Don’t keep your CPU waiting

Speed reading for machines

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

www.pivotor.com

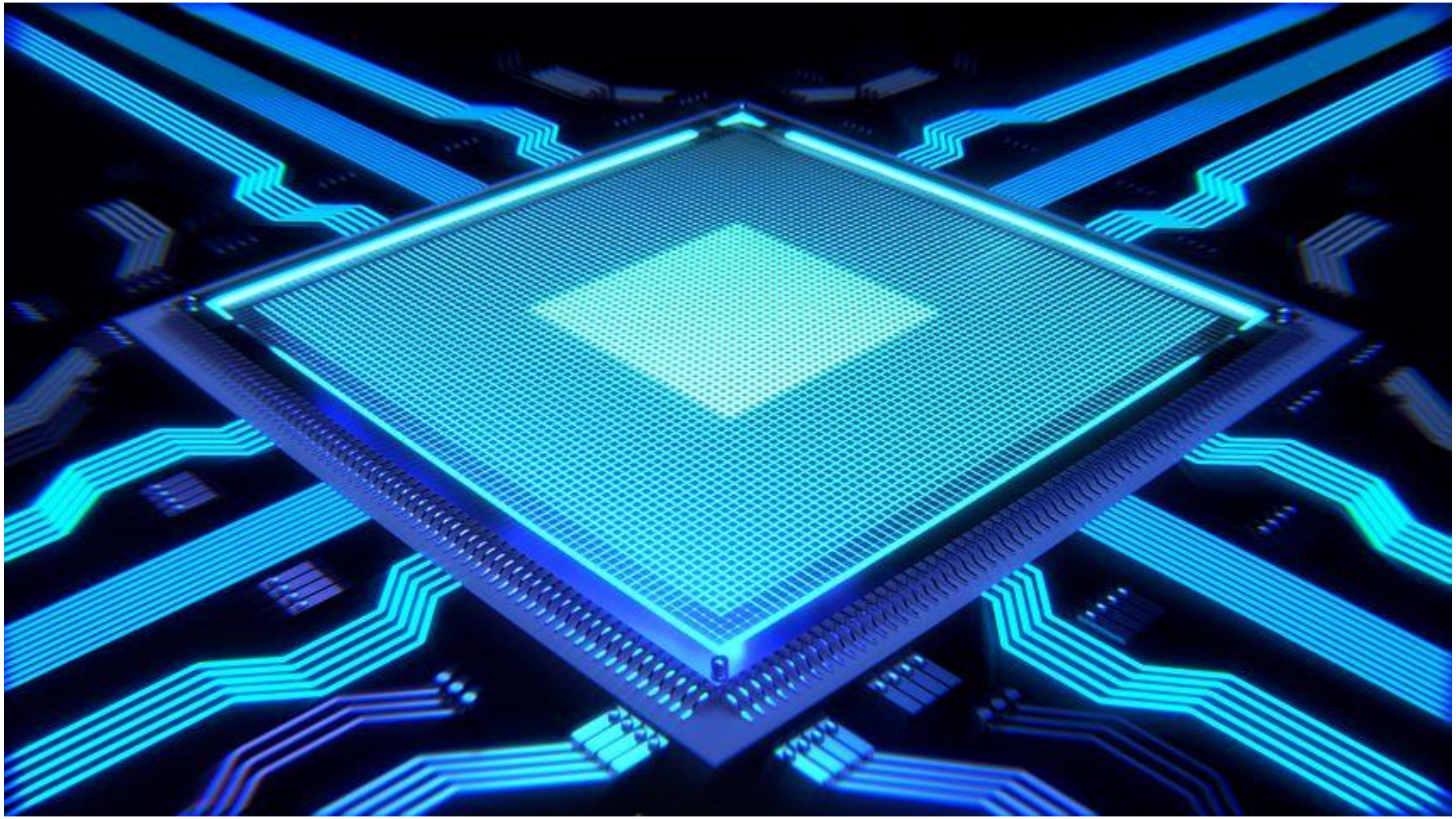
John.Baker@EPStrategies.com

Agenda

- Storage vs storage
- CPU and processor caches
- Storage tiers
  - Sync vs Async
- Basic cache principles
- Improving sync access
  - Hiperdispatch and DAT
- Improving async access
  - DSS stats
  - RMF stats
- zHyperLink

Storage vs Storage

All roads lead to the CPU



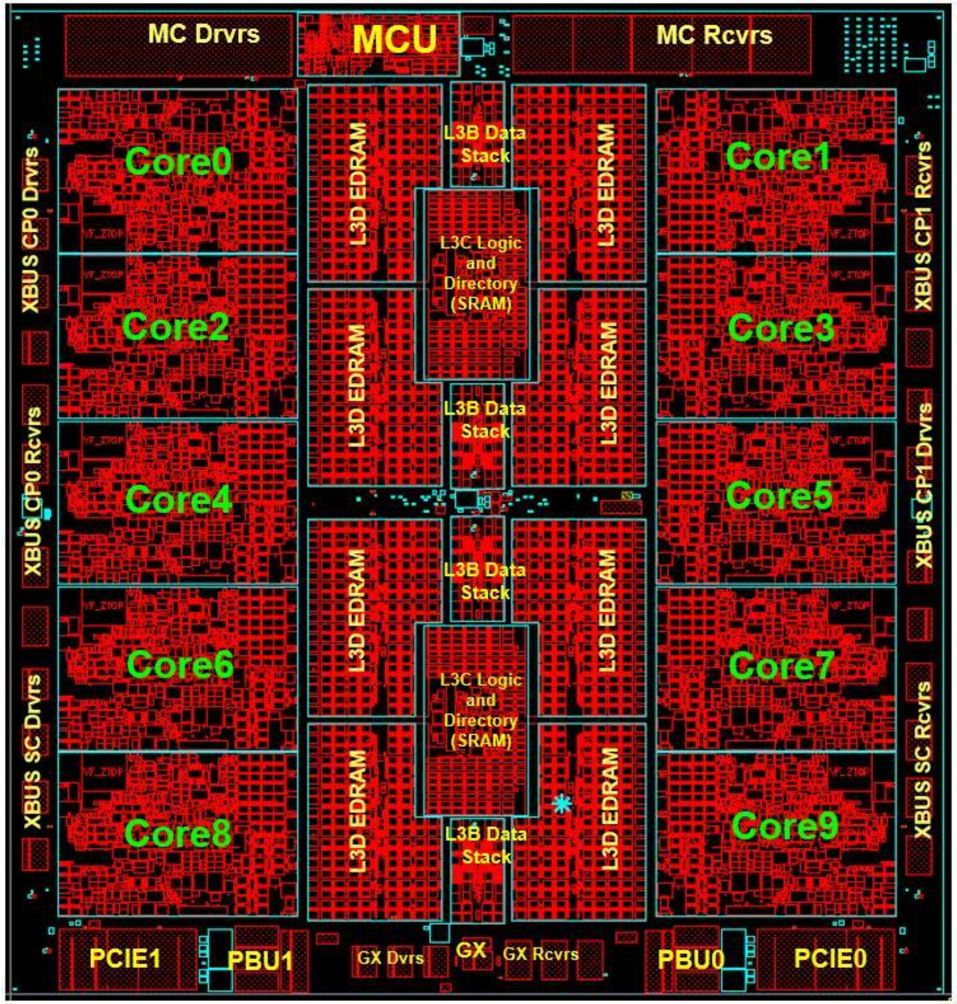
- z14: 5.2 GHz = .192 ns cycle time

How far can I go in .192 ns?

- Speed of light = 300,000 km / sec
- = 30 cm / ns
- 30cm ~ 12 inches * .192

= 2.3 inches

z14 PU Chip



- 14 nm technology
- 6.14 billion transistors
- 23.2 km or 14.4 miles of wire
- Up to 10 cores
- L3 cache on chip

http://www.redbooks.ibm.com/abstracts/sg248451.html?Open

Processor Caches

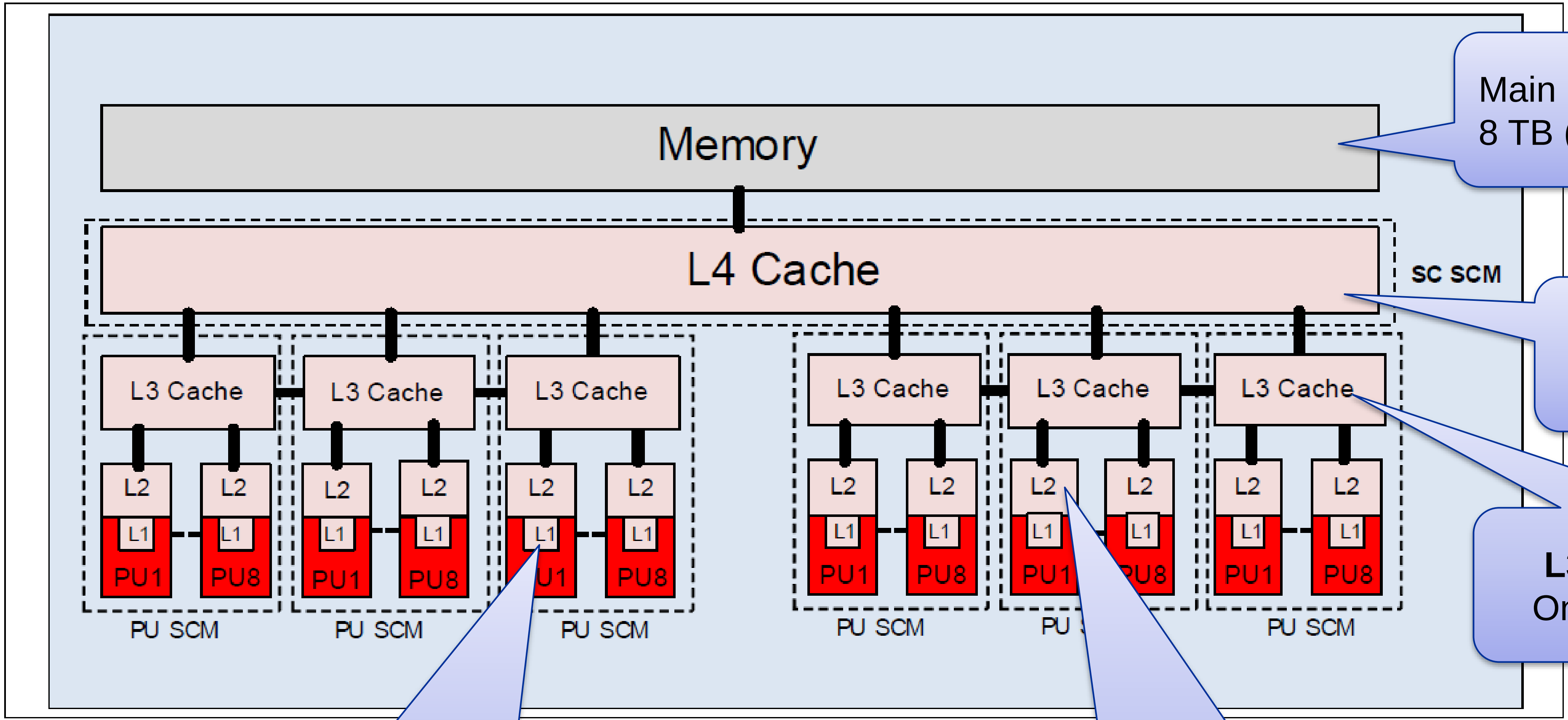


Figure 12-2 Memory hi a fully populated z14 CPC drawer

Main Memory:
8 TB (multiple DIMMs) per drawer

L4: 672 MB per drawer

L3: 64 MB on SCM chip
One shared per 10 Cores

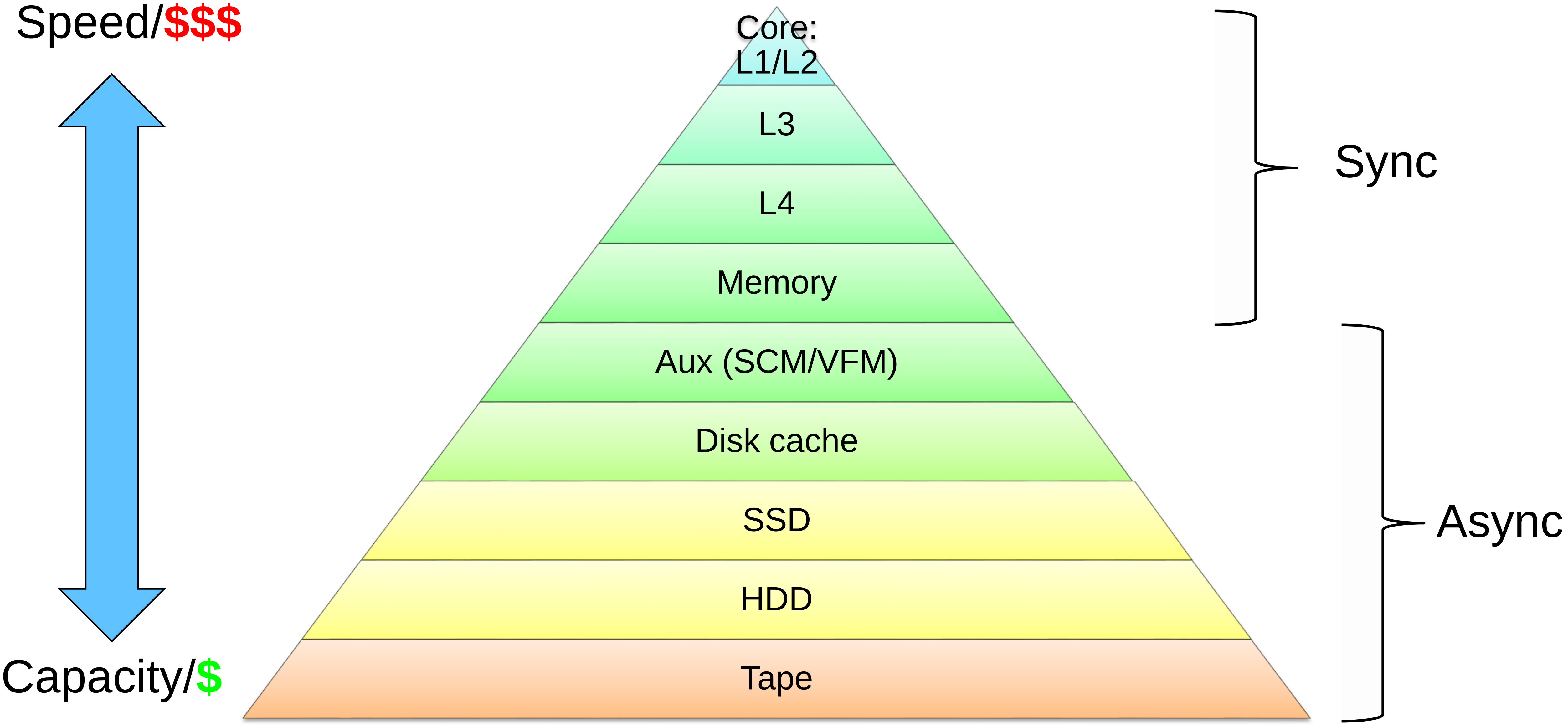
L1: On-Core

- 128 KB Instruction, 128KB Data

L2: On-Core

- 2 MB Instruction, 4MB Data

Storage Tiers



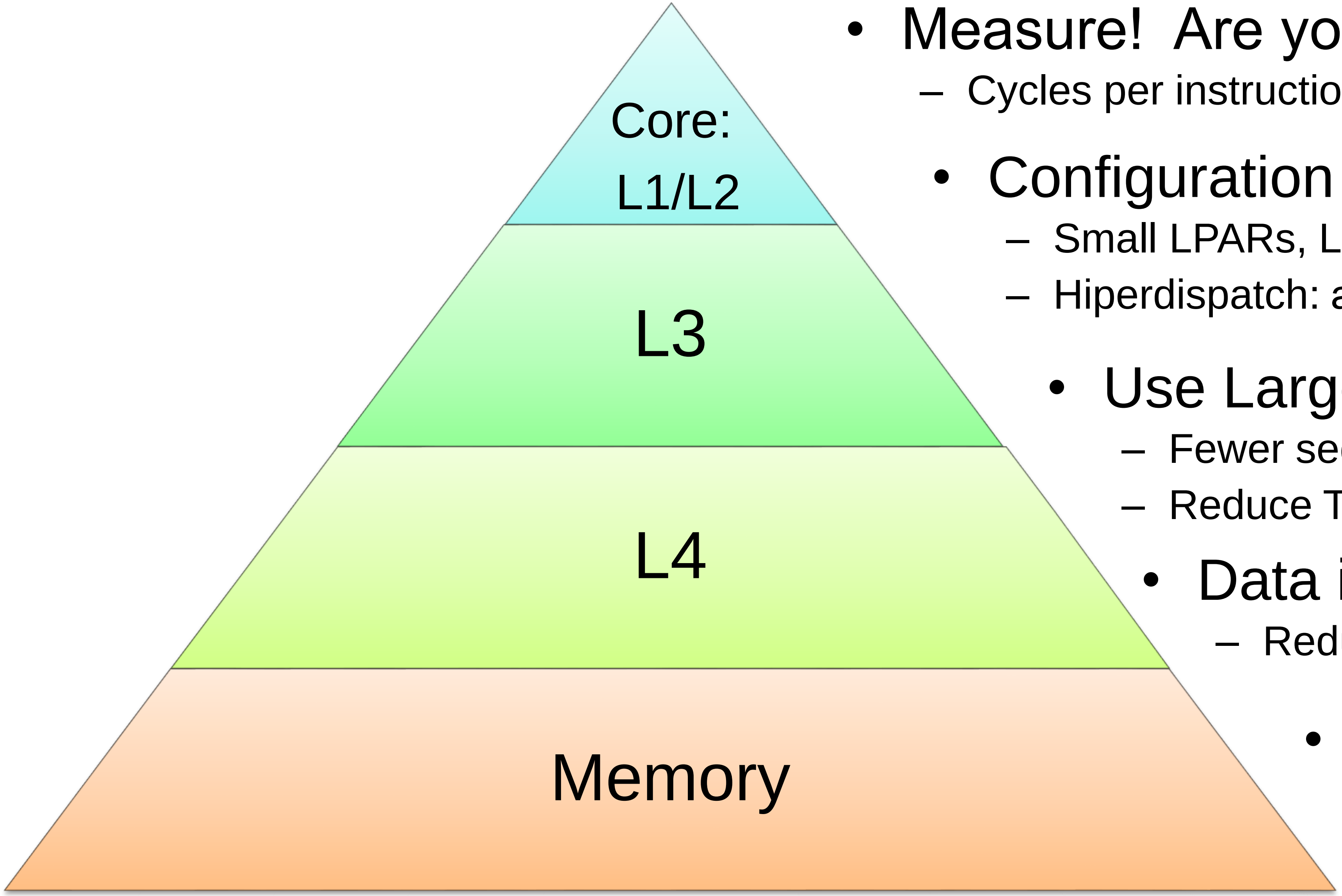
Caching Principles

- Temporal locality: Data that is accessed once, will likely be accessed again
  - Try to hold data in cache as long as possible (LRU)
  - Larger caches can hold more

- Spatial locality: Data near the same location as target data will likely be accessed as well
  - Cache lines and sequential prefetch

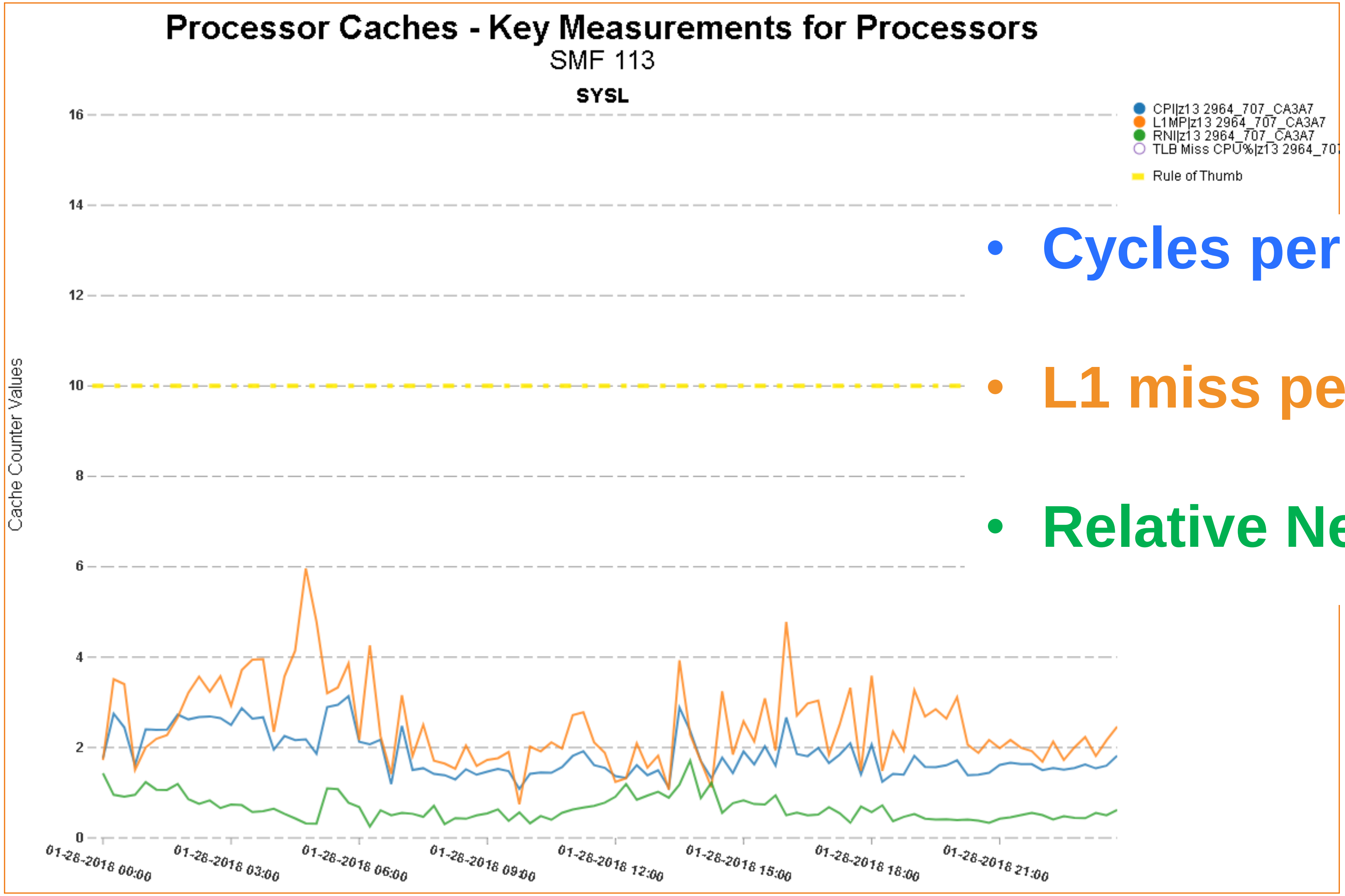
- CPU performance goal: Must be able to access L1 cache in a single clock cycle
  - L1 cache should not be too large to prevent this

Improving Sync Access



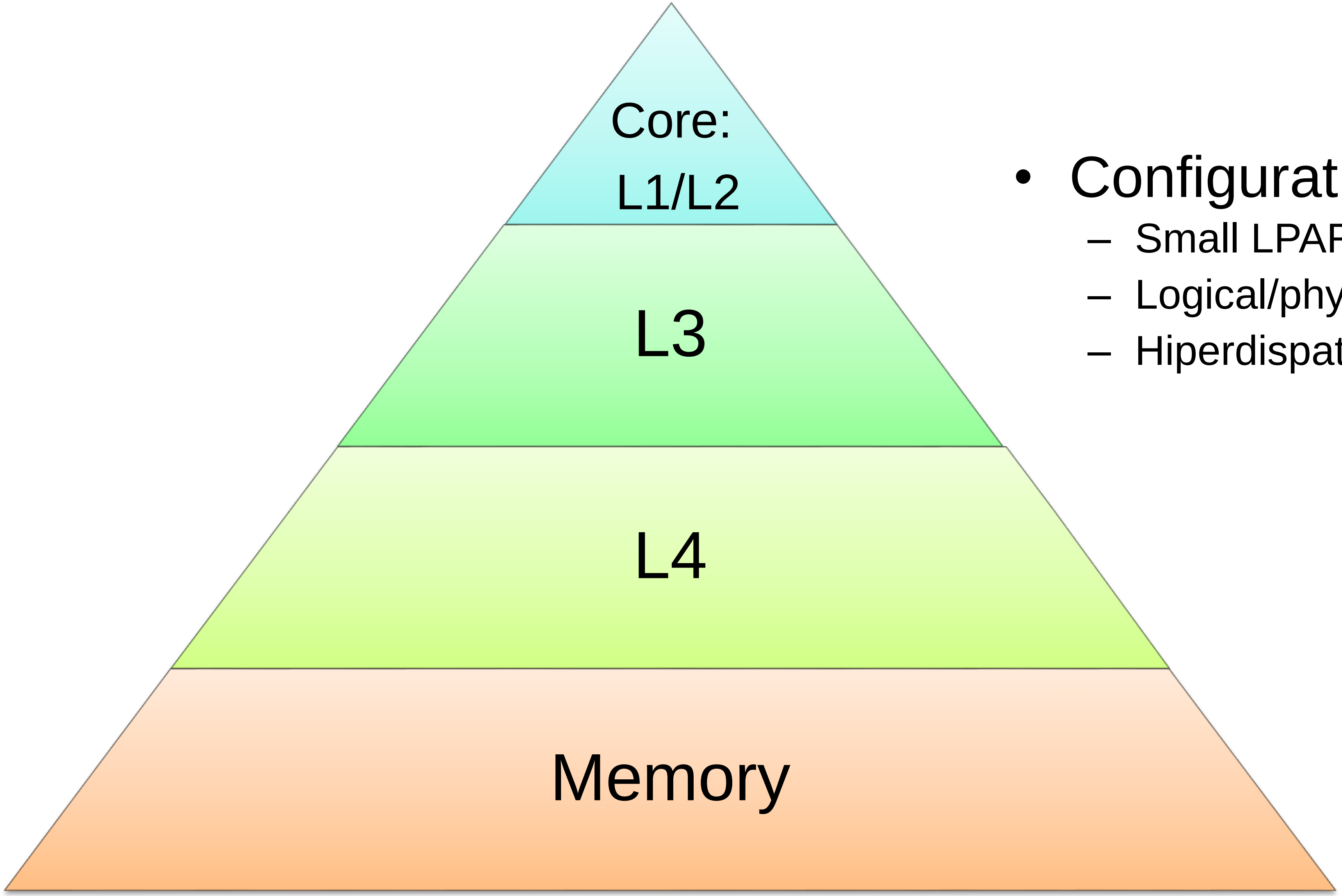
- Measure! Are you collecting 113's?
  - Cycles per instruction, L1 miss percentage, RNI
- Configuration
  - Small LPARs, Logical/physical CP ratio
  - Hiperdispatch: avoid Vertical Low CP's
- Use Large Frames for DB2
  - Fewer segment/page table entries
  - Reduce TLB misses
- Data in memory tech
  - Reduce cycles to fetch busy tables
- Applications
  - Use current compilers
  - Check your old assembler

Processor Cache Measurements



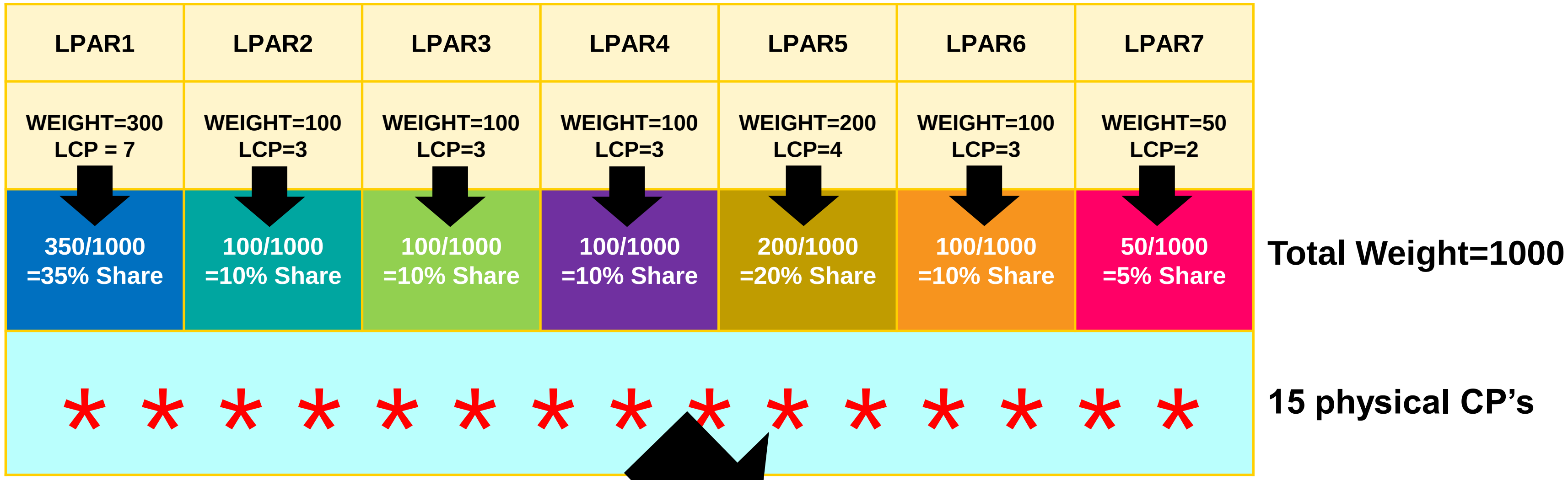
- **Cycles per instruction**
- **L1 miss percent**
- **Relative Nest Intensity (RNI)**

Improving Sync Access

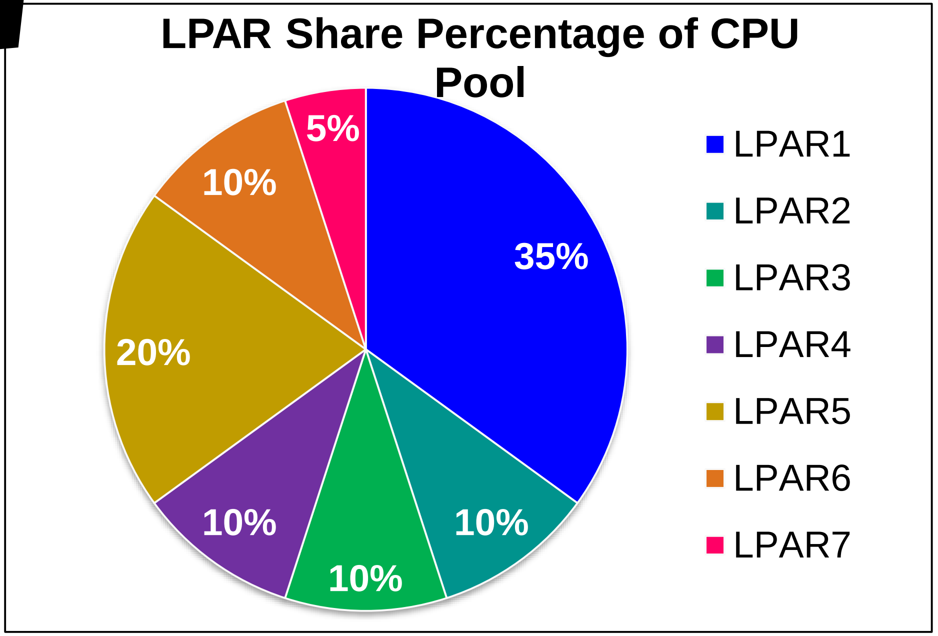


- Configuration
  - Small LPARs
  - Logical/physical CP ratio
  - Hiperdispatch: avoid Vertical Low CP’s

Pre-Hiperdispatch: “Horizontal” alignment



LPAR1	LPAR2	LPAR3	LPAR4	LPAR5	LPAR6	LPAR7	
WEIGHT=300 LCP = 7	WEIGHT=100 LCP=3	WEIGHT=100 LCP=3	WEIGHT=100 LCP=3	WEIGHT=200 LCP=4	WEIGHT=100 LCP=3	WEIGHT=50 LCP=2	
350/1000 =35% Share	100/1000 =10% Share	100/1000 =10% Share	100/1000 =10% Share	200/1000 =20% Share	100/1000 =10% Share	50/1000 =5% Share	Total Weight=1000
* * * * * * * * * * * * * * *							15 physical CP’s



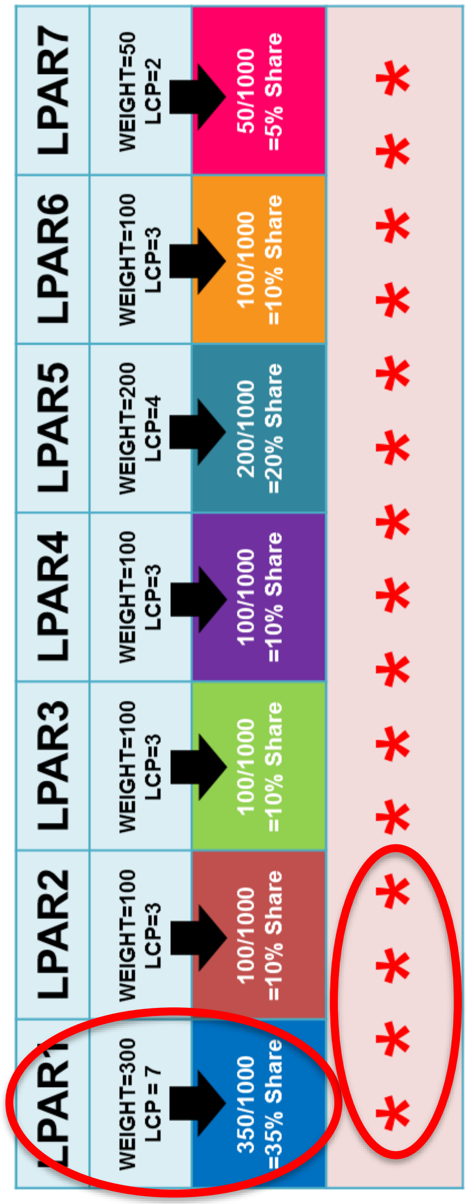
- Weight is guaranteed minimum
- May be exceeded if other LPARs do not use their full share
- Subject to number of LCPs (also limits maximum)

HiperDispatch

- A method of “aligning” physical CPUs with LPARs
- Goal is to take advantage of newer hardware design
- Also reduce multiprocessing overhead (“MP Effect”)
- Default as of z/OS 1.13

HIPERDISPATCH = NO	HIPERDISPATCH = YES
“Horizontal” alignment	“Vertical” alignment
% share distributed across all available physical CPUs	% share distributed by physical CP
All CPs distributed among all LPARs	LCPs become “Vertical High, Vertical Medium, Vertical Low”
Utilization of all CP’s tends to be even	Unused (Vertical Low) CPs are “parked”

Vertical Alignment



LPAR1 WEIGHT=300 LCP = 7 350/1000 =35% Share

LPAR2 WEIGHT=100 LCP=3 100/1000 =10% Share

LPAR3 WEIGHT=100 LCP=3 100/1000 =10% Share

LPAR4 WEIGHT=100 LCP=3 100/1000 =10% Share

LPAR5 WEIGHT=200 LCP=4 200/1000 =20% Share

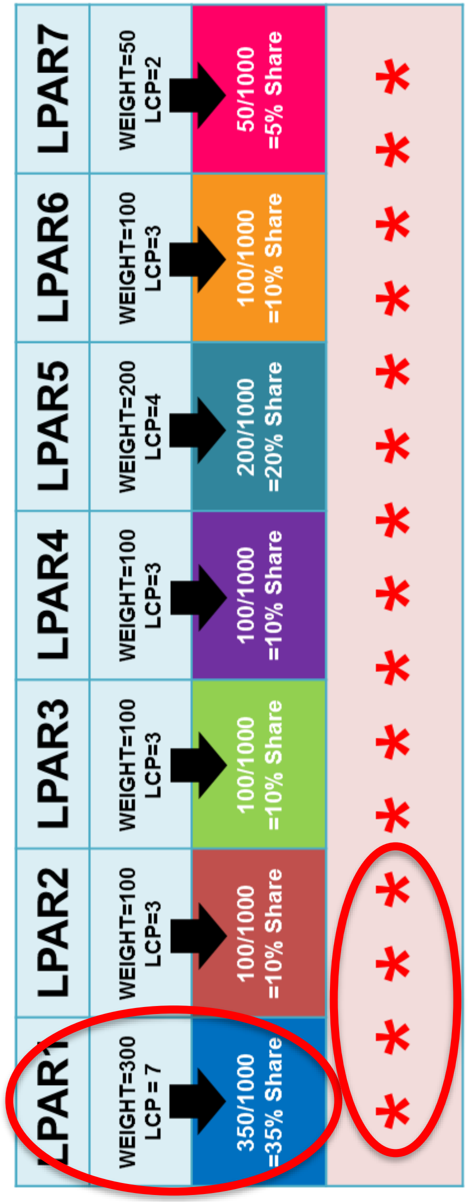
LPAR6 WEIGHT=100 LCP=3 100/1000 =10% Share

LPAR7 WEIGHT=50 LCP=2 50/1000 =5% Share

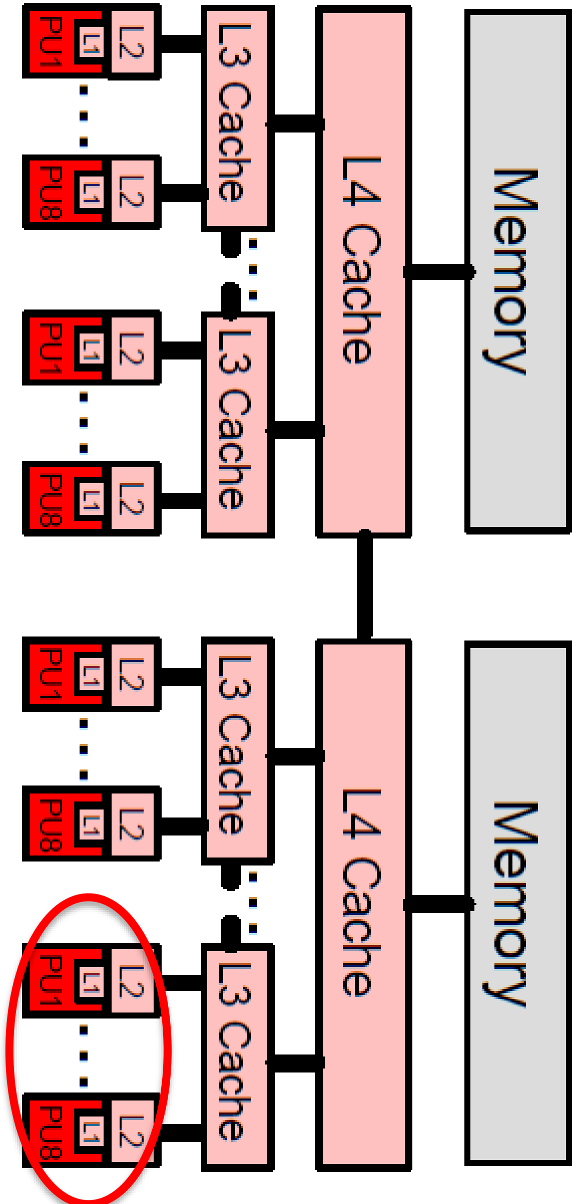
- LPAR1 example:
  - 35% share, 7 LCP's = .35 * 15 CP (Cores) = **5.25 CPs**
  - 1.0 = VH, .5 - .99 = VM, <.49 = VL *******
  - First every LPAR receives at least one VM = ".5 CPs"
  - 5.25 - .5 = 4.75
  - 4.75 - 4 (FOUR "whole CPs" = VH) = .75
  - .75 - .5 = .25 (one more VM)
  - .25 (less than .5) = VL
- = 4 * VH, 2* VM, 1* VL

https://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD106389

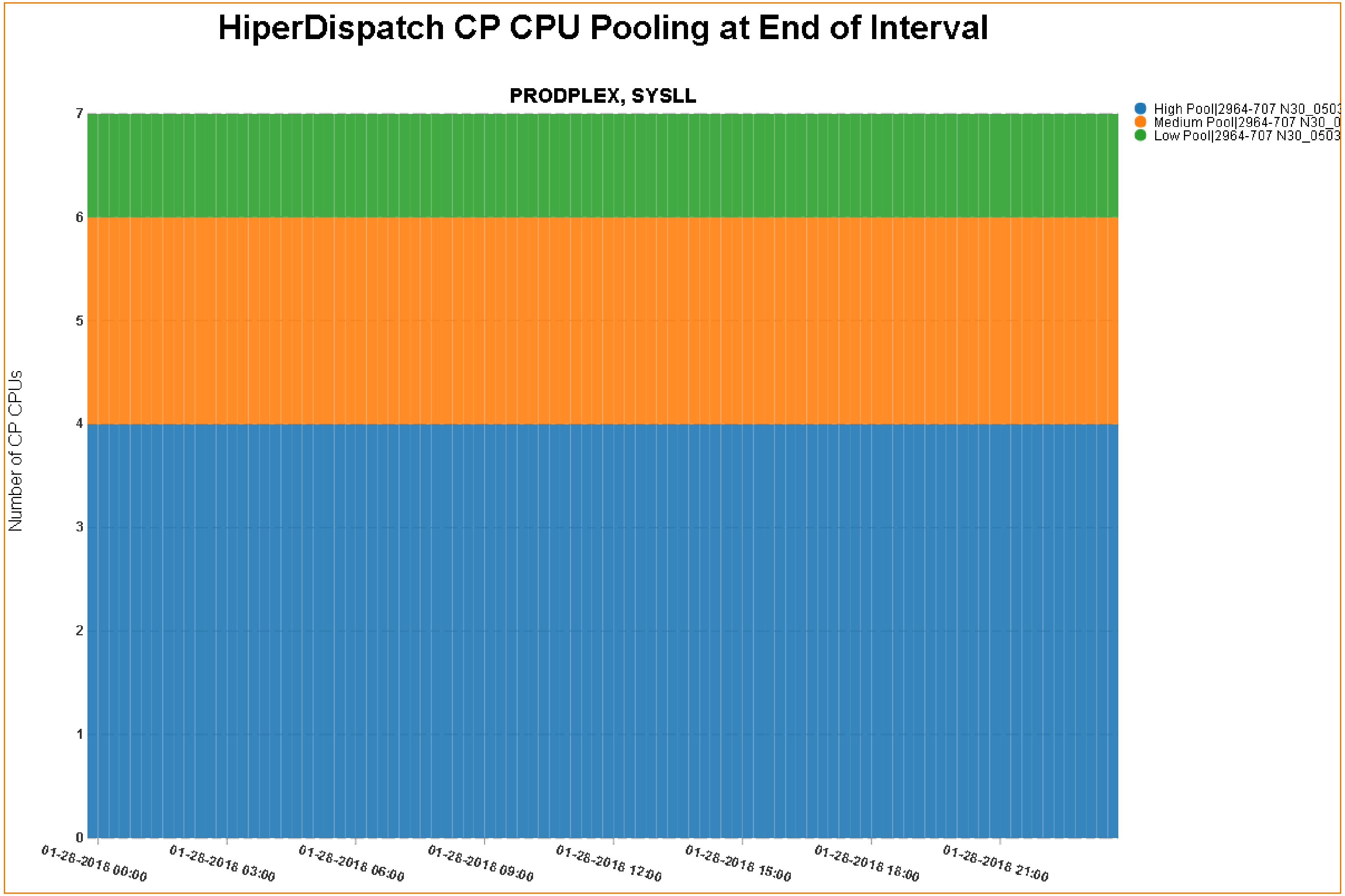
PR/SM Association



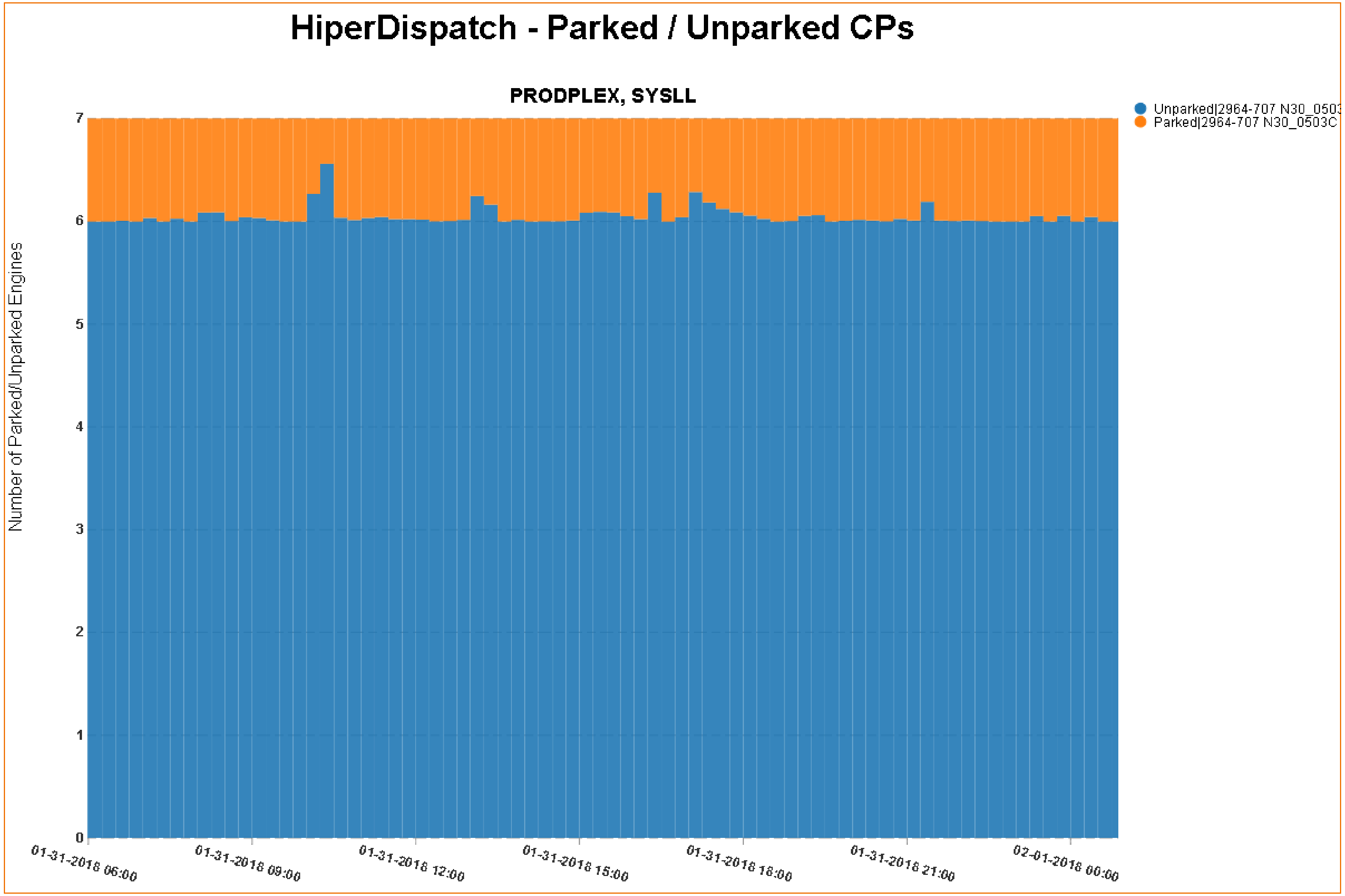
- LPARs and Logical Processors associated with drawers, nodes, and Physical Processors



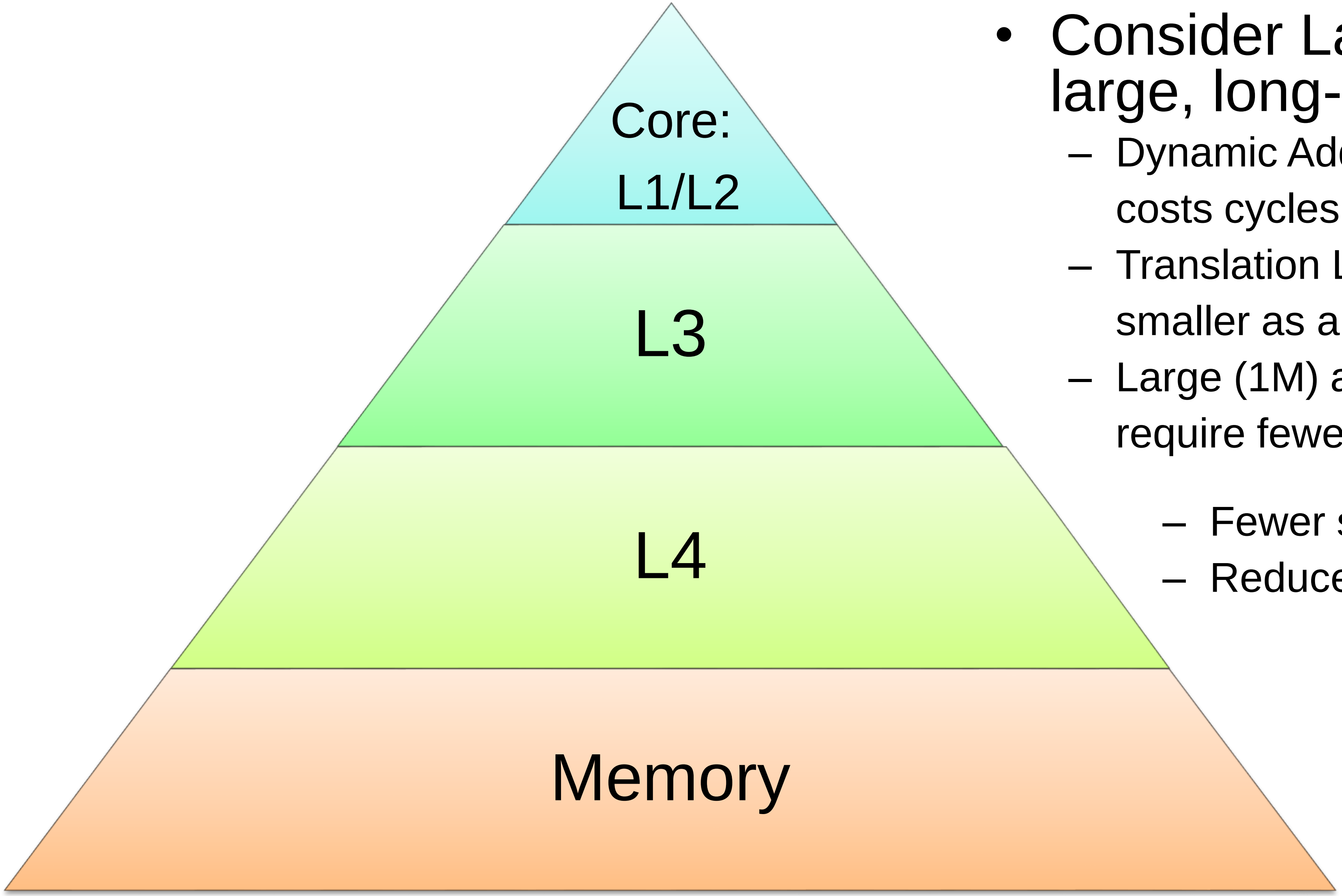
Hiperdispatch Config



Parked Processors

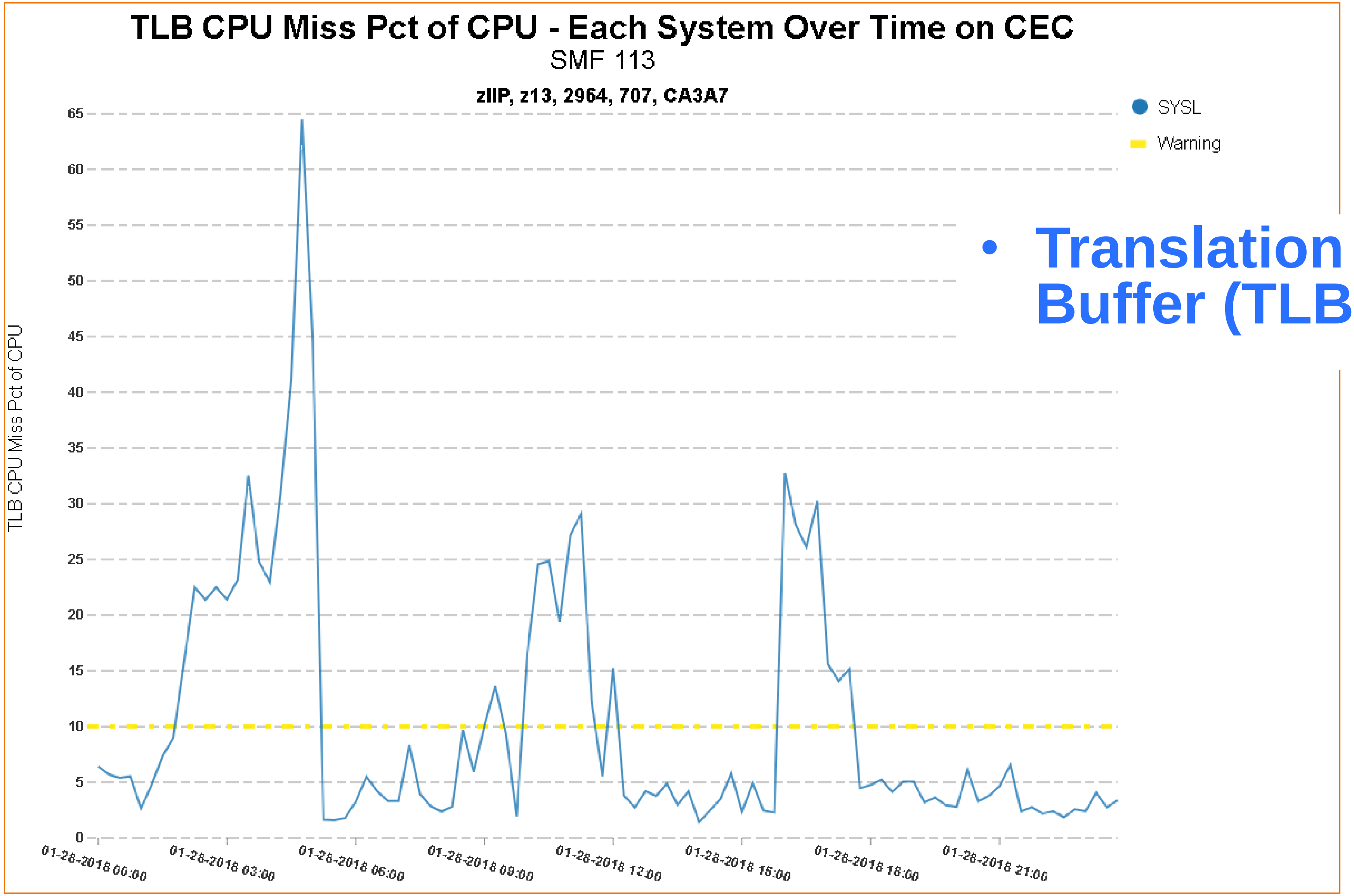


Improving Sync Access



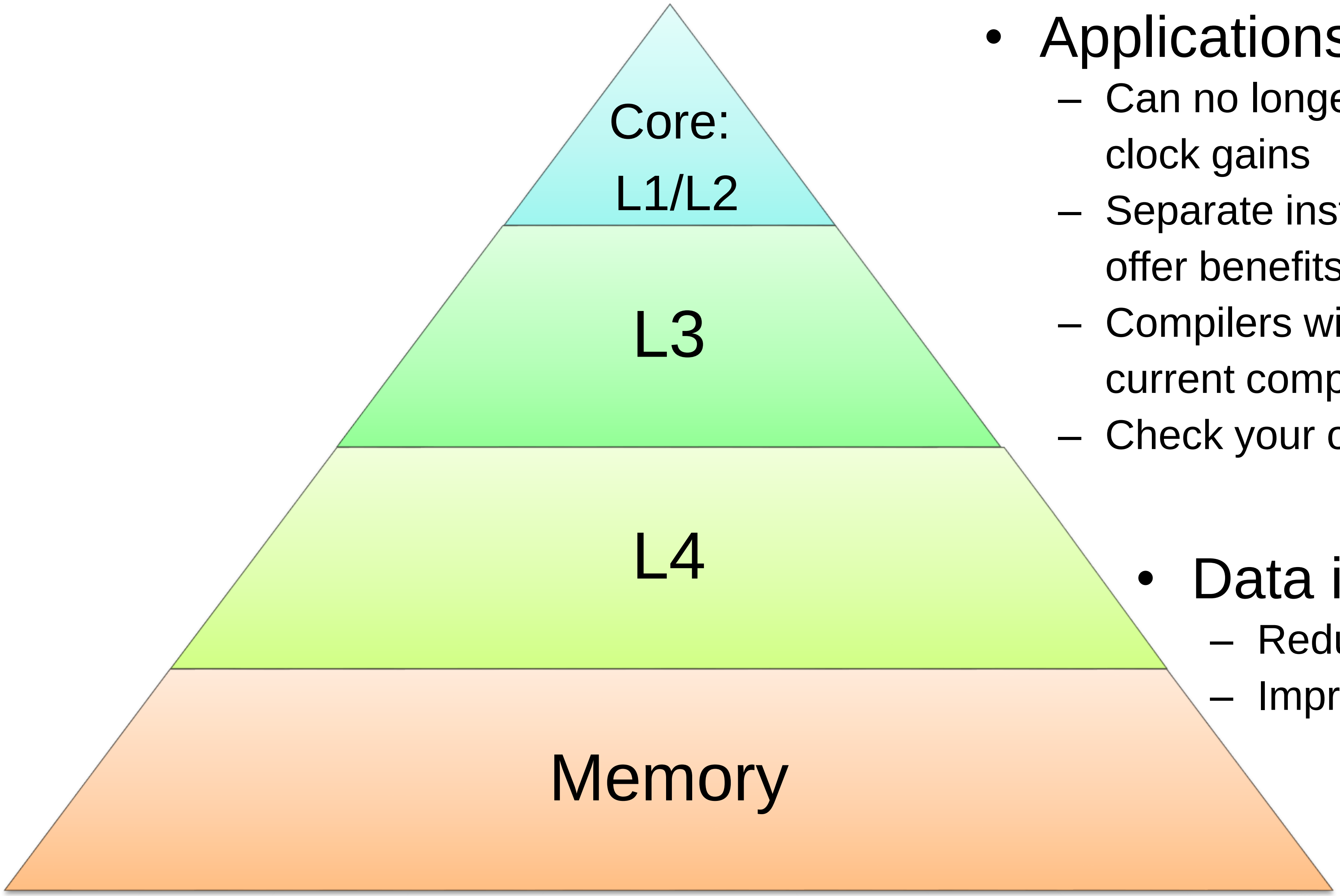
- Consider Large Frames for large, long-running DB2 work
  - Dynamic Address Translation (DAT) costs cycles
  - Translation Lookaside Buffer (TLB) smaller as a % of total memory
  - Large (1M) and giant (2G) frames require fewer TLB entries
    - Fewer segment/page table entries
    - Reduce TLB misses

DAT Cost



- **Translation Lookaside Buffer (TLB) miss percent**

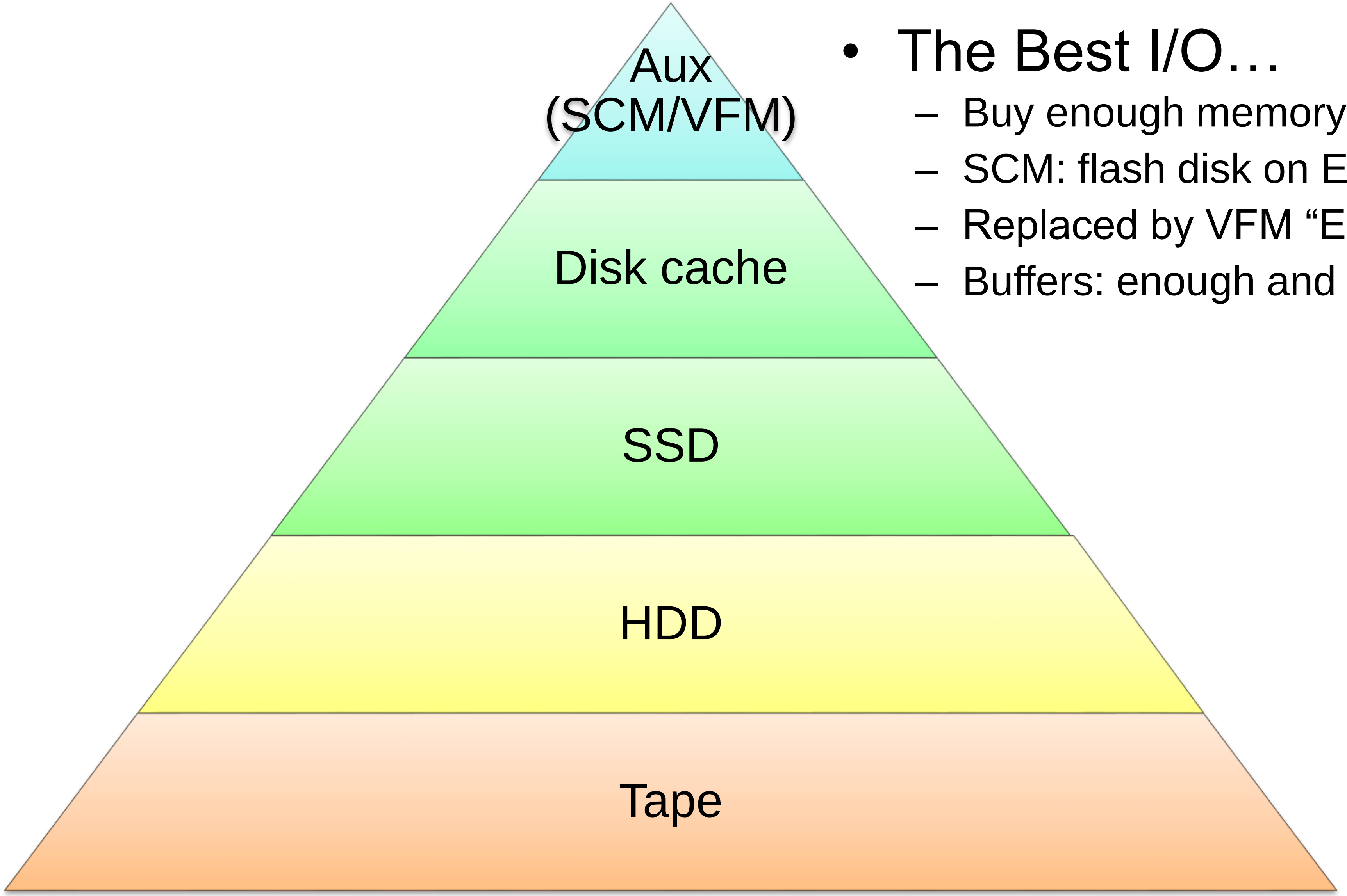
Improving Sync Access



- Applications
  - Can no longer count on large CPU clock gains
  - Separate instruction/data caches offer benefits and drawbacks
  - Compilers will take care of it – use current compilers!
  - Check your old assembler

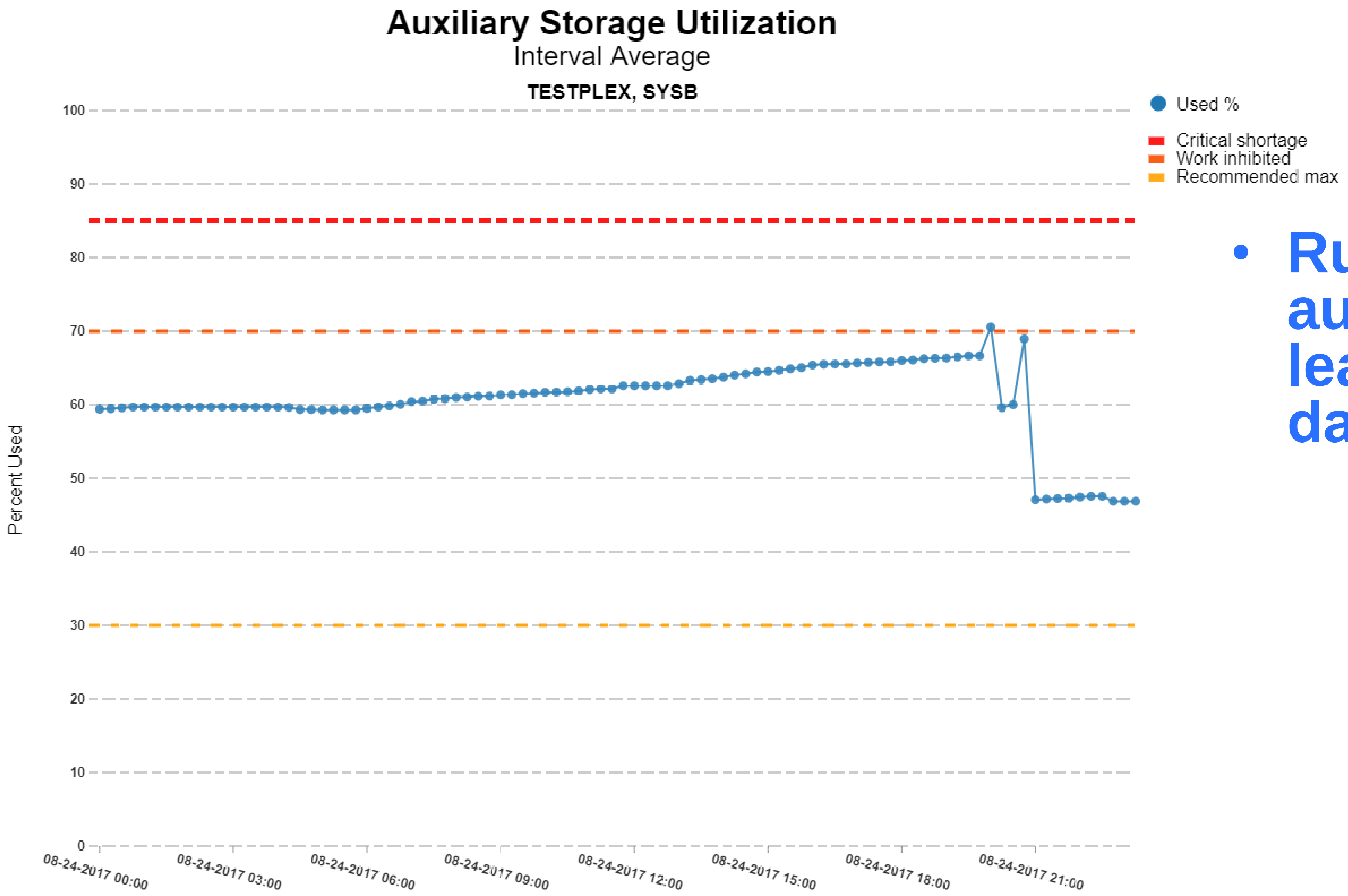
- Data in memory tech
  - Reduce cycles to fetch busy tables
  - Improve response at costs

Improving Async Access



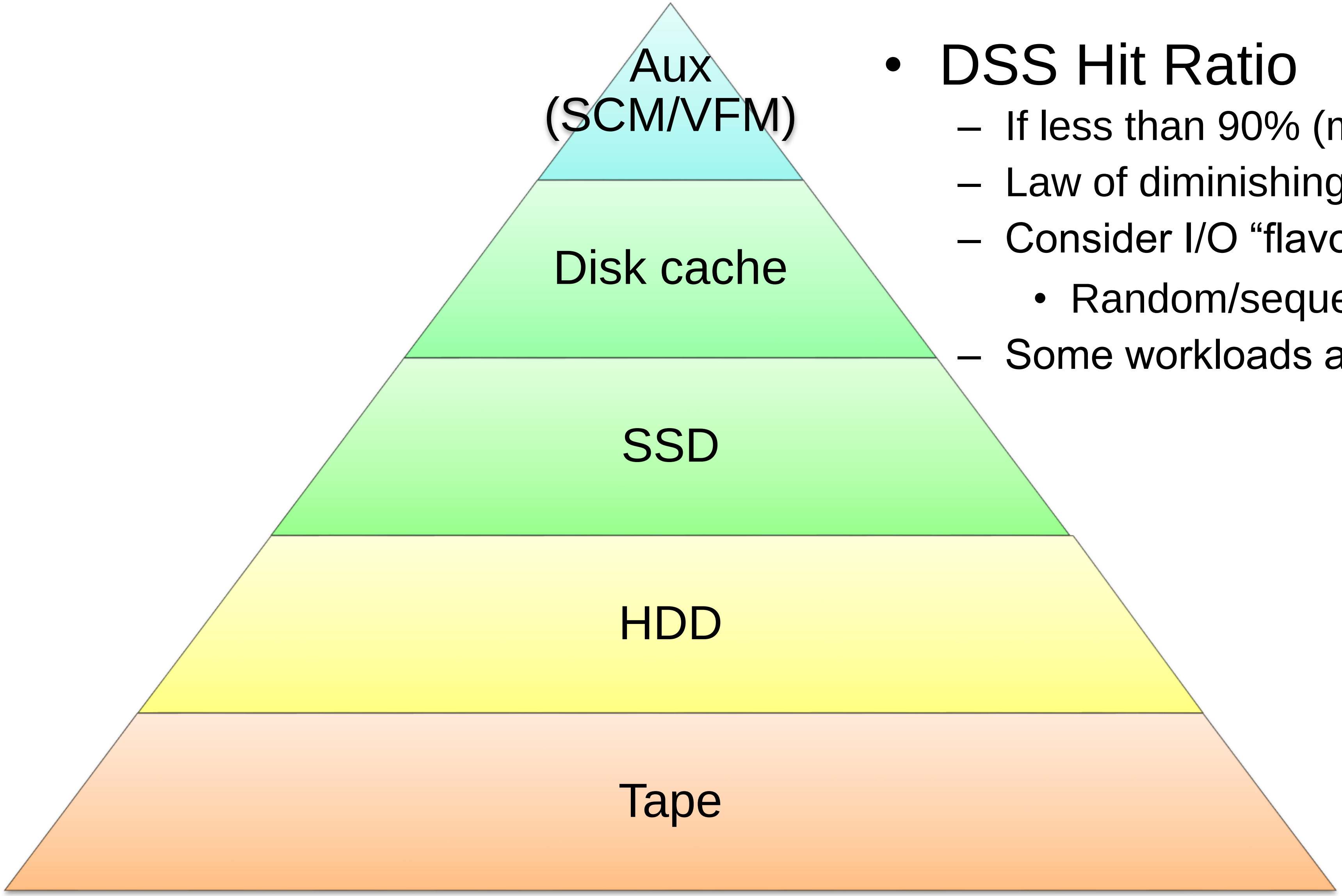
- The Best I/O…
  - Buy enough memory to avoid paging
  - SCM: flash disk on EC12/z13
  - Replaced by VFM “Expanded memory” on z14
  - Buffers: enough and the right type (random VSAM)

Aux Storage Use



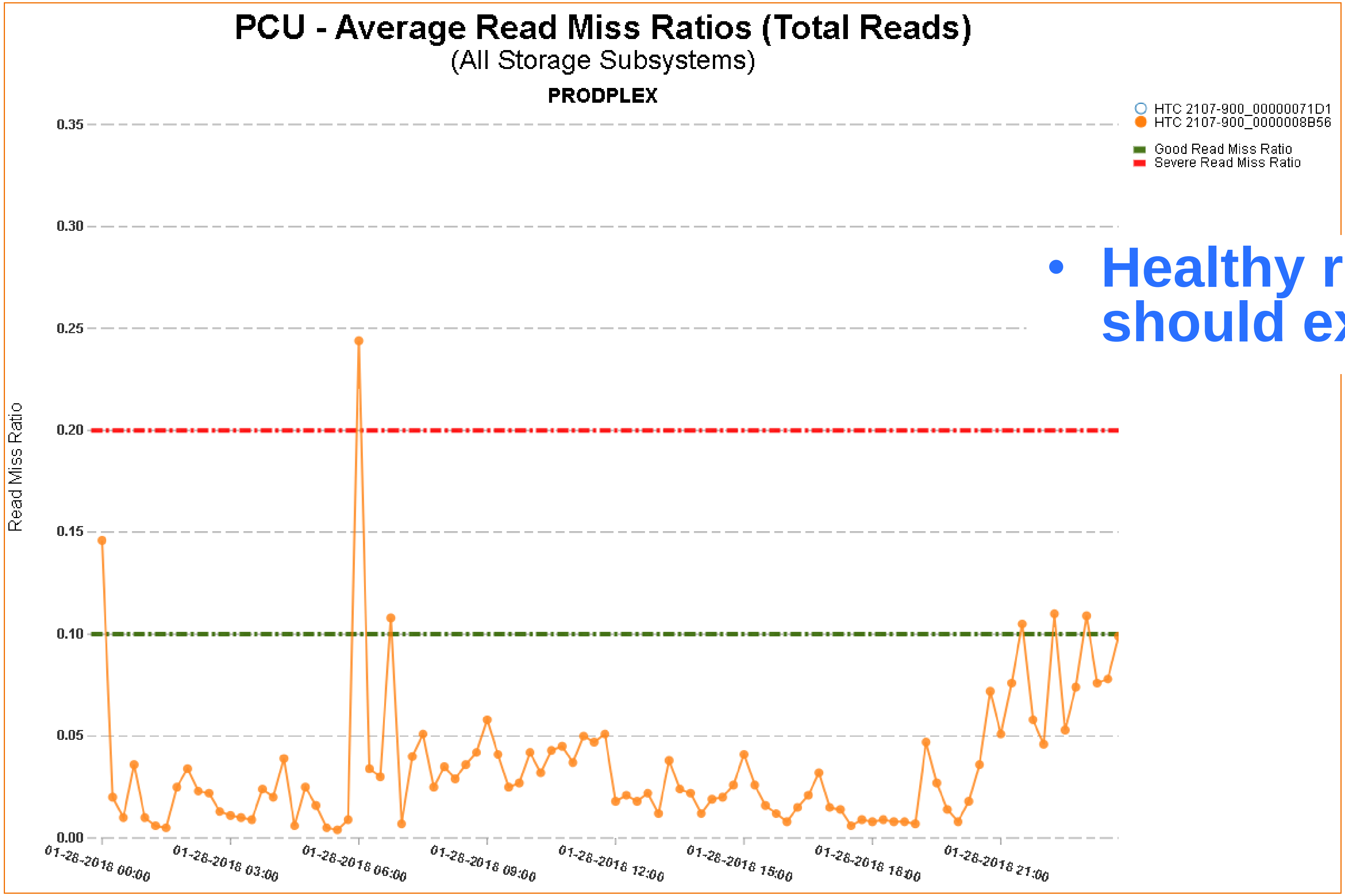
- **Running out of auxiliary storage can lead to a very bad day…**

Improving Async Access



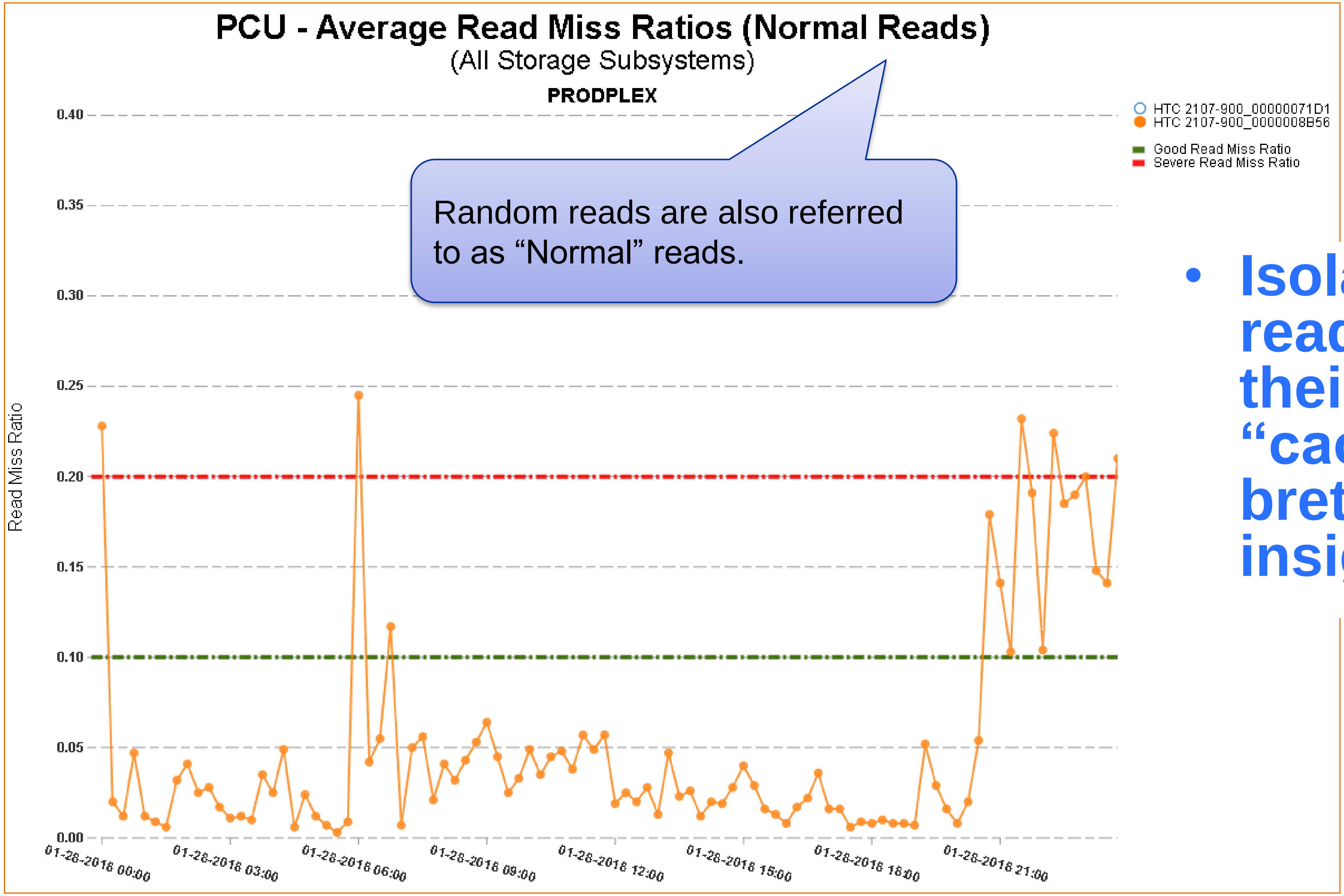
- DSS Hit Ratio
  - If less than 90% (most workloads) > more cache
  - Law of diminishing returns applies
  - Consider I/O “flavor”
    - Random/sequential, read/write
  - Some workloads are inherently “cache unfriendly”

DSS Cache Read Miss %



- **Healthy read hit ratios should exceed 90%**

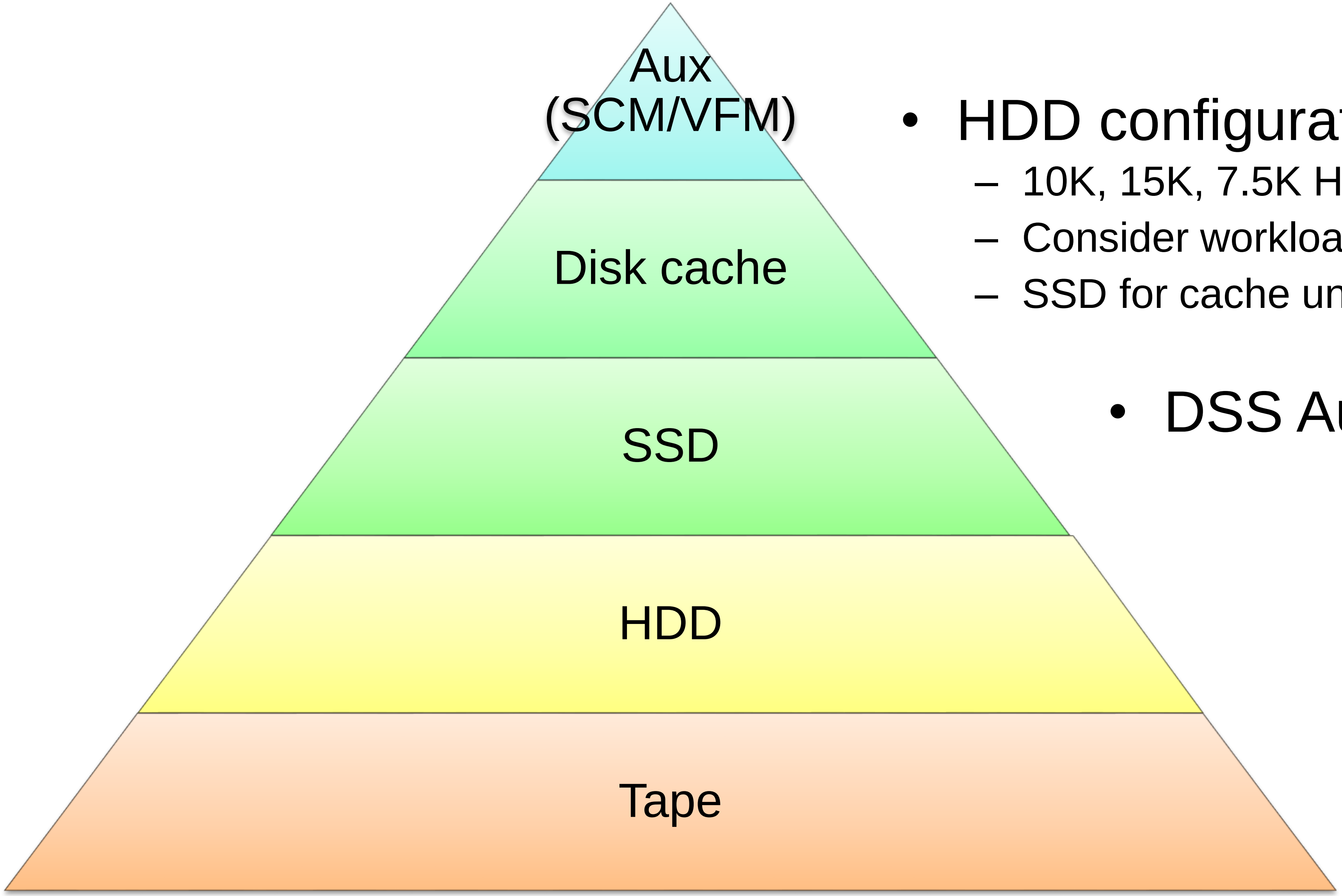
Random Reads only



Random reads are also referred to as “Normal” reads.

- **Isolating random reads, by removing their sequential “cache friendly” brethren, gives more insight**

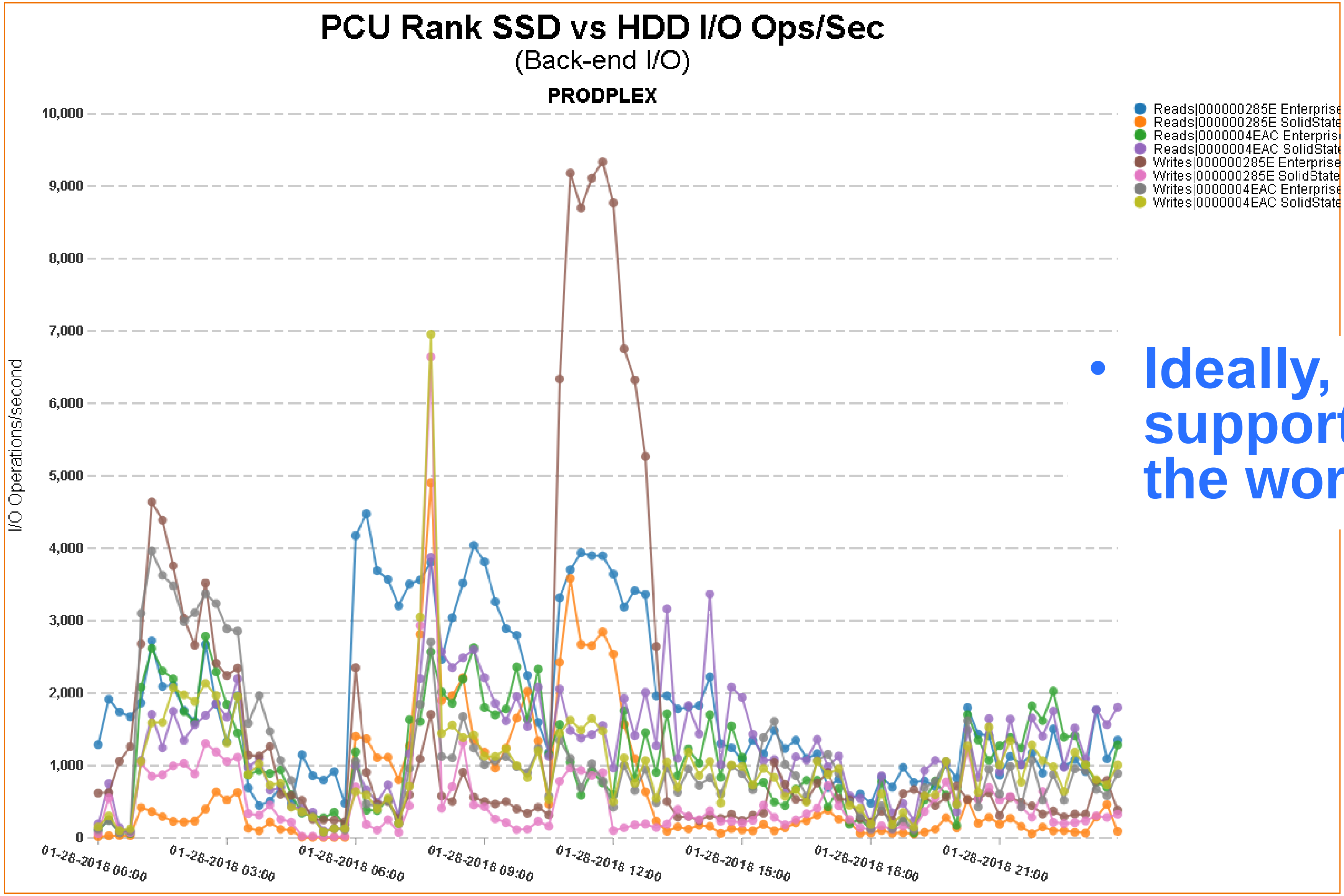
Improving Async Access



- HDD configuration
  - 10K, 15K, 7.5K HDD
  - Consider workload access density
  - SSD for cache unfriendly workloads

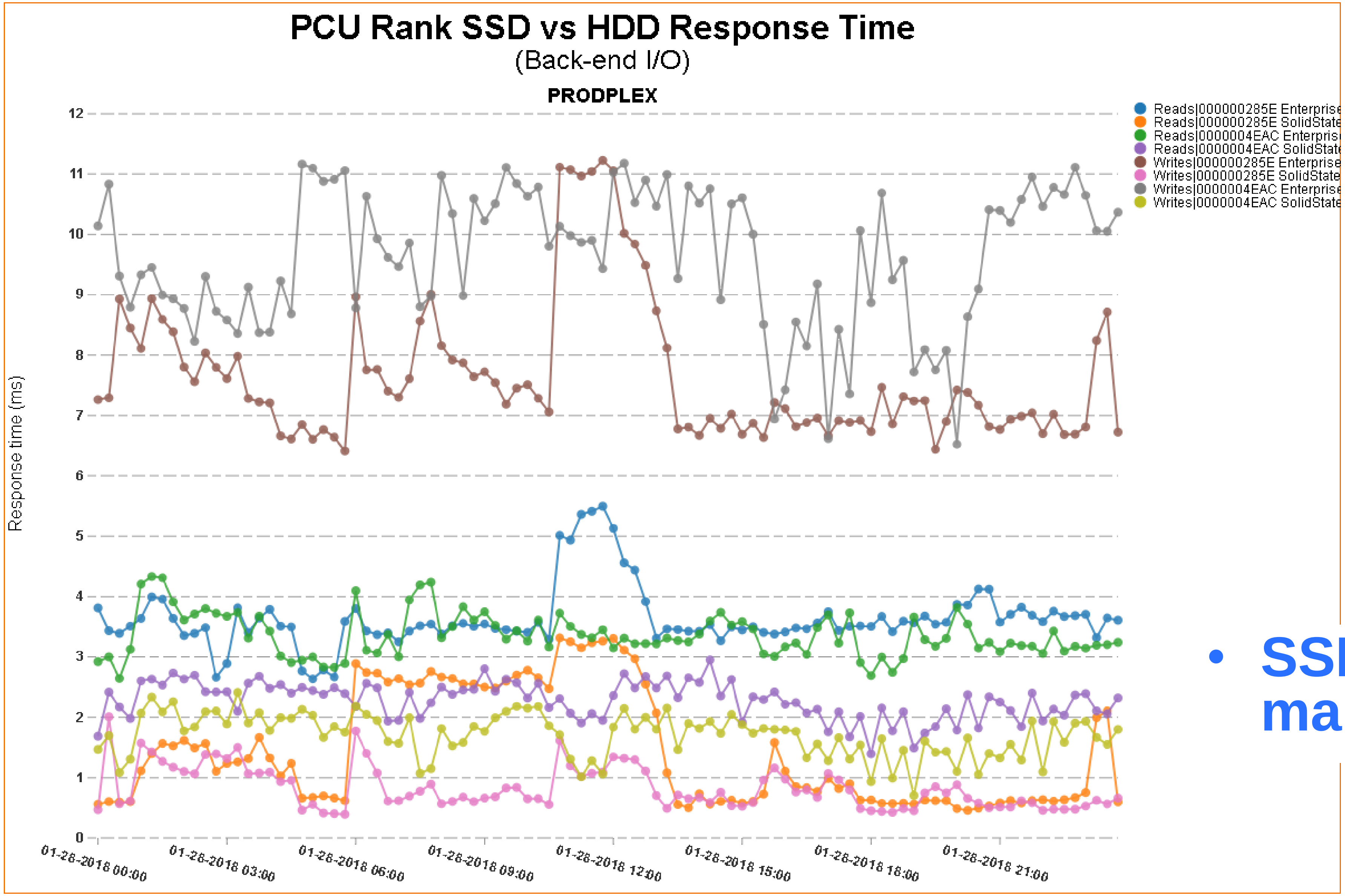
- DSS Auto Tiering or all flash

Disk Activity by Type



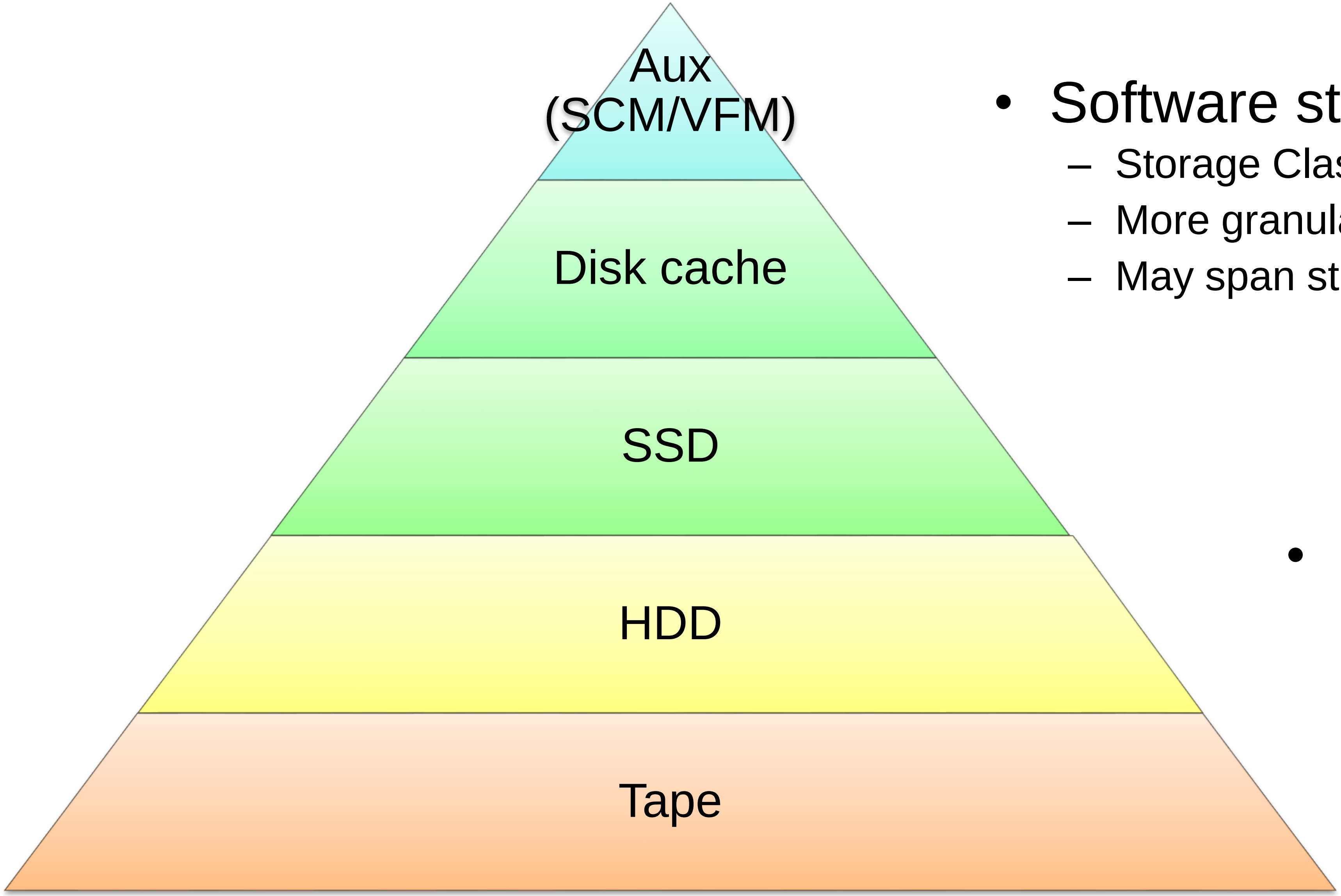
- **Ideally, the SSD’s are supporting the bulk of the workload**

Response by Disk Type



- **SSD benefit is marginal here**

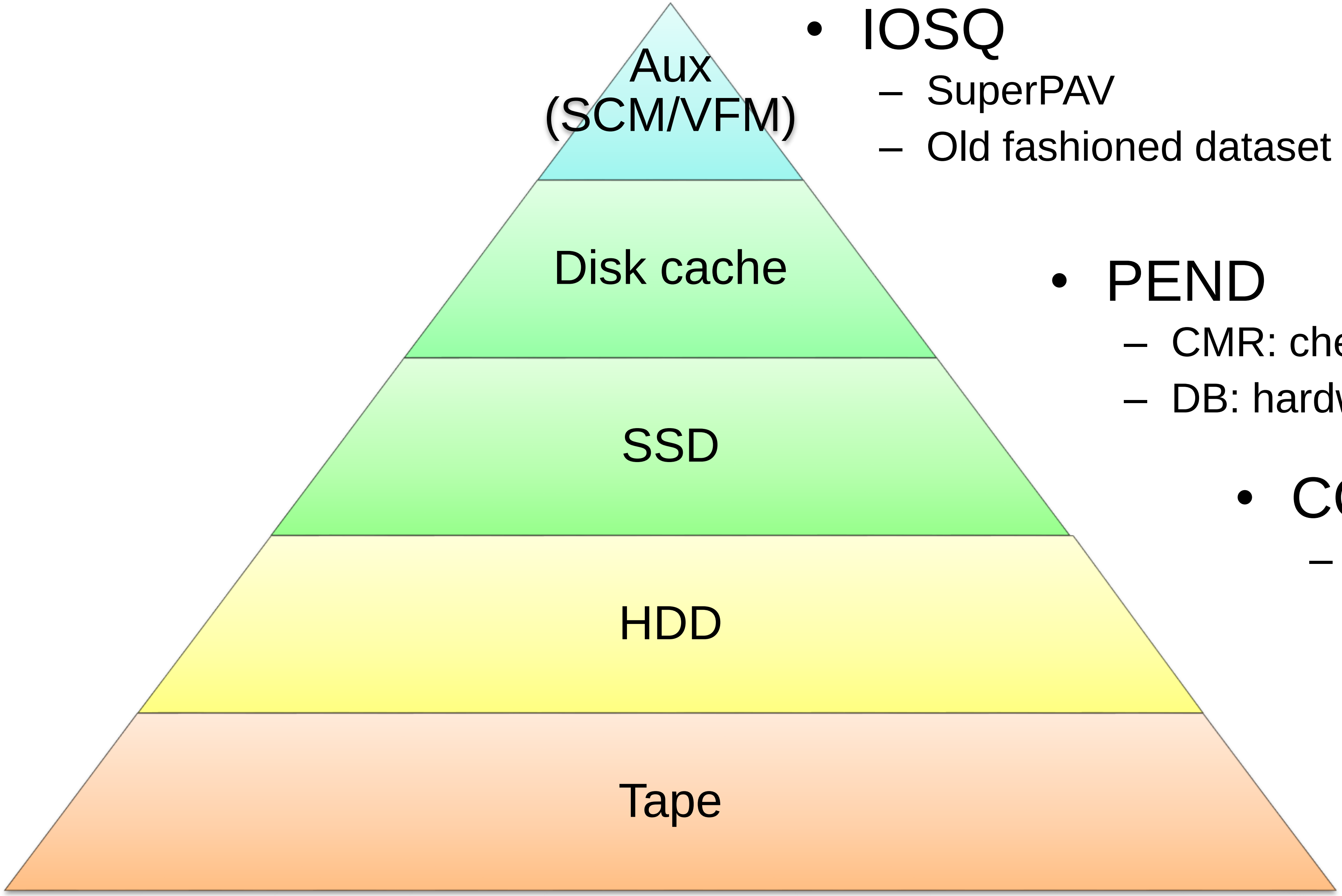
Improving Async Access



- Software striping
  - Storage Class, sustained data rate
  - More granular
  - May span storage subsystems

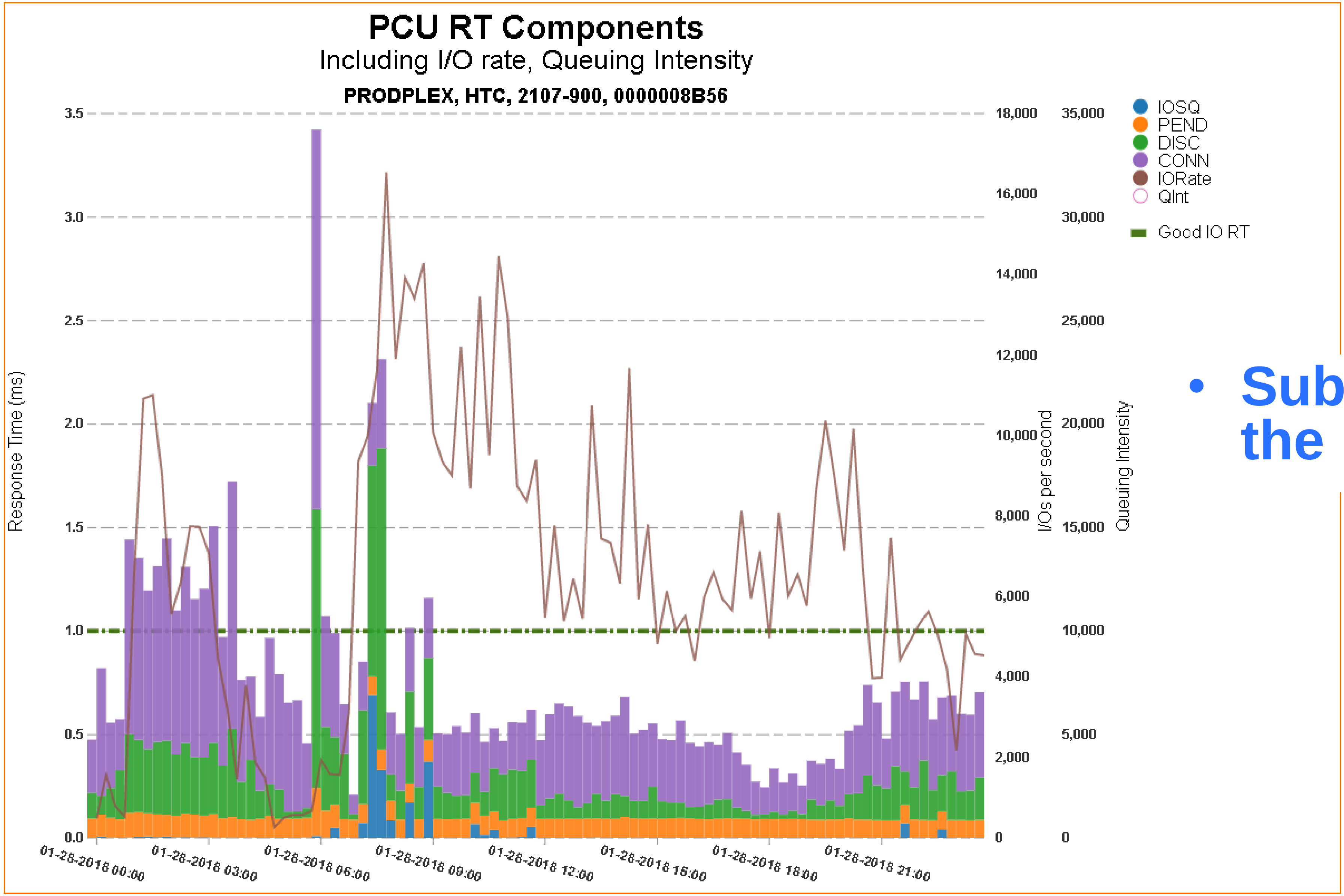
- Compression
  - zEDC ends the debate

Improving Async Access (2)



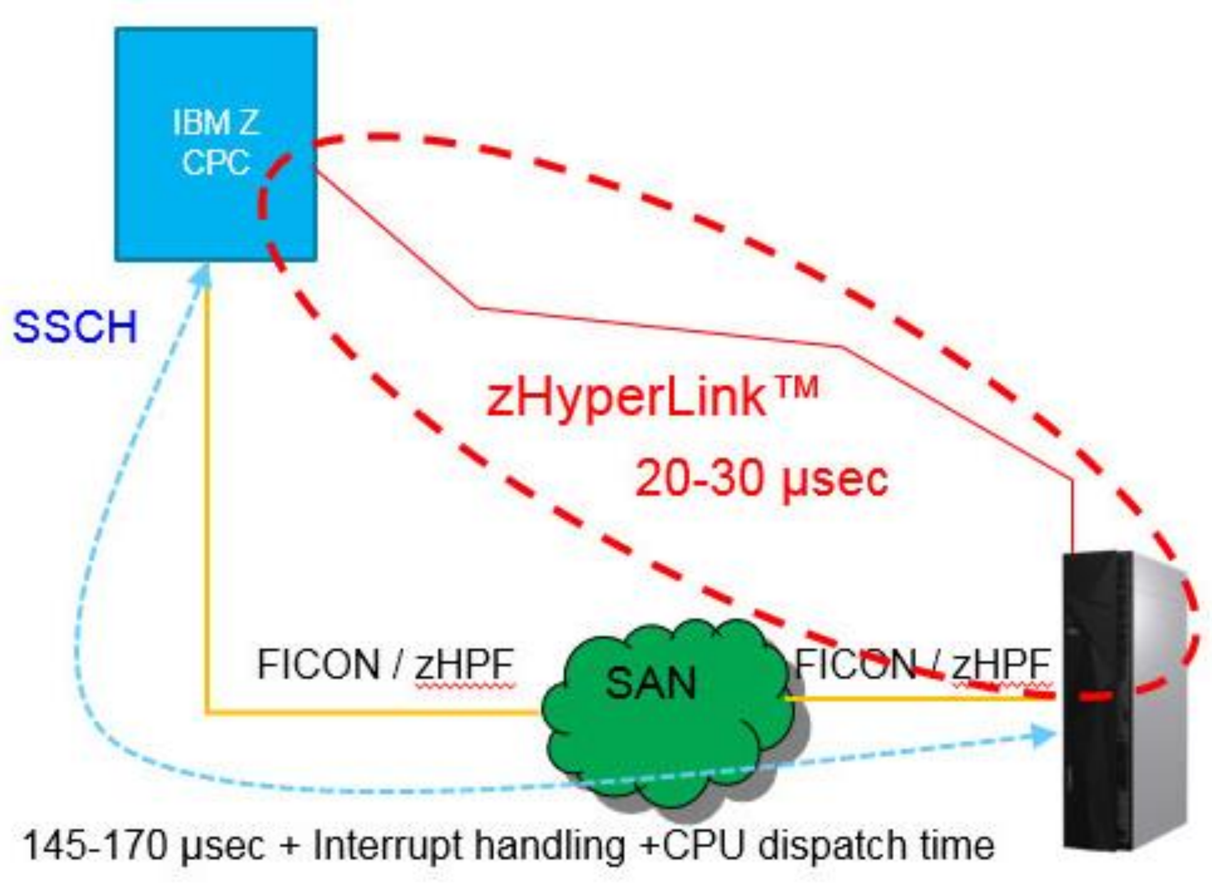
- IOSQ
  - SuperPAV
  - Old fashioned dataset placement. SMF 42
- PEND
  - CMR: check DSS front-end
  - DB: hardware reserves
- CONNECT
  - zHPF, channels, HA’s
- DISCONNECT
  - DSS hit ratio
  - SSD

Response Time Components



- **Sub-millisecond is the new standard**

Something new: zHyperLink



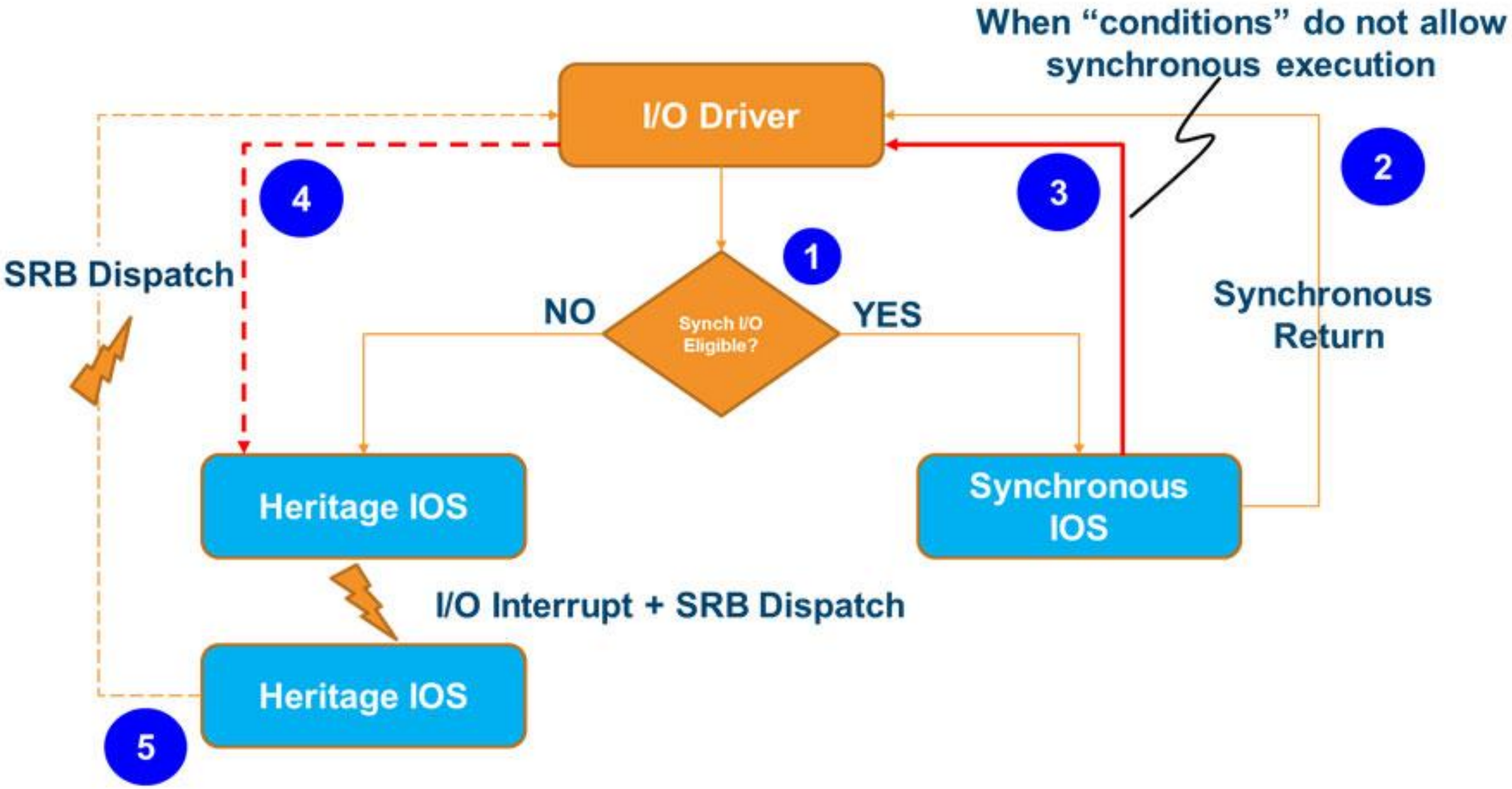
http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/redp5186.html?Open

zHyperlink Requirements

- Z14
- z/OS 2.1
- DS888x with zHyperlink feature (FC #0431)
- DB2 v11+ only (for now)

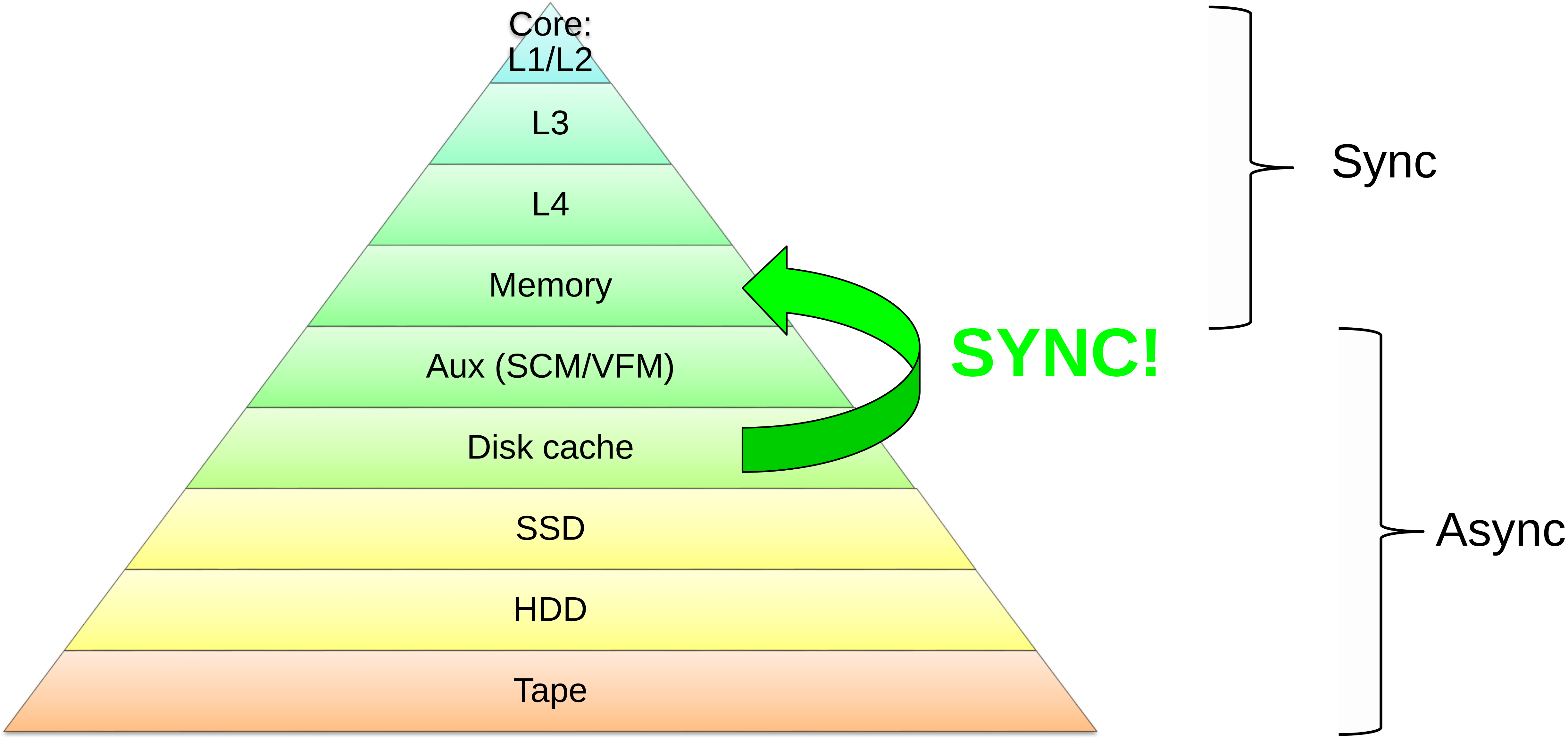
zHyperlink Flow

- Conditions
- Why not make everything sync?



http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/redp5186.html?Open

“External memory”



Thank you!

www.pivotor.com

John.Baker@EPStrategies.com