Evolution of z/OS Memory Management:
Large memory, large pages, and how to use them

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

Do you remember when “large” memory was measured in single-digit GBs, or maybe even MBs? Yeah, I feel old too! It’s a different world today! With z/OS 2.5 now supporting up to 16 TB of real storage z/OS memory management has had to evolve to keep pace. This support for large memory sizes can transform application performance but too few sites are taking advantage of it. Come to this session to learn about how memory management has changed in z/OS, why large pages are important, and get some ideas for how you can leverage large memory to improve the performance of your systems.

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

z/OS Performance workshops available

During these workshops you will be analyzing your own data!

- Essential z/OS Performance Tuning
  - March 20-24, 2023
- WLM Performance and Re-evaluating Goals
  - October 2-6, 2023
- Parallel Sysplex and z/OS Performance Tuning
  - May 2-3, 2023
- 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
Evolution of z/OS Memory Management: Large Memory, Large Pages, and How to Use Them	Scott Chapman	Mon 1:15	
Back to Basics – Introduction to Parallel Sysplex and Data Sharing	Peter Enrico	Wed 9:15	
z/OS WLM - Revisiting Goals Over Time	Peter Enrico	Wed 10:30	
PSP: z/OS Performance Tuning – Some Top Things You May Not Know	Peter Enrico Scott Chapman	Wed 1:15	

Agenda

- A bit of history
- A bit of ranting
- A bit of advice

History and Background

A line walks into a bar.

The bartender throws him out saying “We don’t serve ropes here!”

The dejected line stands outside for a few minutes, then unraveles his ends a bit, tangles himself up, and walks back into the bar.

The bartender looks up and says:

“Hey you’re not one of those ropes are you?”

The line replies: “Nope, I’m a frayed knot”

The Line and the Bar

- In the earliest days of MVS memory addressability was limited to 24 bits
  - Use your 16MB wisely!
- MVS/XA arrived in 1983 with support for 31 bit addressability
  - Yay! 2GB of addressability!
  - Bimodal addressability: below and above the 16 MB “line”
- z/OS 1.1 arrived in 2001 with support for 64 bit addressability
  - Trimodal addressability: 24-bit, 31-bit, 64-bit
  - Initially only data (not instructions) above the 2 GB “bar”
  - Theoretically 16 Exabytes of addressability, but limited to 4TB
  - Maximum physical LPAR size also 4 TB

Comparisons

	16 MiB (24 bits)	2 GiB (31 bits)	16 EiB (64 bits)
Introduced	1974 MVS	1983 MVS XA	2001 z/OS 1.1
Compared to previous		128x	8,589,934,592x 2,048x (4 TiB “practical”)
Hardware limit at launch	8 MiB System/370 Model 168	64 MiB IBM 3084	64 GiB z900
PCs	N/A	640 KiB (but likely <=256 KiB) IBM PC	4 GiB (but likely <= 256 MiB) Windows XP

Today

- Up to 40 TiB of memory available on z16
- z/OS 2.5 now support 16 TiB per LPAR on z14 and above
  - 8 TiB on z14-ZR1
  - 10 TiB on z13 (still 4 TiB on z13s)
- 8 GiB to IPL z/OS (without warnings)
- LPARs with 100s of GiB of memory quite common
  - And somebody was probably happy with the new 16 TiB LPAR limit!
  - Recent review of our customers' data:
    - Smallest LPAR was 3GB
    - Largest LPAR was about 2.2TB

Storage Class Memory

- zEC12 introduced Flash Express
  - Basically: use internal SSDs for paging, dump processing, and certain CF structures
  - More generically called “Storage Class Memory” since it was using SSDs
- Replaced with Virtual Flash Memory on z14
  - Allocates some memory on the machine for the SCM use cases
  - Up to 6 TiB (in 512 GiB increments) on z16
- Provides easier paging configuration and faster dumps
  - Little value in over-configuring it though: hopefully you’re not paging and dumping much!
  - Paging to Virtual Flash Memory very fast but still takes some CPU

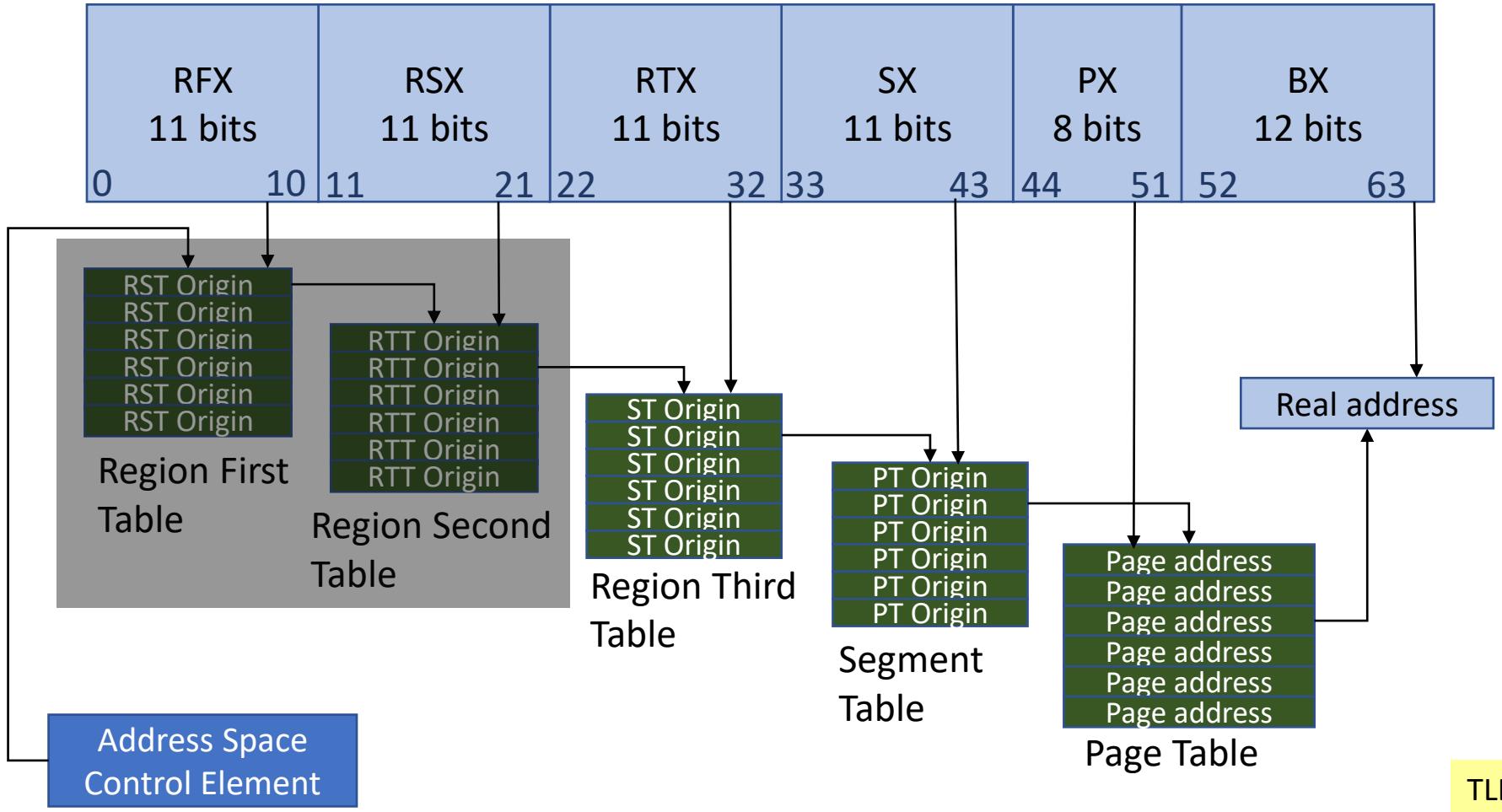
Speaking of paging...

- Three page sizes available to z/OS
  - 4 KiB historical default
  - 1 MiB “Large” pages
  - 2 GiB “Large” pages
    - Should we call these “Giant” pages? Or “Extra-Large”?
- Only 4K and 1M pages can actually be paged out: 2G pages always fixed
- Large pages are more efficient due to reduced DAT overhead
  - Although DAT did get much more efficient on z14 and later
- Every (modern) z/OS system has some large page usage today
- 2G page use cases still somewhat limited but seeing more usage of them
  - DB2 Buffer pools – most common use case
  - Java heaps
  - ??

Dynamic Address Translation

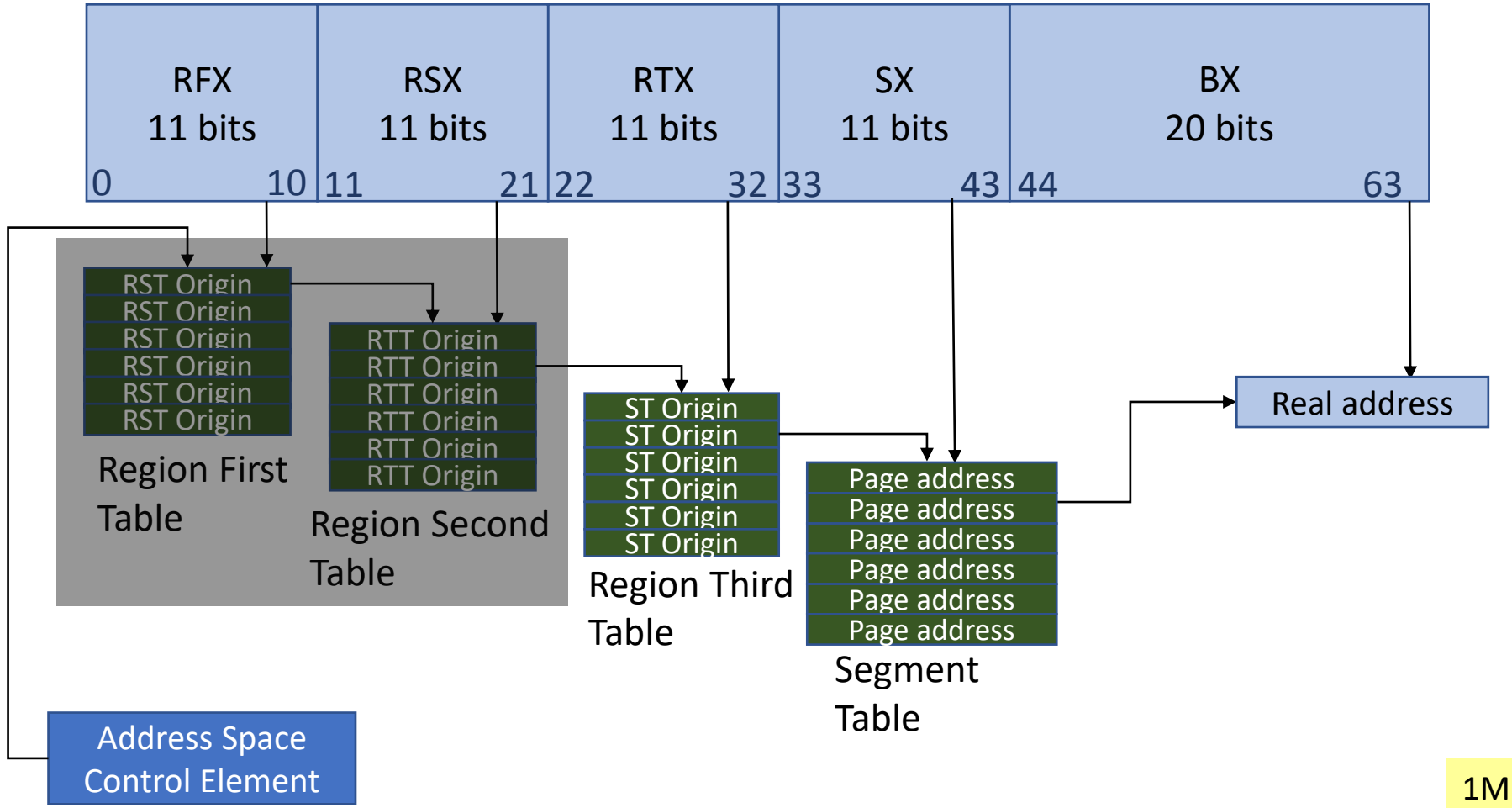
- DAT performed using multiple tables that point to different ranges of storage
- DAT is not free!
  - z14 hardware changes greatly reduced the cost of DAT, but still is typically 1-2% of all CPU consumption
- Result of DAT cached in Translation Look-aside Buffers (TLB)
- TLBs are in L1 cache and managed by the hardware
- Relatively small
- Flushed when DAT table changes
- 1MB & 2GB pages make DAT and TLB more efficient

z/OS 64-bit Address Translation (4K)



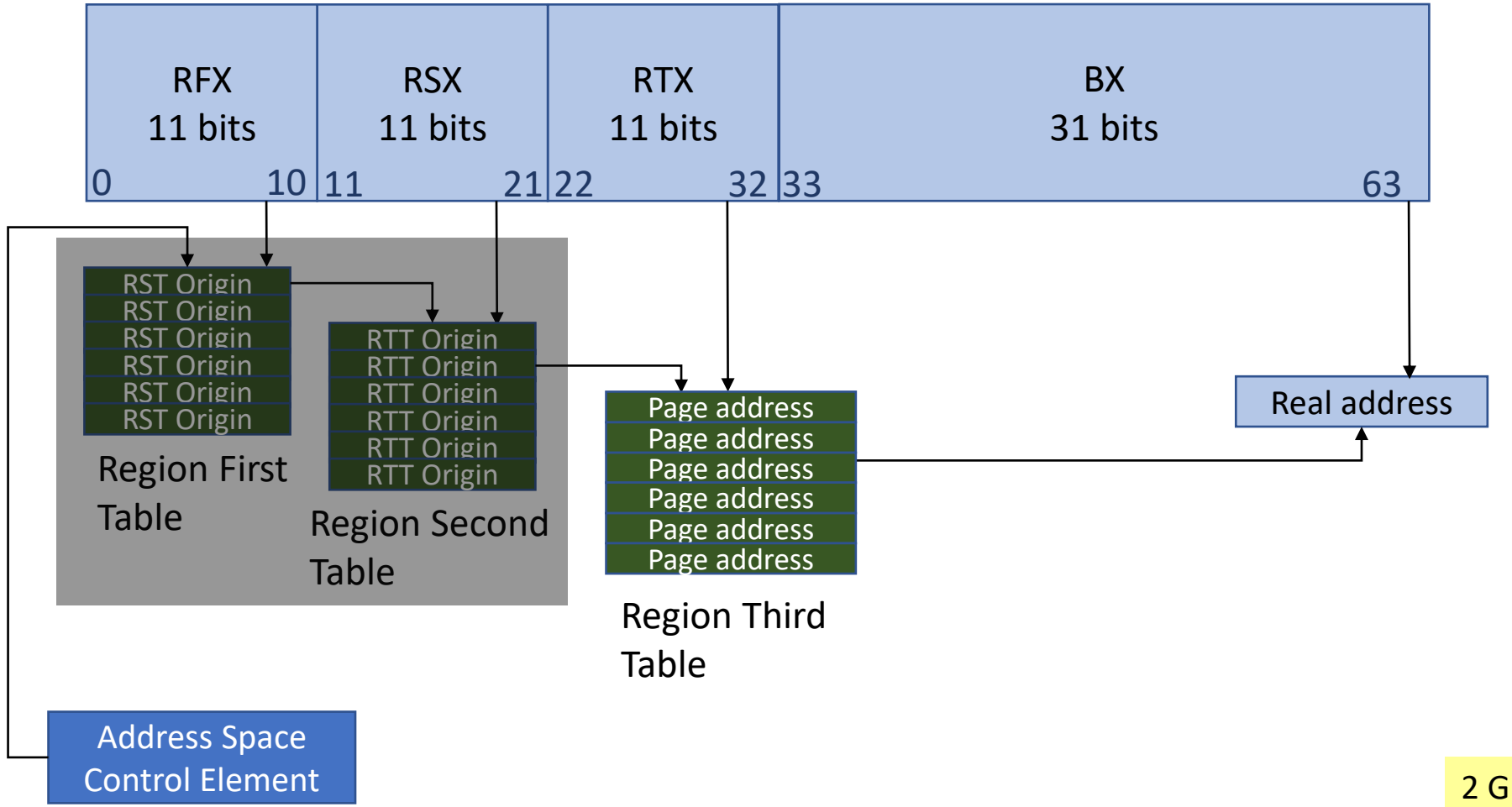
TLB miss means evaluating these tables

Large Page Address Translation



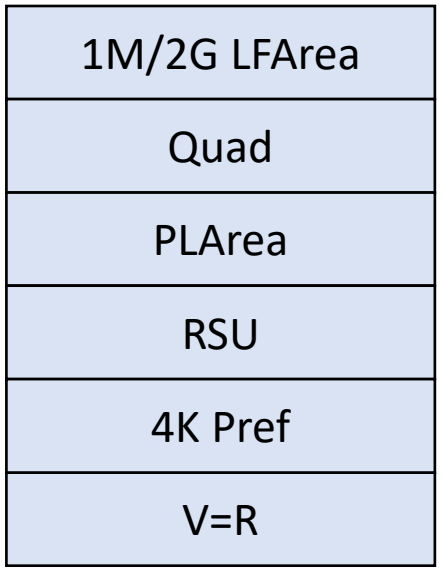
1MB pages means 1 less table

Giant Page Address Translation



2 GB pages again removes a table

Storage Areas (pre-z/OS 2.3)

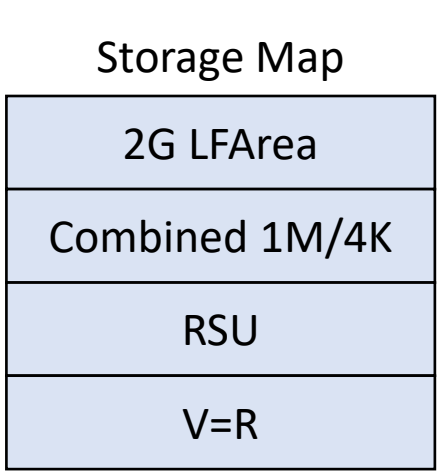


- LFArea – Fixed 1M/2G pages
- Quad frames – 1/8th of online storage
  - 4 contiguous (in real) 4K pages on particular boundary
- PLArea – Pageable 1M (1/8th of online – Quad)
  - 10.9375% of memory (sometimes erroneously said to be 1/8th)
  - Created if running on a zEC12 or later machine
- RSU – Reconfigurable Storage
  - Probably want this 0 or effectively 0 with “offline”
- 4K Pref –Preferred area for 4K frames
- V=R reserved from IEASYSxx REAL parm
  - Recommended to be 0

Storage availability (pre-z/OS 2.3)

- Requests for pageable 1M pages taken from PLA
  - If no 1M pages available:
    - Fixed 1M frames converted to pageable
    - If no more 1M frames available, use multiple 4K frames for the request
- If request for a 4K page can't be satisfied, demote pages in order:
  - Pageable 1M frames
  - Quad frames
  - Fixed 1M frames
  - Note: 2G pages will not be broken down for a 4K request!
- In other words:
  - Pageable 1M can "spill" into fixed 1M area
    - But fixed 1M frames are limited to what was specified for them
    - Running out of fixed 1M frames means you probably could use more
  - 4K page requests can cause demotion of 1M pages if 1M pages over-specified

Storage Areas (z/OS 2.3)



- LFArea – Fixed 2G pages only
  - LFAREA 1M parm now just specifies the maximum number of 1M fixed frames that are allowed to be allocated (i.e. no specific area set aside for 1M fixed frames)
- No PLArea
  - No arbitrary limit of 10.9375% for 1M pageable frames
  - Allows potentially many more pageable 1M frames
- 1M & 4K frames allocated from same area
  - Reducing pools avoids overhead of managing pools

Pageable 1M allocations now recorded in SMF71PL* instead of SMF71L6* (* = M, X, or A)

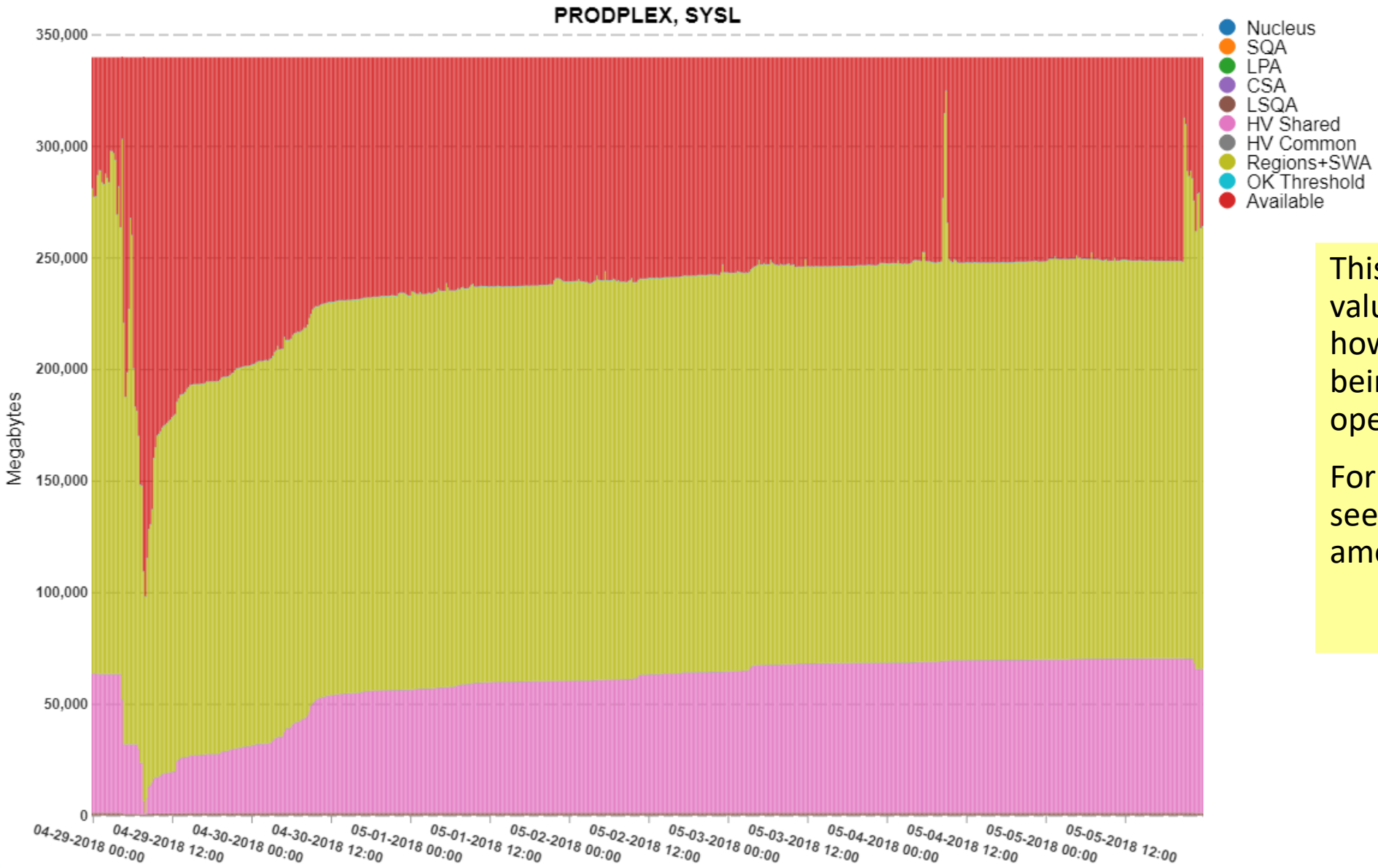
z/OS 2.5 Support for 16 TB LPARs

- With z/OS 2.5 and z14+ can have LPARs up to 16 TB (vs. 4)
  - z14 ZR1 limit 8 GB, z13 limit 10 GB
  - I haven’t seen LPARs pushing the 4 TB limit, but…
- Memory >4 TB is all 2 GB pages
- Consider physical configuration within the drawers
  - Although frankly off-drawer memory access is still way faster than going to disk!

Measuring and Ranting

Central Storage Areas

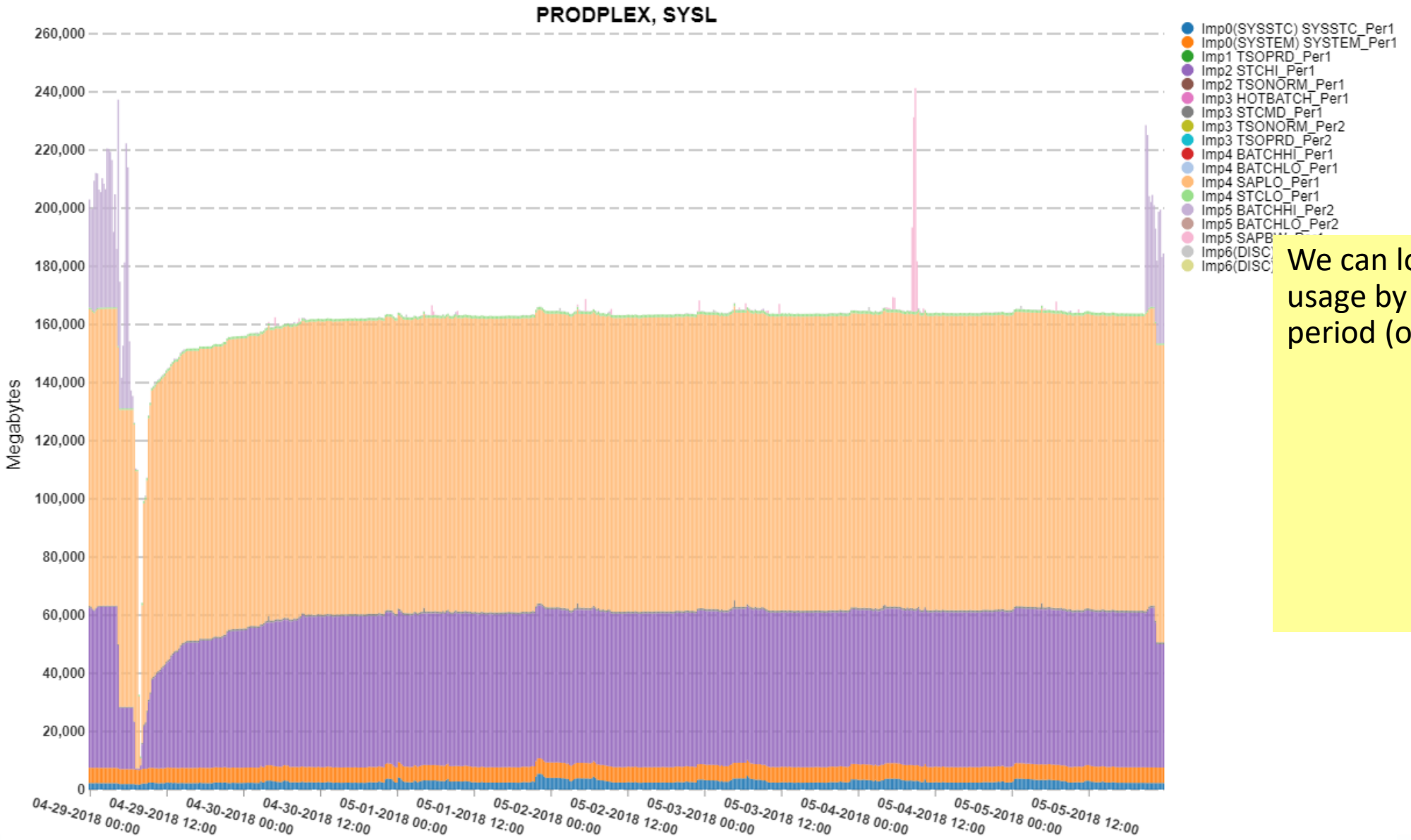
Average Megabytes



This is the average values per interval by how the memory is being used by the operating system.

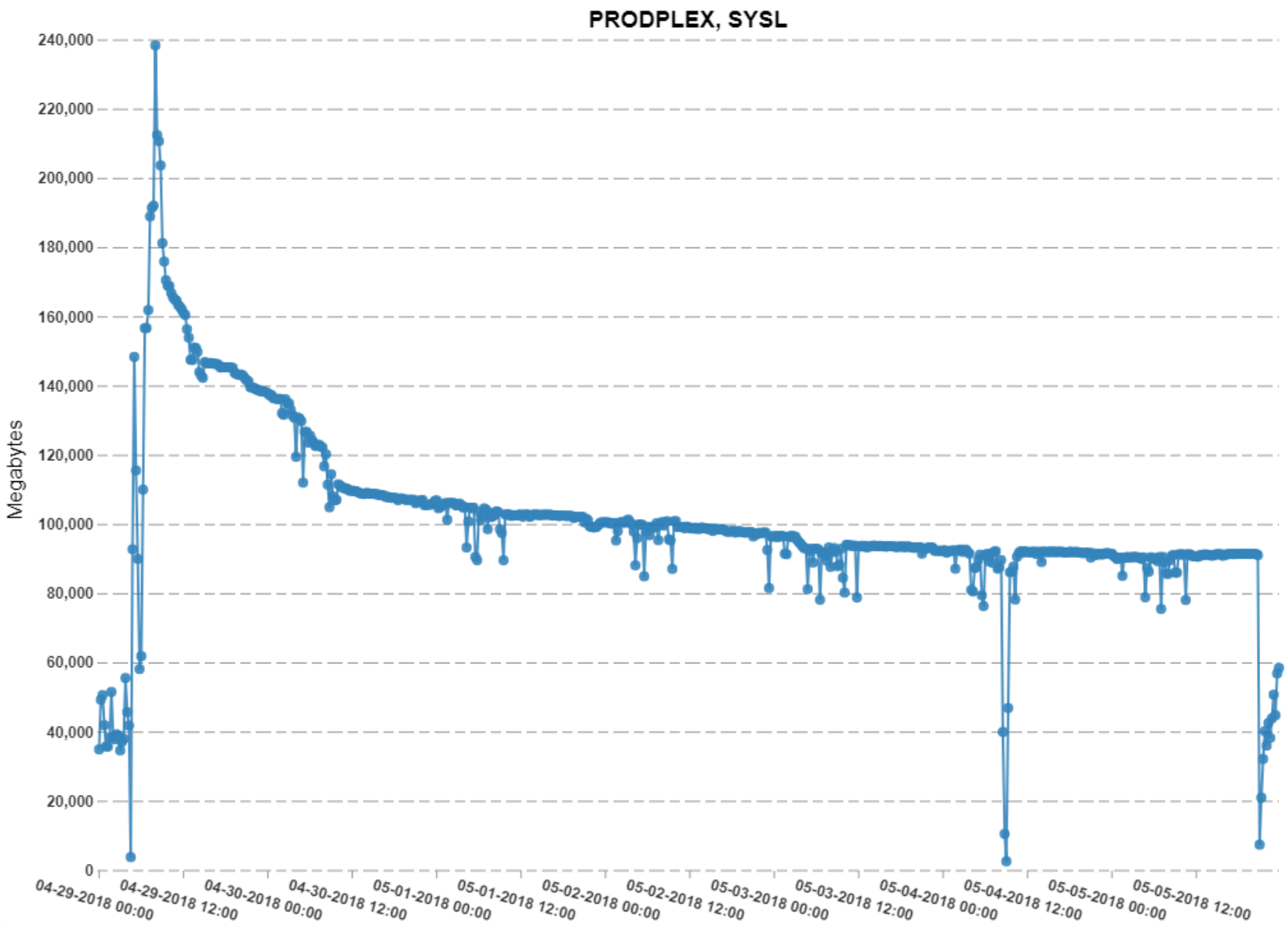
For the most part there seems to be healthy amounts of free storage.

WLM Storage - MB Storage Occupied by Service Class Periods



We can look at memory usage by service class period (or report class).

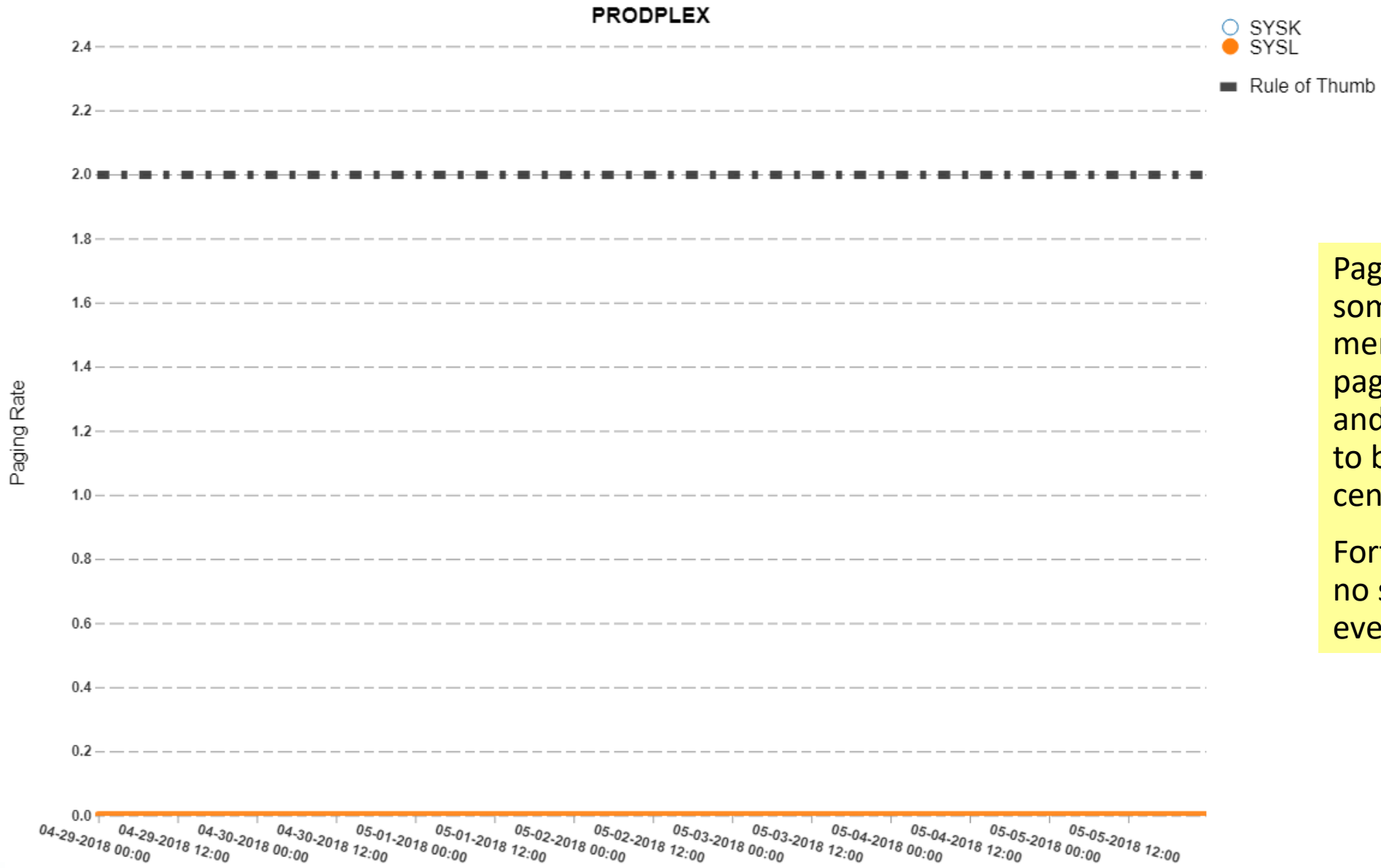
Minimum Available Central Storage



The minimum available value shows the minimum in the interval, which is a better indicator of how low memory is really getting.

The deep dips may not be a problem, they may be effective (brief) use of the available memory, perhaps for sort work. Which is fine, and probably will degrade fine if there's less memory available.

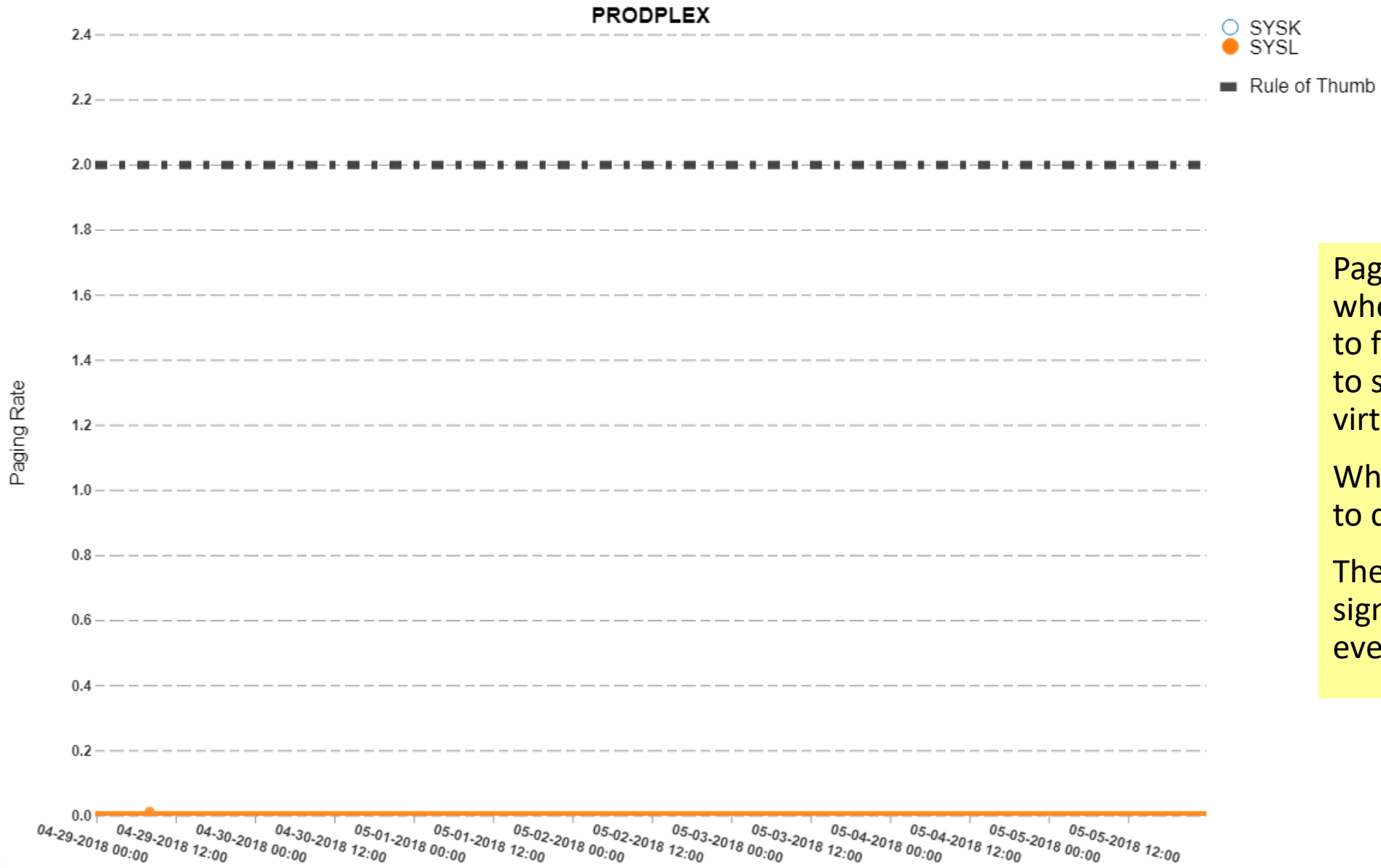
Page-In Rate - All Systems



Page-in means that something referenced memory that had been paged out to aux storage and so had to wait for it to be read back into central storage.

Fortunately, there were no significant page-in events.

Page-Out Rate - All Systems

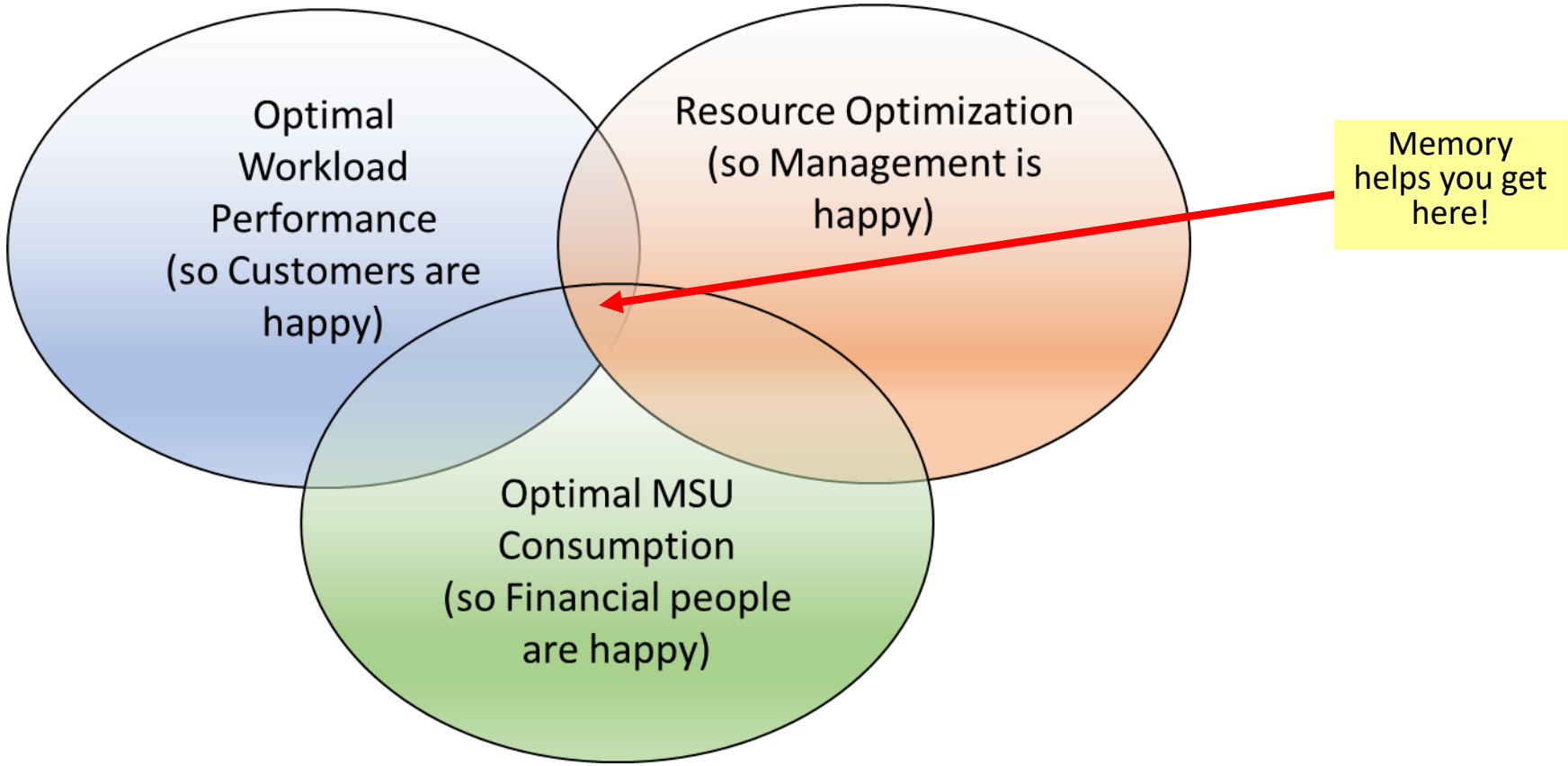


Page-out primarily occur when the system needs to free some real storage to satisfy a demand for virtual storage.

What goes out often has to come back in.

There were no significant page-out events either.

Performance Management Thoughts



Still: The only good I/O is no I/O

- Yes, I/O can be really fast today, but it still takes time
  - But memory then would be really^2 or really^3 fast
- I/O still takes CPU
  - And giving up the CPU to do I/O means that likely when redispatched the work won't have its data and instructions in L1 cache
- Software cost driven by CPU utilization
  - Usually: Software Cost > Hardware Cost
- Performance gated by bottlenecks
  - I/O not always the bottleneck, but is a common one

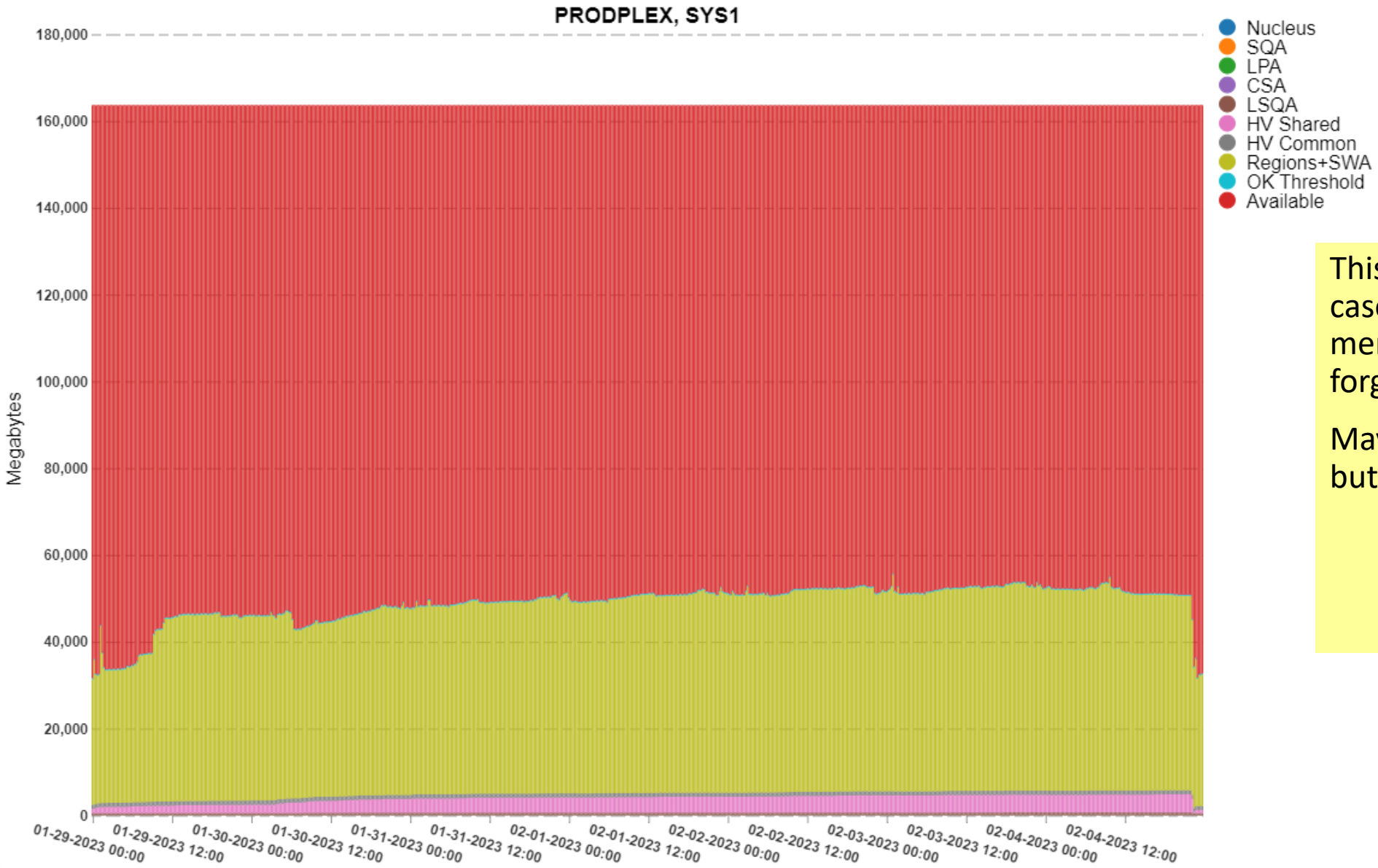
So, if you avoid I/O…

- Performance is improved, making the users happier
  - To the degree that users are happy with better performance
- Reduce CPU consumption, possibly reducing software cost
  - Financial people are only happy with zero cost, but maybe they'll be less unhappy?
- Possibly make better use of unused resources, i.e. memory
  - Management will find something else to critique

- So why do I keep coming across LPARs like these…

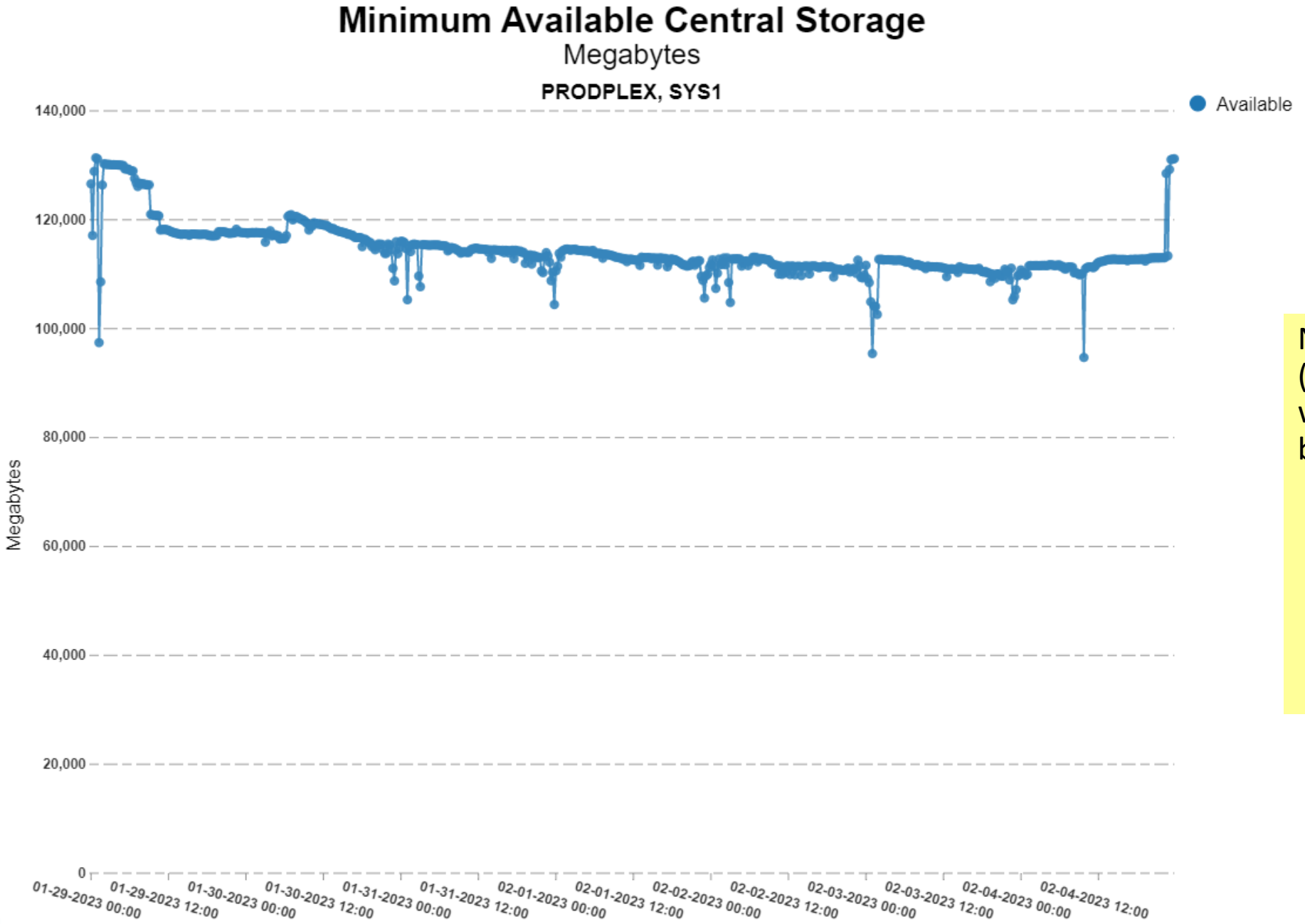
Central Storage Areas

Average Megabytes



This looks like the classic case of “we did the 3x memory deal and then forgot to use it”.

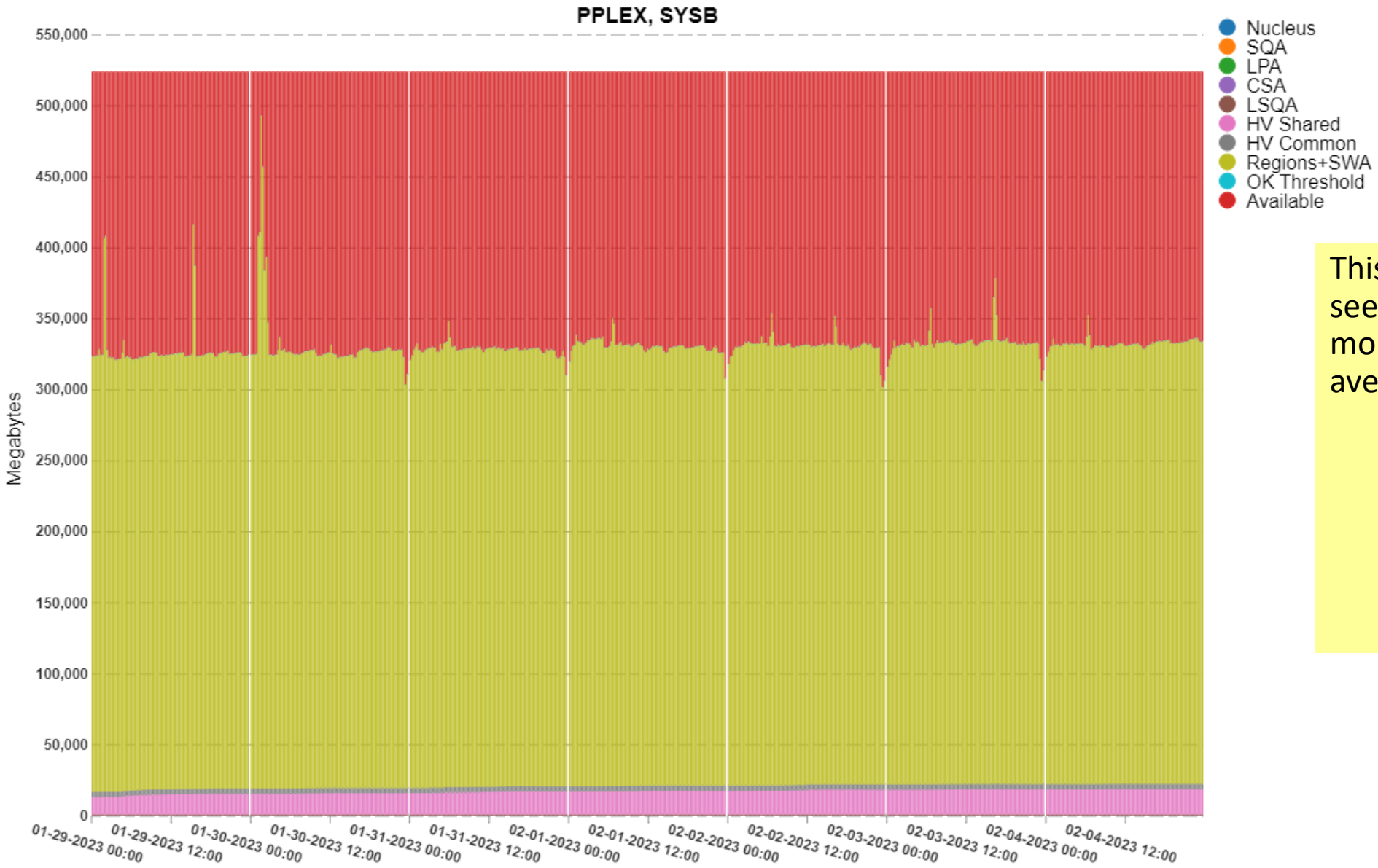
Maybe it’s being used but just very briefly?



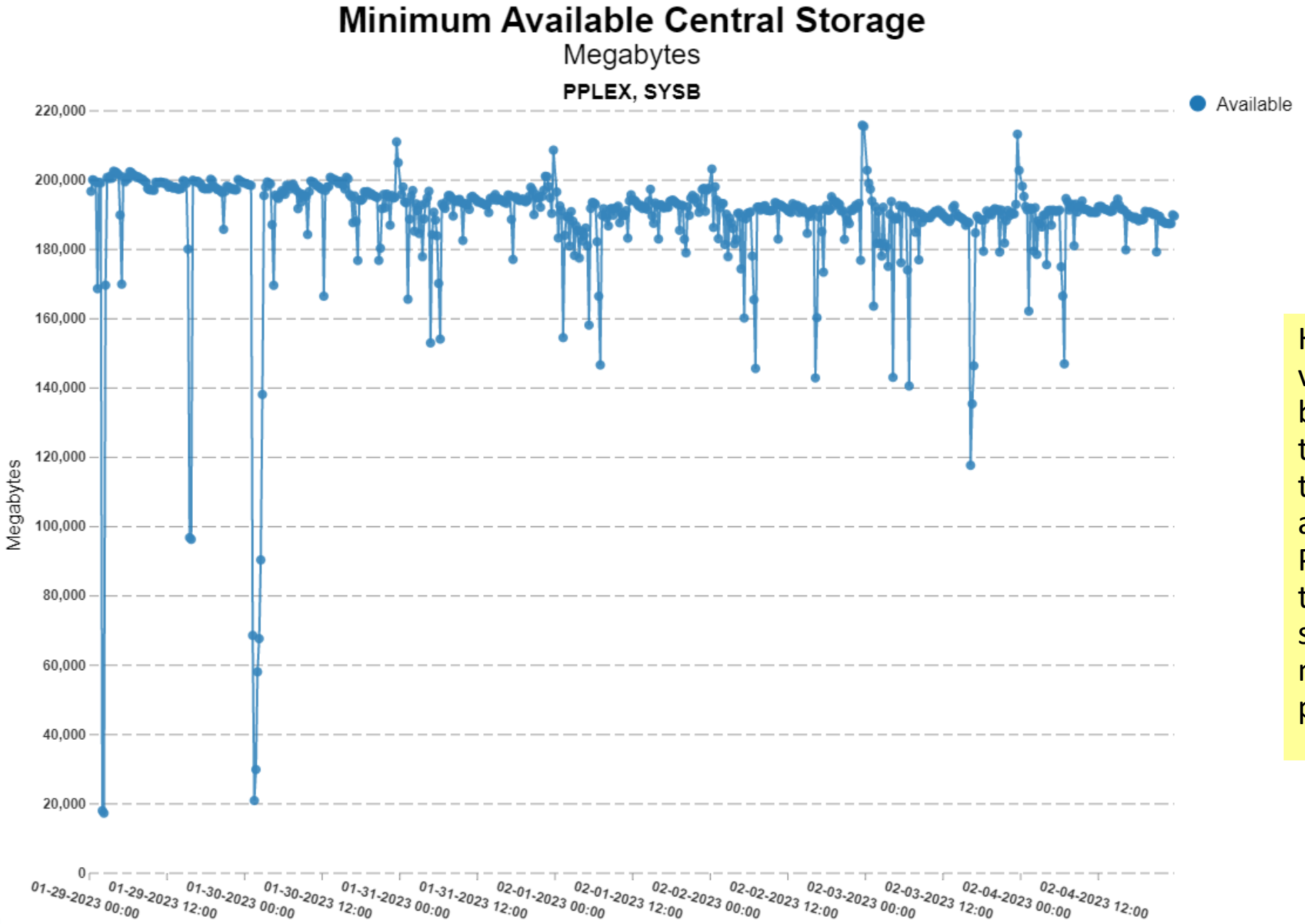
Nope: minimum rarely (over the course of a week) rarely drops below 100GB free.

Central Storage Areas

Average Megabytes



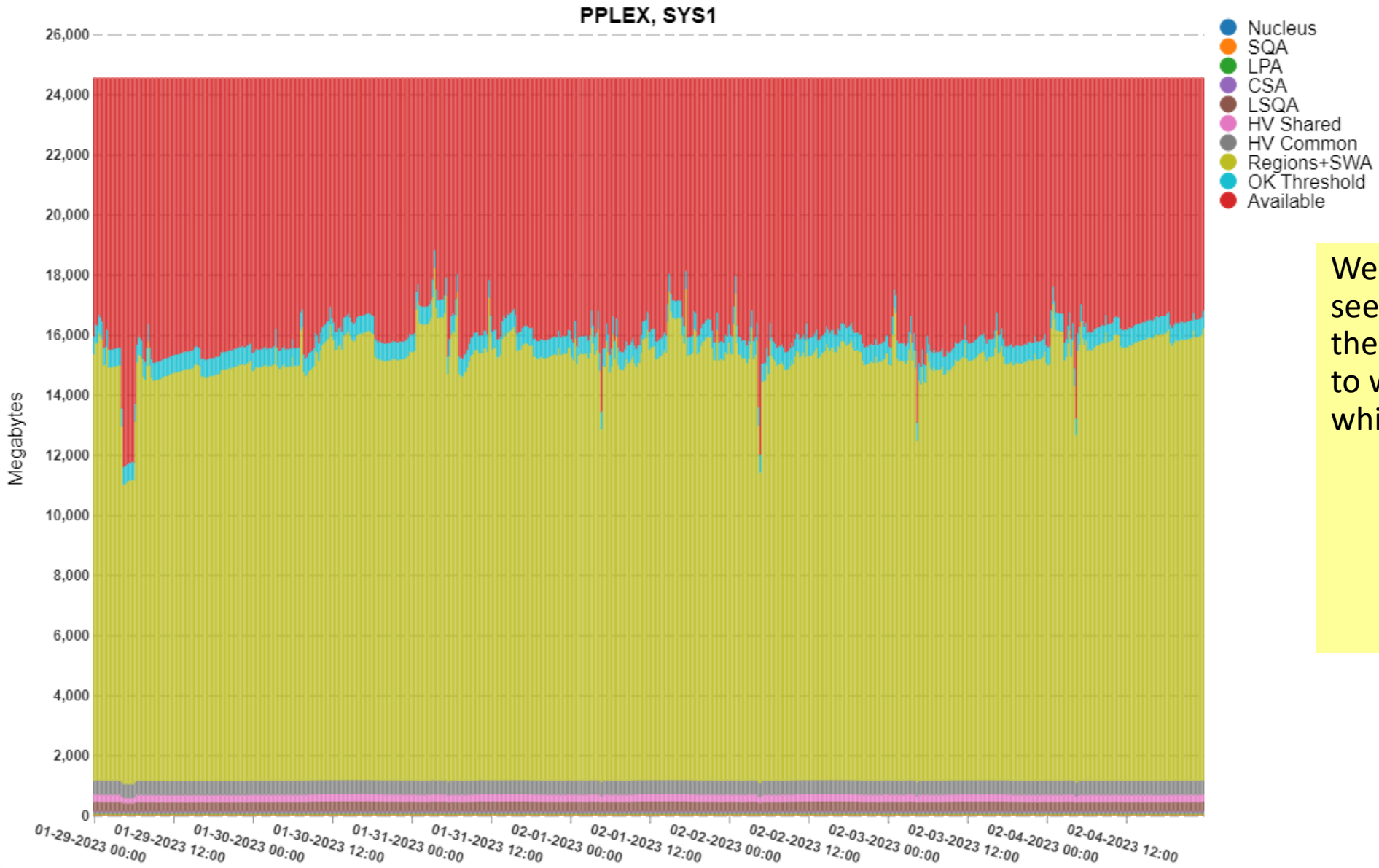
This system is larger but seems to have even more free storage on average.



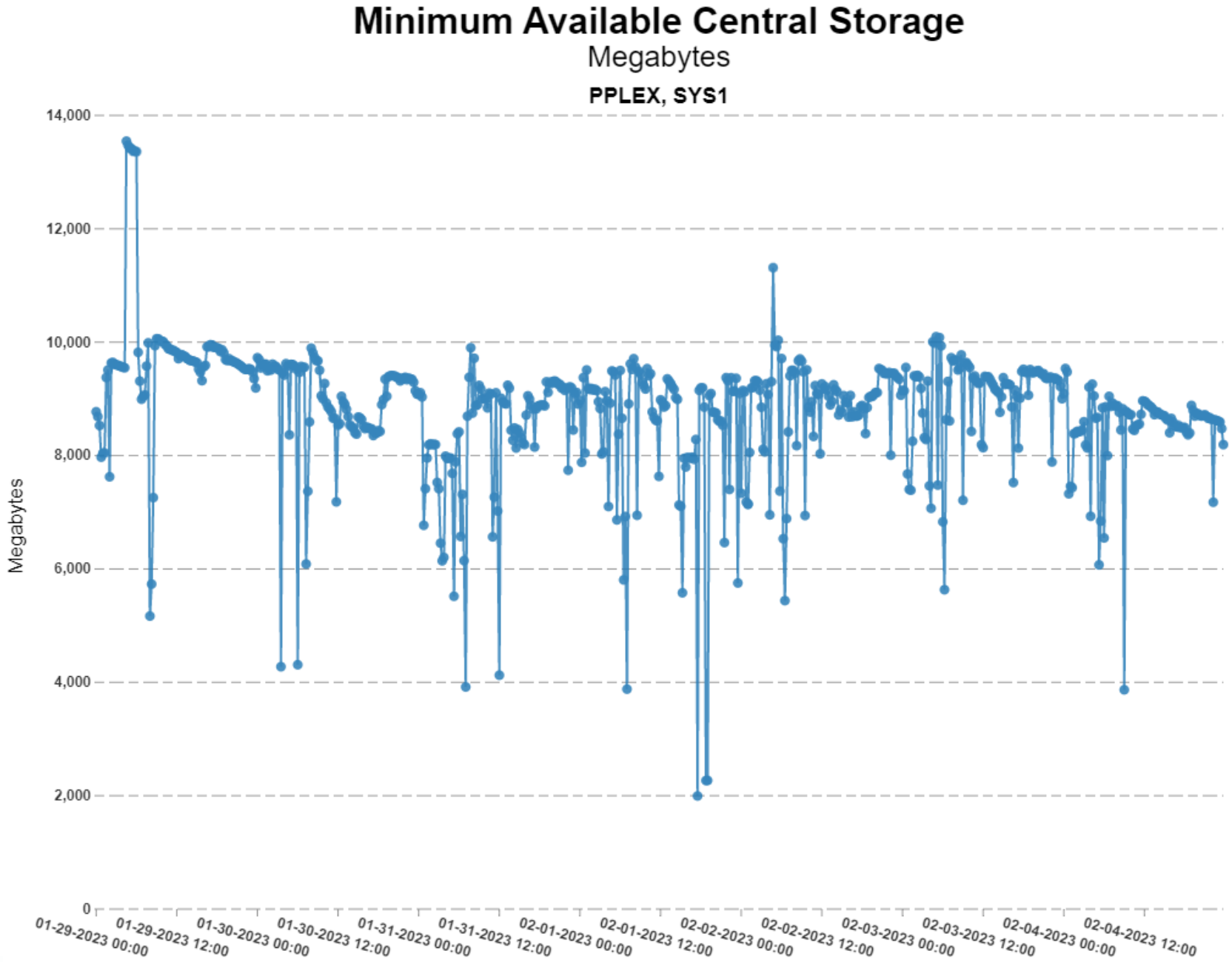
Here the minimum values do show it is being briefly used and there are some events that might be sort activity or dumps. Probably not a big deal though to use some significant chunk of that memory for other purposes.

Central Storage Areas

Average Megabytes



We do sometimes still see small LPARs. Here there's only about 8GB to work with on average, which is a lot tighter.



And the minimums are even smaller.

Of course, there may also be extra memory on the machine not defined to the LPAR.

But on the face of it, it's going to be harder to make good create use of "large" memory if your memory isn't large.

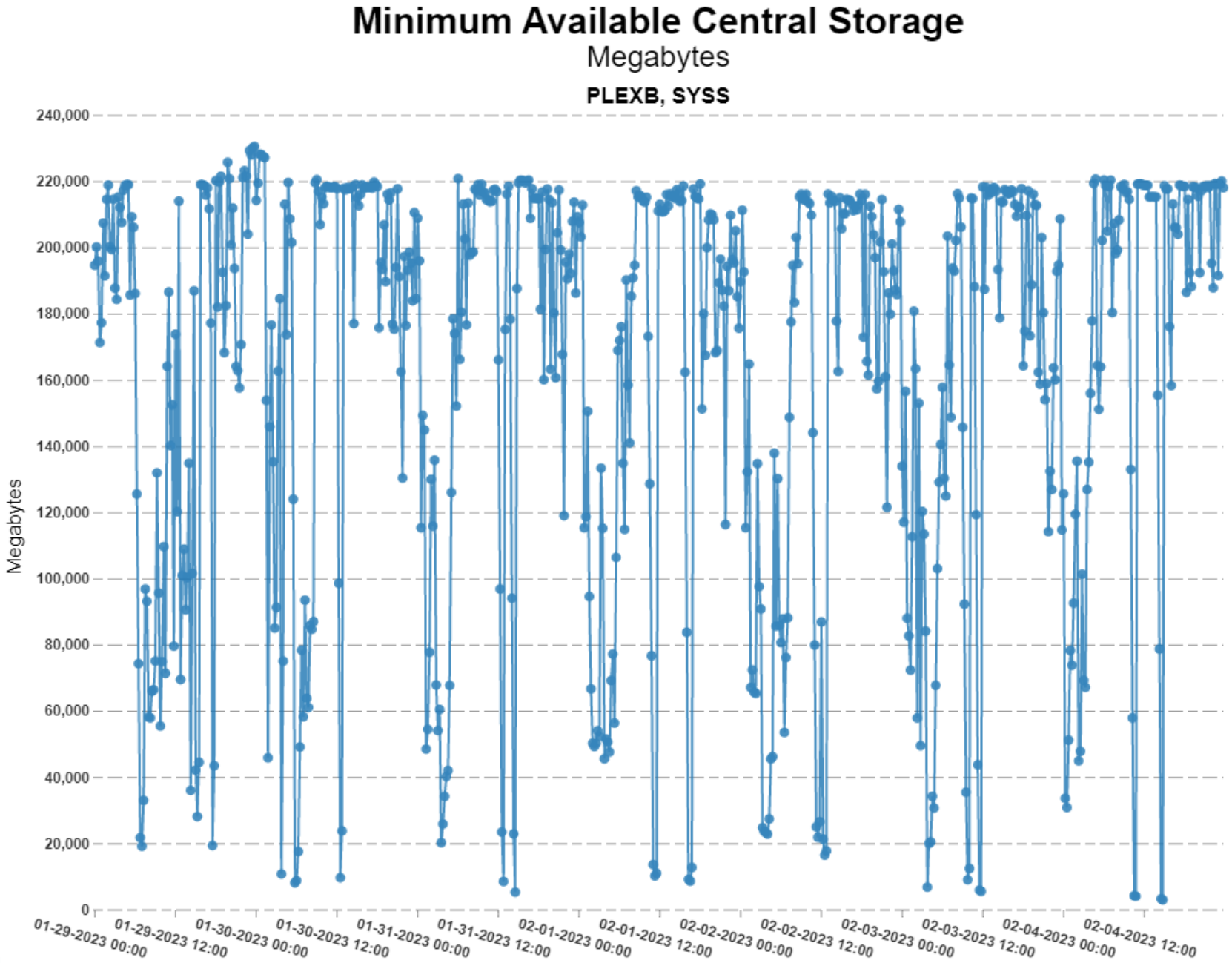
To be fair….

- We didn't look at what workloads might be improved by taking advantage of large memory
  - Maybe all those LPARs are already doing very little I/O
- Maybe they really do have plans for that memory
- In general though: if you have memory use it!
  - And if you're still using LPARs the same size as they were 10 years ago… is your business still the same size?
- I'm not complaining about everybody though…

Central Storage Areas

Average Megabytes

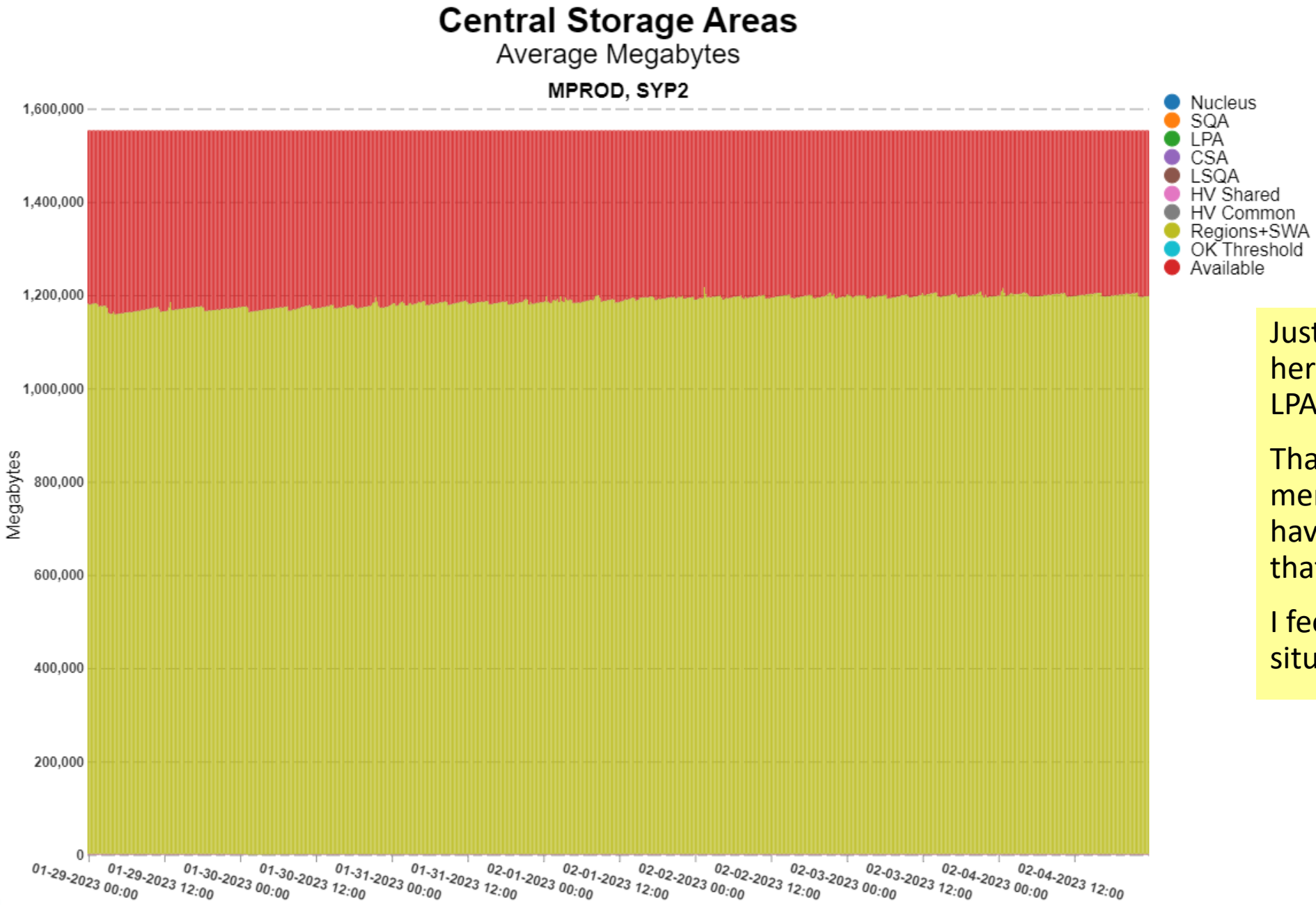
Here's a fairly good size LPAR but they are regularly making use of the available memory even if it's not used consistently.



Their minimum values do get pretty low at times, but as long as paging is under control, that's probably ok.

In reality, this system does do a small bit of paging (single-digit pages/second) to Virtual Flash Memory, so not a big deal.

I feel good about this situation.



Just for completeness, here's a pretty large LPAR.

That's a lot of data in memory, and they still have some room to grow that workload!

I feel good about this situation too.

Thoughts and Recommendations

Things to look into

- Take inventory of what you have
  - Do your LPARs have unused memory?
  - Is there purchased memory on the CEC that hasn't been allocated to LPARs?
- Do you have significant I/O rates?
  - "Significant" is of course relative
- Do you have applications whose performance is I/O limited?
  - "Doing I/O" is not always the same as "limited by I/O"
- Are there business problems you haven't even tried to solve?

“Healthy” Available Memory

- One theory is to try to have average available be about equal to your dump MAXSPACE setting
  - DB2 recommends: MAXSPACE = DB2 address spaces + common – Buffer Pools
- Large dumps can be problematic
  - SCM/VFM can help significantly (discussed shortly)
- But do you really want to reserve that memory for a hopefully rare event?
  - Maybe: depends on your environment
  - If you find those events are not rare, may be good time to bring some reserved storage online
  - Remember that you can have some reserved storage that’s “shared” between all the LPARs so you can bring it online if you suddenly find yourself in trouble

Virtual Flash Memory

- On z15+ replaces Storage Class Memory
  - Is now same memory, instead of locally attached SSDs
  - Orderable in 512GB increments (up to 6 TB)
- I'm generally a fan of this
  - Faster dump processing
  - Faster paging
  - Easier Aux Storage management
- Still would prefer to not page, but… this paging hurts much, much less
  - Paging will still cost some CPU
  - Don't plan to use this as a replacement for real storage

IEAOPTxx settings

- DB2 buffer pools *should* be fixed and can be a largish percentage of the total storage for some systems
- IRA405I(2) – Percent of total storage fixed triggering warning message
  - Default is 50%
- MCCFXTPR - Specifies the percentage of online storage that might be fixed. SRM uses this threshold to determine when a shortage of pageable storage exists.
  - On small systems (less than 320 GB), the target is 80 percent. On large systems (more than 320 GB), the target is total storage minus 64 GB.
  - May need to adjust this if trying to make good use of memory such that you're leaving less than 64 GB free and the majority of your storage is fixed

Some ways more memory can help

- Obvious answer is more buffers
  - In DB2, you can “pin” objects in buffer pools too
  - Review your buffer pool design in light of having lots of memory
    - Even if you don’t have lots today, dream of the possible for the next upgrade
- Even better: make application changes to avoid I/O
  - Not even calling the database to do the I/O even faster and less CPU
    - I.E. fetch data once and buffer it instead of re-fetching it.
  - Products also available to help with this
- There’s other DB2 memory consumers that could potentially be expanded
  - In-memory sorts, RID Pool, Fast Traversal Blocks, etc.
- Don’t short-change your JVM heaps
  - GC overhead often negligible until it falls off the cliff: make sure the cliff is far away

In summary…

Parting thoughts

- We've a come a long way
  - In size of memory
  - In how memory is managed
- Don't manage your memory like it's 1999!
  - Or even 2009!
- Large memory can improve performance and reduce CPU consumption
  - You pay for memory once, you pay for CPU (via software costs) continuously

Questions??

Your feedback is important!

SHARE
EDUCATE • NETWORK • INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation