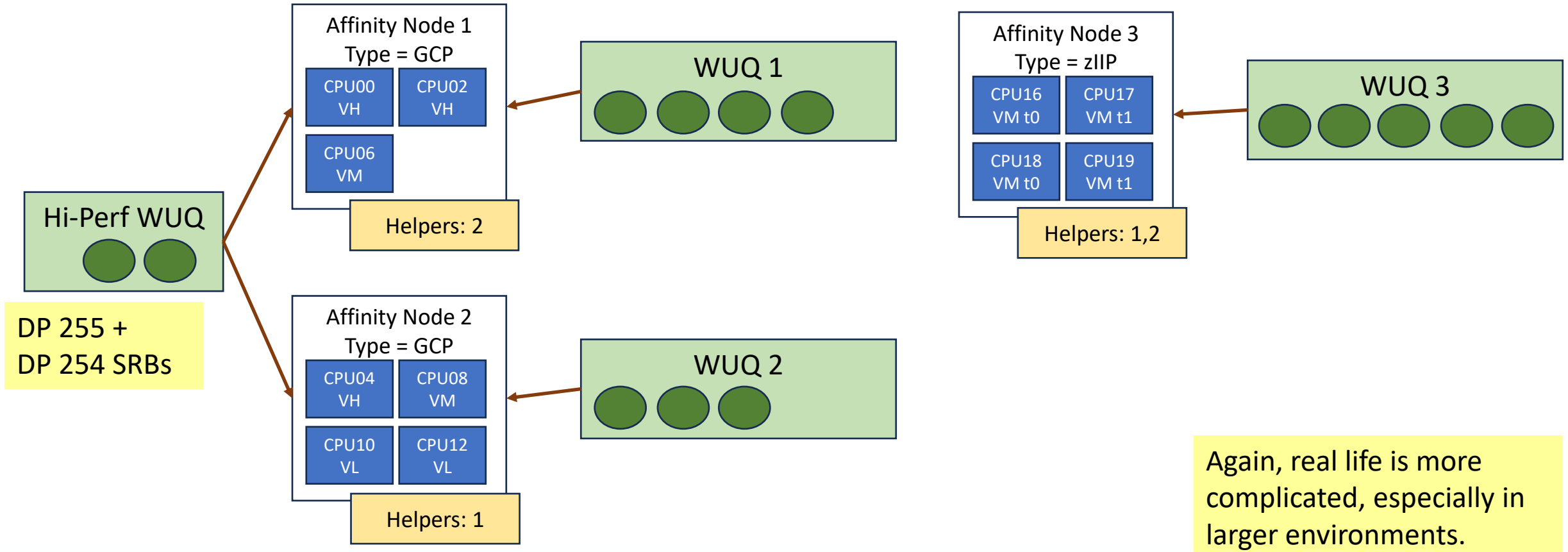
Real life is a bit more complicated

More z/OS Responsibilities

- Parking/unparking low pool processor
  - Based on demand from the LPAR as well as availability from other LPARs
- Running zIIPs in SMT enabled mode
  - PR/SM just dispatches the zIIP core to z/OS, z/OS decides whether to dispatch 1 or 2 threads to the core
  - Note too that z/OS densely packs the cores in part because PR/SM is moving cores between LPARs
- Managing CPU Affinity Nodes (groups of CPs to dispatch work to)
  - Improves cache effectiveness by trying to get work back to (maybe) the same CPU it was on before (in conjunction with PR/SM trying to get the logicals back to the same physicals)
  - Affinity nodes also have “helper” nodes that come into play at times
- Recording CPU time

Affinity nodes

- z/OS groups CPUs to “affinity nodes”, each with their own work unit queue



Again, real life is more complicated, especially in larger environments.

CPU Time Measurements

- CPU time = total time that a CPU has spent performing work for task
  - Time that a workload is dispatched to a CPU
- z/Architecture provides CPU timers with nominal resolution of 1 µs
  - Although the times aren't usually externalized to that precision
- When a CPU is interrupted to process something else, the CPU timer is readjusted once the interrupted task is dispatched again
- Across all your LPARs, you can't consume more CPU time per second than you have physical CPUs
  - And you'll actually record less in the SMF 30/72 records because some will be uncaptured

Summary

- Code runs on a physical CPU/core
- But in modern environments, the operating system most often sees “virtual” or “logical” CPUs
- PR/SM manages dispatching physical cores to LPARs
- z/OS manages dispatching work to those logical CPUs
- PR/SM, z/OS and HiperDispatch optimize cache utilization and therefore performance by cleverly managing the relationships between the work, logical CPUs, and physical cores

- Are you amazed it all works so well?

Your feedback is important!

SHARE
EDUCATE • NETWORK • INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation