I/O, I/O: It's Home to Memory We (should) Go

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!)

Optimizing z/OS system and application performance today is all about having data as close to the processor as possible when it's needed. Despite some incredibly fast I/O times today, memory is still orders of magnitude faster and (still) the only good I/O is no I/O. Keeping data in memory can not only improve application performance but reduce CPU consumption as well. Fortunately, many systems today have sufficient memory to make potentially significant improvements. But memory is not infinite so how do you find the best opportunities for improvement?

In this session Scott Chapman will discuss data sources and methods to find, evaluate, and implement I/O reduction opportunities. You're sure to leave with good ideas that will help you eliminate unnecessary I/O and improve application and system performance.

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!

- Essential z/OS Performance Tuning
  - September 16-20, 2024 (tentative)
- Parallel Sysplex and z/OS Performance Tuning
  - August 20-21, 2024 (tentative)
- WLM Performance and Re-evaluating Goals
  - February 19-23, 2024 (tentative)
- 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!

Agenda

- What inspired this presentation
- Why the only good I/O is no I/O
- Identifying I/O reduction opportunities
- Ways to reduce I/O
  - Db2 (and possibly other databases)
  - VSAM
  - Catalog
  - Sort

Trivia

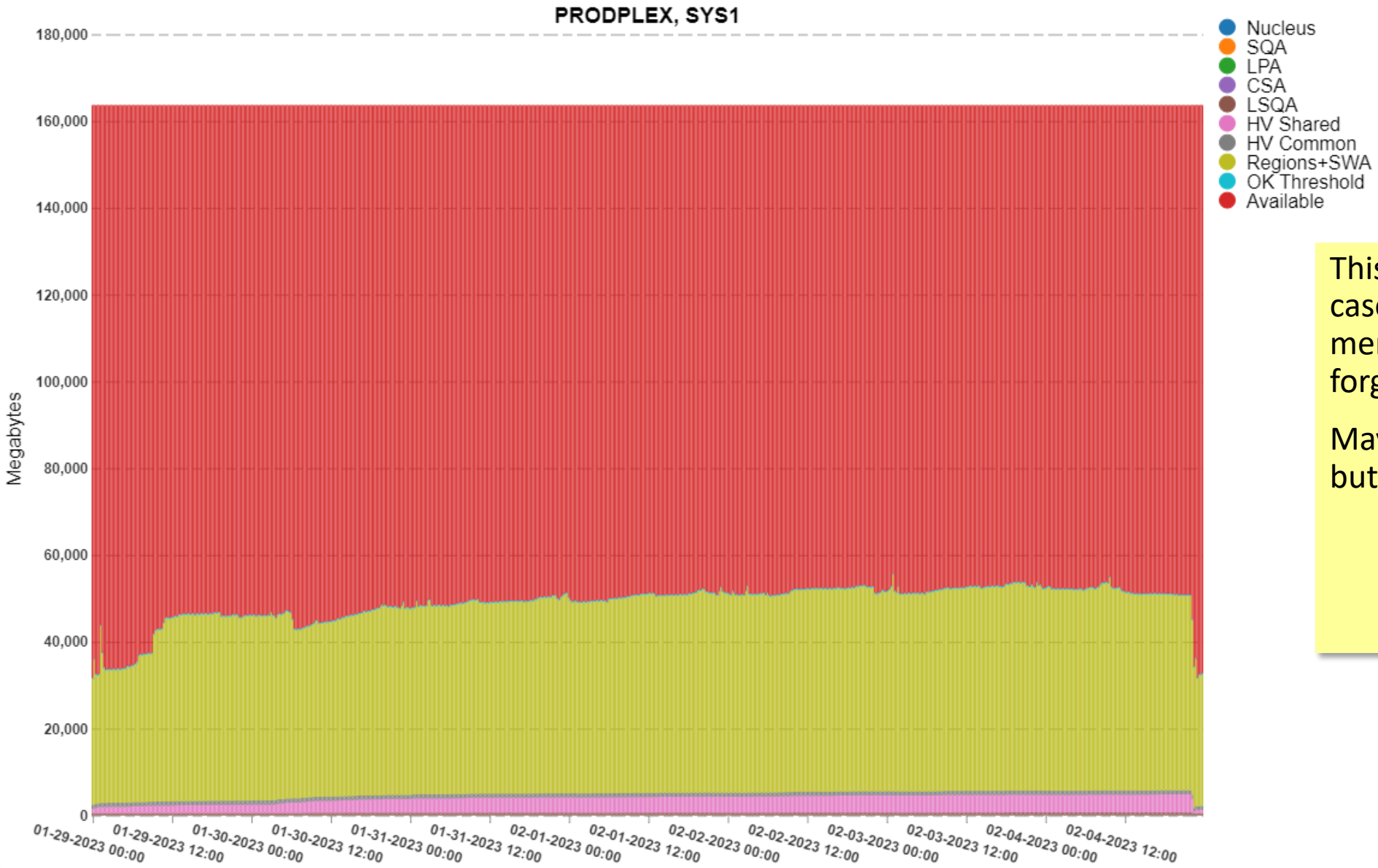
- The original song sung by the Seven Dwarfs does not contain:
  - Off to work we go
- It's actually:
  - Heigh-ho, Heigh-ho
    It's home from work we go
- Doesn't that sound better?
- But… they later return to work and briefly do sing "Off to work we go"
  - Sigh

Inspiration behind this talk

- Many customers now have a memory-rich environment
- If you're relatively short on z/OS memory either:
  1. You're already actively leveraging large memory
  2. You have memory on the box that you've not assigned to the z/OS LPARs
  3. You didn't consider the opportunity cost of configuring less memory
  4. You have a lot of LPARs
- The third may be the most common, especially on customers on the "small" machines
  - z16 T01 – minimum orderable memory is 512 GB (same as z15)
  - Z16 T02 – minimum orderable memory is 64 GB (I wish IBM would change this)
- Second sometimes happens because they got a 512GB machine and just configured the LPARs same as they were on a smaller machine
- First is probably least common

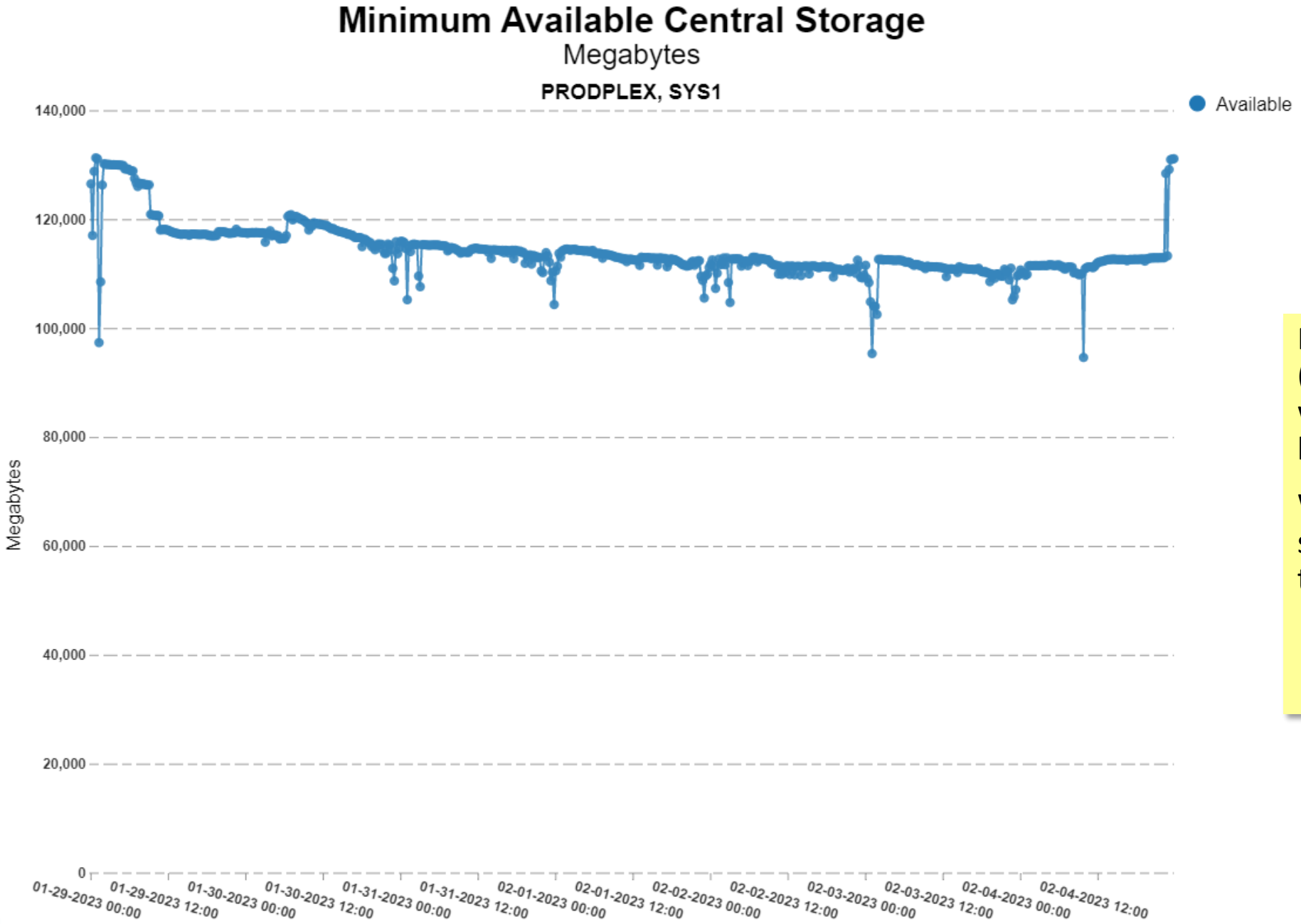
Central Storage Areas

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.

We see this sort of situation far more often than I think we should.

The only good I/O is no I/O

- Yes, I/O can be really fast today, but it still takes time
  - But memory is really^2 or really^3 fast
  - I/O: hundreds of microseconds
  - z/Hyperlink: tens of microseconds
  - Memory: fraction of a microsecond
- 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

But what about DASD cache?

- Controller cache is good, but somewhat limited
  - IBM DS8900F – max cache size is 4.3 TB
  - Hitachi DS 5600 – 2 TB/controller block up to 6 TB
  - Dell PowerMax – 15* TB on PowerMax 2500 and 45* TB on PowerMax 8500
    - But those are raw numbers and there's cache mirroring
- Processor can have huge memory in comparison
  - z16 A01 – Max 40 TB
  - z16 A02 – Max 16 TB
  - z/OS LPAR – 16TB (z/OS 3.1)
  - Recent review of our customers' configs showed largest LPAR was 4 TB!
    - LPARs > 1TB certainly becoming more common

What about zHyperLink?

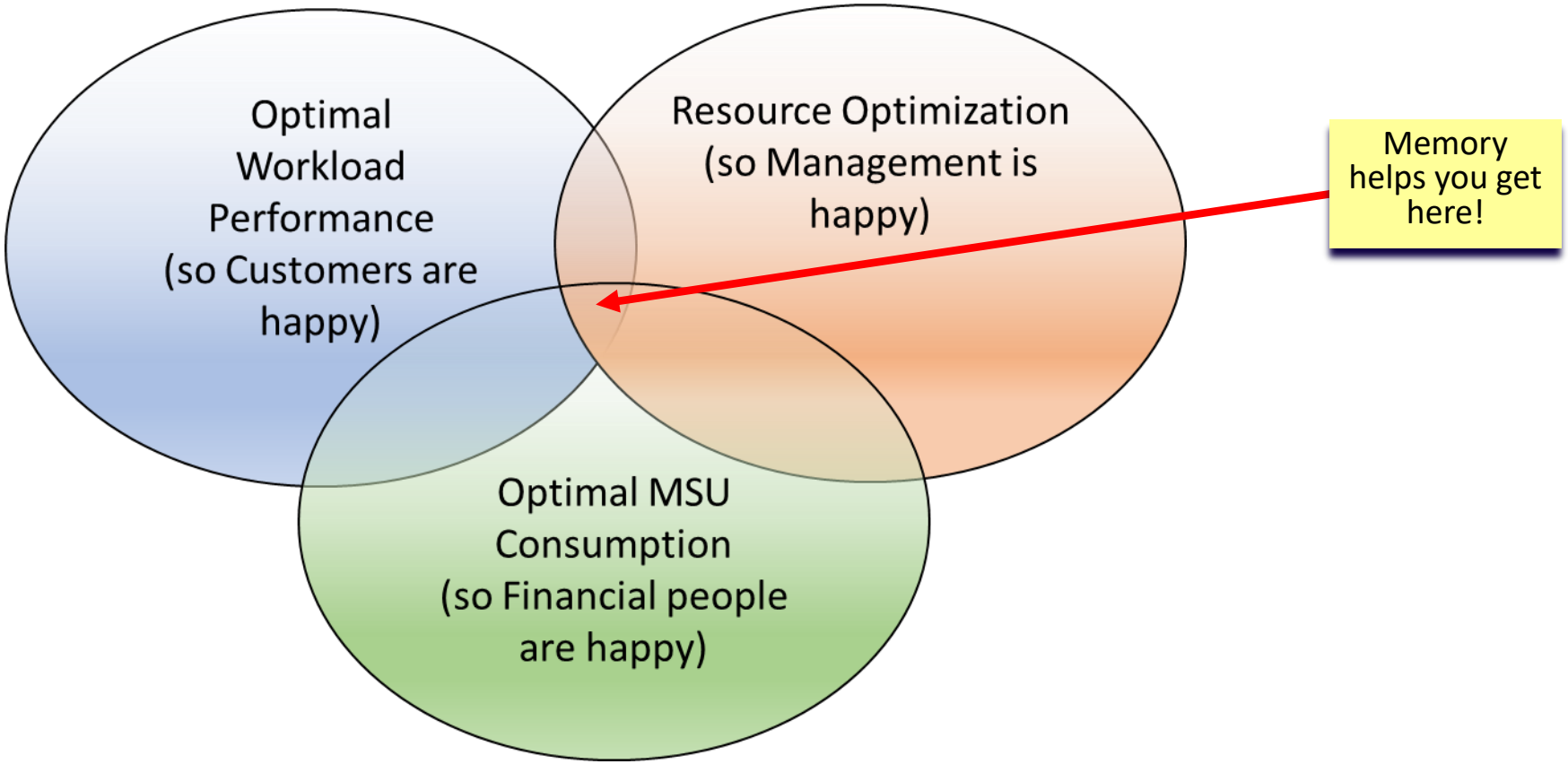
- zHyperLink promises response times on order of 10s of µs for cache hits
  - Also, no I/O interrupt delay because processor spins while waiting on the I/O
  - Some of I/O overhead of spinning offset by improved CPU L1/L2 cache hits
- This is still much slower than main memory
  - For z16 Tellum chip, on-chip L3 is said[1] to be 12ns latency (0.012µs)
  - From the published RNI formulas, impact of memory access is scaled at 13.5x L3
  - So, roughly, a L4 processor cache miss may be on the order of 162ns (0.162µs)
    - That's a rough but probably fair estimate, although memory on a different book will be longer: possibly 4x longer based on the RNI formula
    - Of course if the memory to be accessed is in a closer processor cache, it will be much faster!
- Reading 4K from memory probably *at least* 15x faster than zHyperlink
  - Means 1/15th the CPU time impact too!

1: https://www.anandtech.com/show/16924/did-ibm-just-preview-the-future-of-caches

So, if you avoid I/O…

- Performance is improved, making the users happier
  - To the degree that users are happy with better performance
  - (And the degree that they notice)
- Possibly 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 avoid "unnecessary" I/O
  - Is any read I/O "necessary"? (Yes, but… maybe pretend not!)

Performance Management Thoughts



Finding opportunities

Finding opportunities in SMF data

- I/O related information is all over in the SMF records
  - Type 14/15 – Old DD-related I/O
  - Type 30 – Summary I/O at job/step
  - Type 42 – Volume and dataset level I/O
  - Type 64 – VSAM Status
  - Type 71 – Paging
  - Type 72 – I/O by service class/report class
  - Type 74 – Volume level I/O details
  - Type 75 – I/O by page dataset
  - Type 100-102 – Various Db2 details, including Db2 I/O
  - Type 110 – CICS details, including CICS I/O
  - Others – Sort, Vendor-specific DASD measurements, etc.

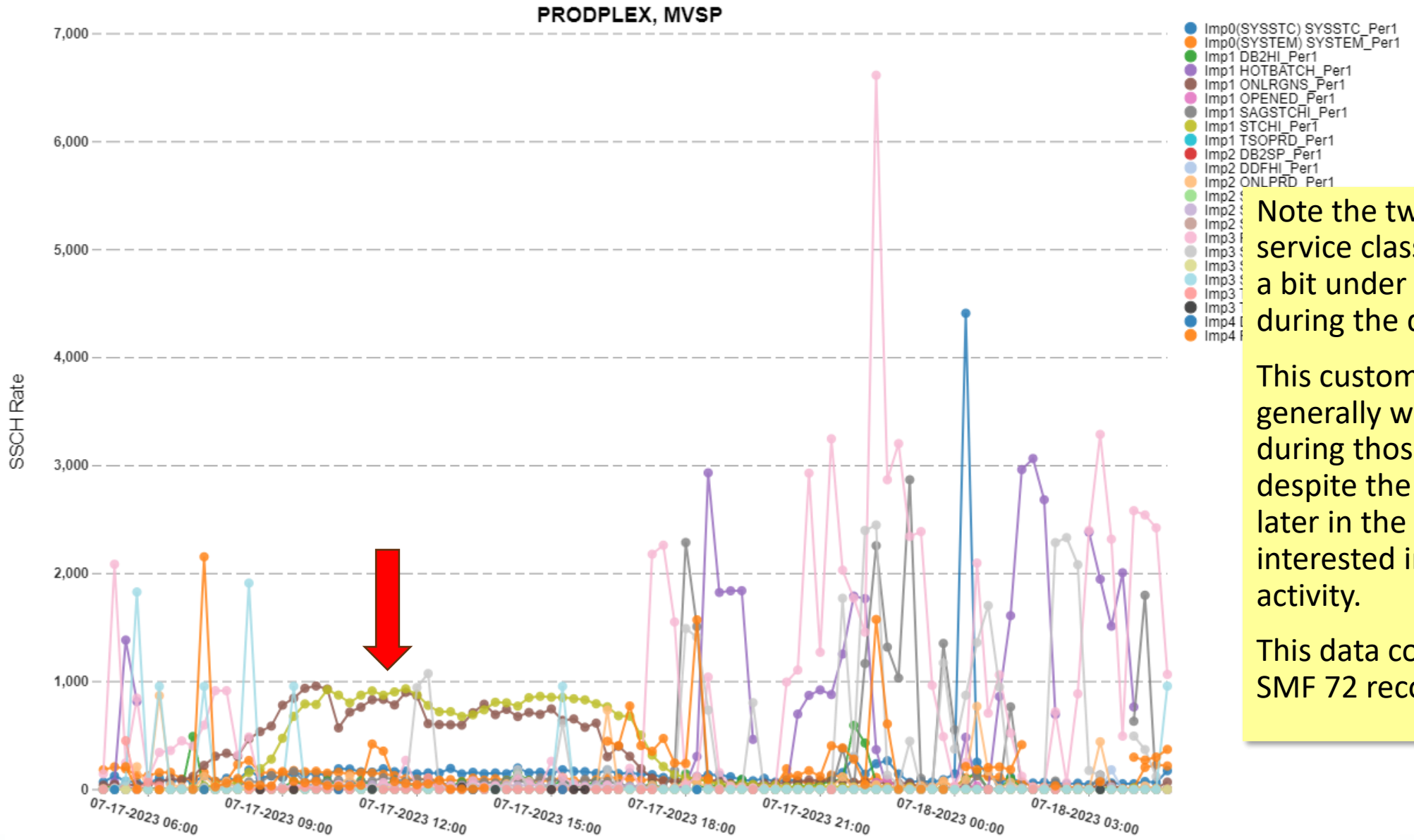
Particularly interesting and “easy”

SMF 42: not quite ideal, but good

- Has both interval and "close" statistics at dataset level
  - Interval statistics controlled by SMF interval: ideally 5, 10, or 15 minutes
    (If your interval is >15 minutes please change to 15, and sync those intervals!)
  - But intervals are not written if no I/O, so final close record may cover larger than expected timeframe
- Has some I/O response times at microsecond level precision
  - Some also at 128-microsecond (0.128ms) precision like RMF/CMF
- Has details about cache hit/misses
- Overall, 42.6 excellent source for understanding what is doing I/O!

Example 1

WLM DASD - I/O SSCH Rate by Service Class Period Over Time



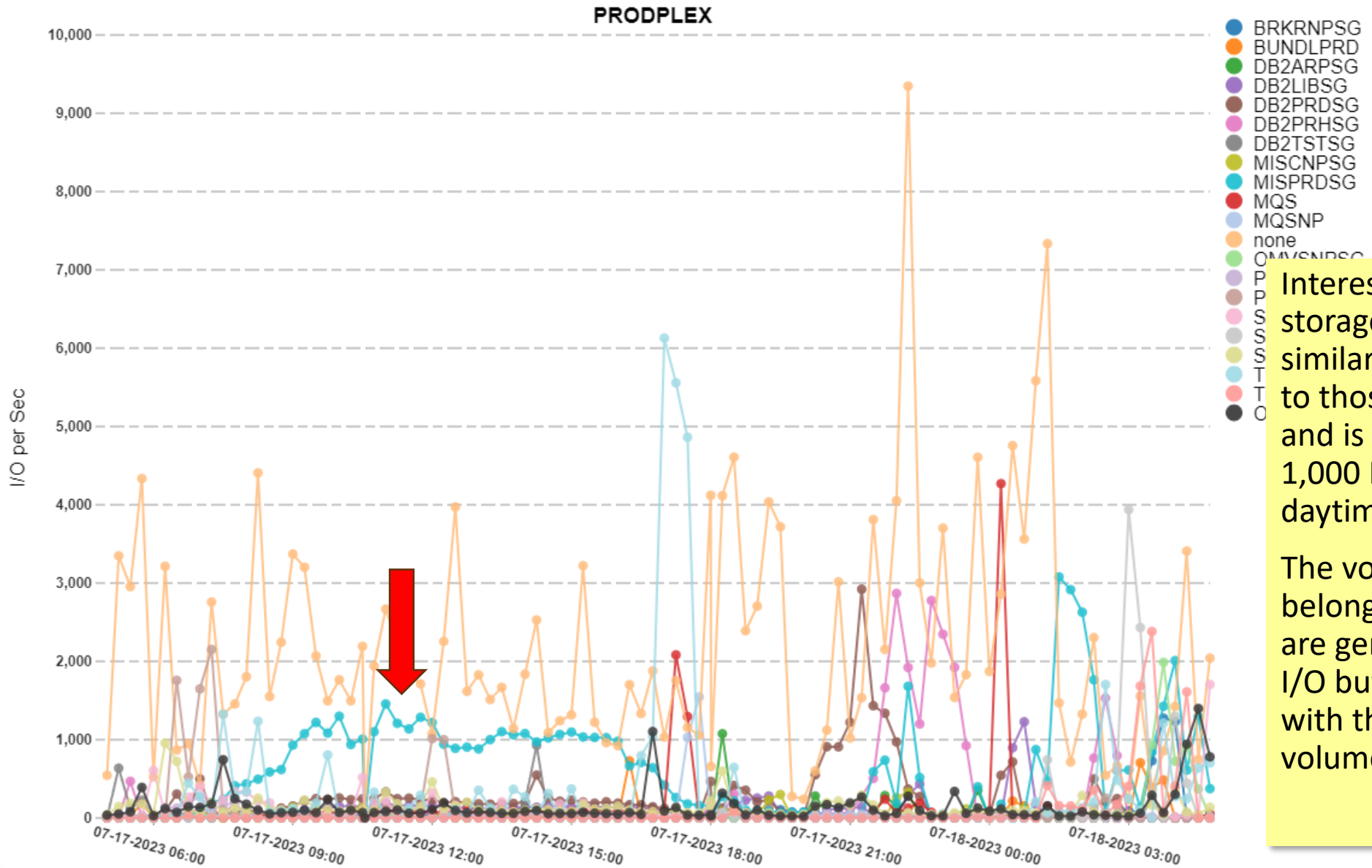
Note the two importance 1 service classes each doing a bit under 1000 IOPS during the daytime hours.

This customer's CPU generally was highest during those hours, so despite the higher I/O rates later in the day I was most interested in that daytime activity.

This data comes from the SMF 72 records.

Storage Group - I/O Rate for Top Storage Groups

Top Storage Groups

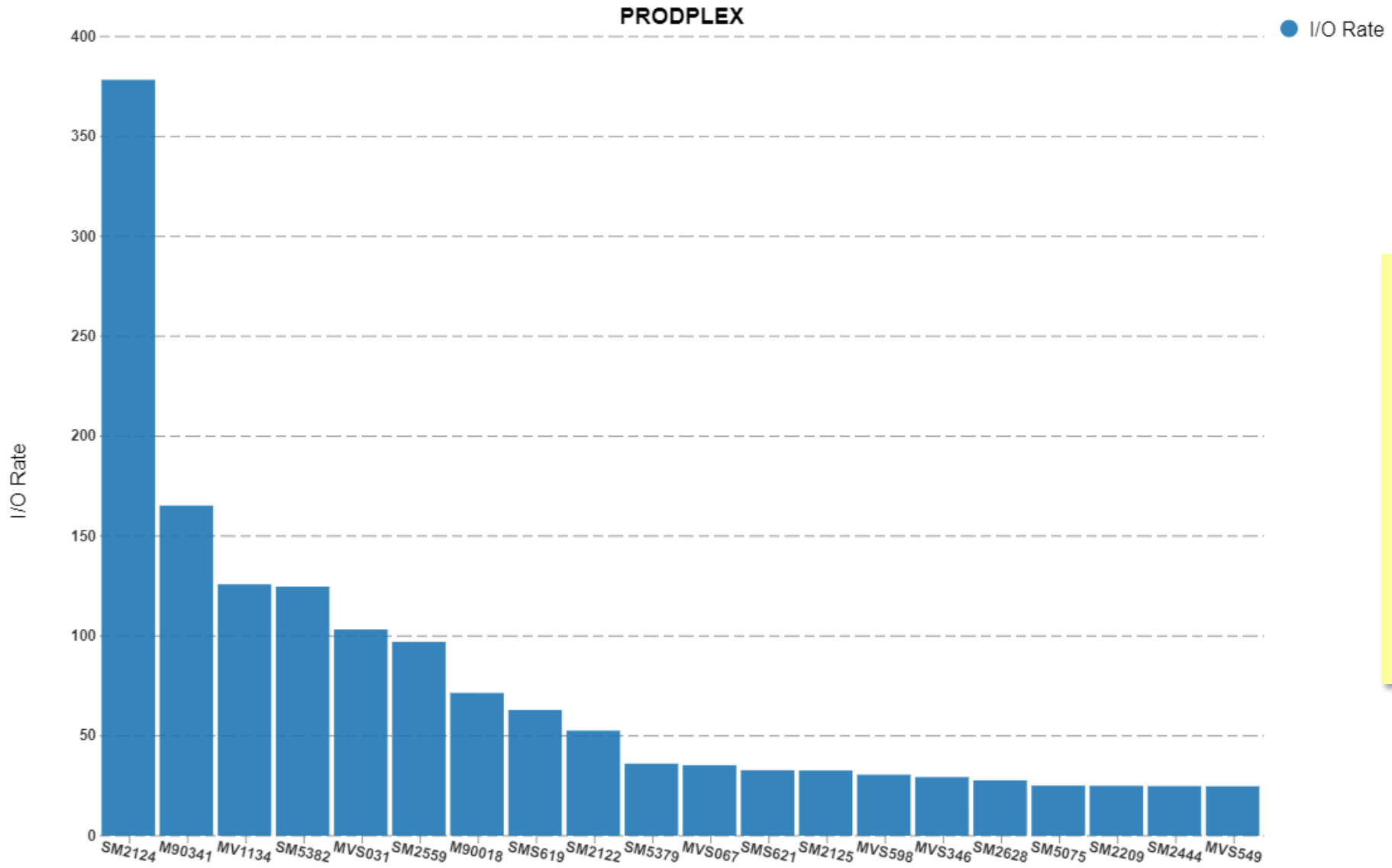


Interestingly, the MISPRDSG storage group shows a similar profile of I/O activity to those two service classes and is actually a bit over 1,000 IOPS during those daytime hours.

The volumes that don't belong to a storage group are generating even more I/O but I thought I'd start with the storage group volumes.

LVs with Highest I/O Rates

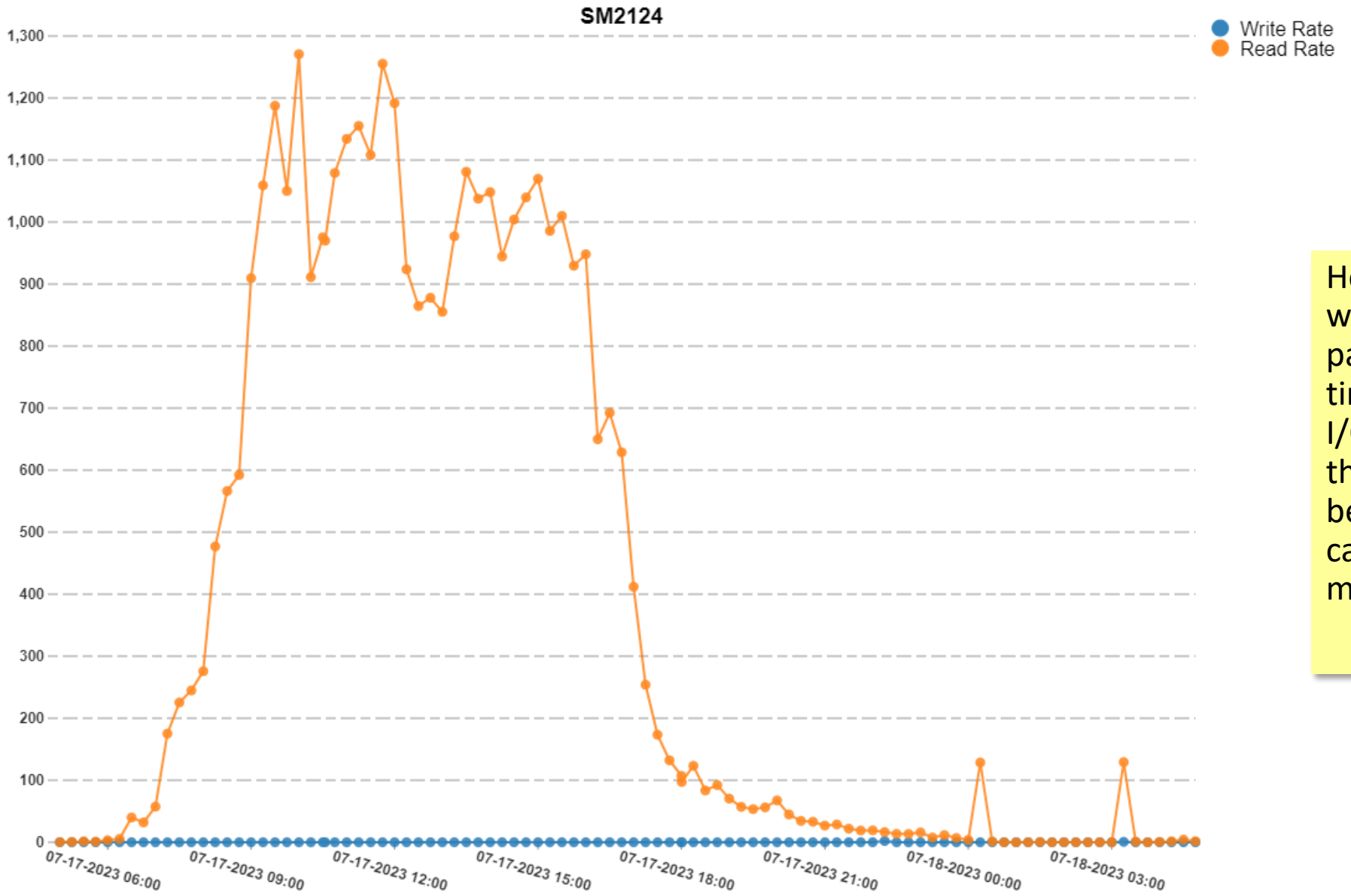
(Averaged Over Period of Study)



So I wondered if any of these top volumes would just happen to be in that storage group of interest. As it turned out, in fact SM2124 was!

Note this is an average I/O rate over 24 hours.

Logical DASD Volume Explorer



Here's the read and write rate for that particular volume over time. Virtually all of the I/O is read I/O, implying that perhaps that could be avoided if we could cache that data in memory.

Volume Details

2023-07-17

smfdate	Storage Gro…	Volser	Device Number	LCU ID	Logical DevType	Capacity GiB	Alloc GiB	Free GiB	Free %	Frag Index	Free DSCBs	Free VIRs
Select Fil	Select Filter	Selec	Select Filter	Sele	Select Filter							
2023-07-18	MISPRDSG	SM2124	961F	96	3390-30	25.932	1.440	24.492	94.000	101.000	90,623.000	1,907.000

In this case we also have at least the volume-level DCOLLECT data from the customer so we can see that surprisingly, there's less than 1.5 GB of data stored on the volume!

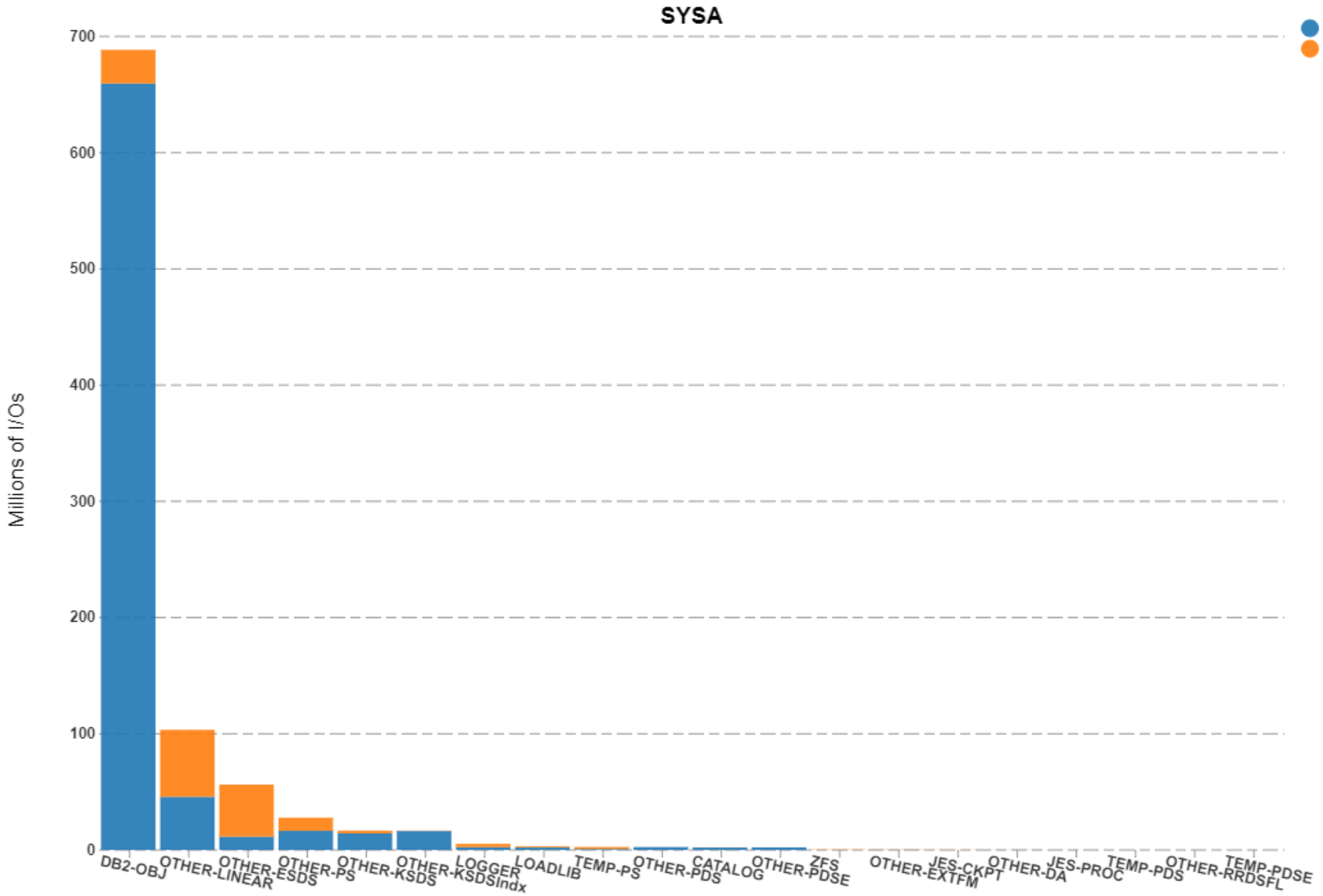
Lacking the SMF 42.6 records, we don't know what datasets are doing the I/O but 1.5 GB is small enough that they could easily store all of that in memory. Doing so could get rid of a significant chunk of their daytime I/O.

Example 2

Coming from the SMF 42 Perspective

Top Dataset I/O Counts by Dataset Usage

Total I/Os for Study Period

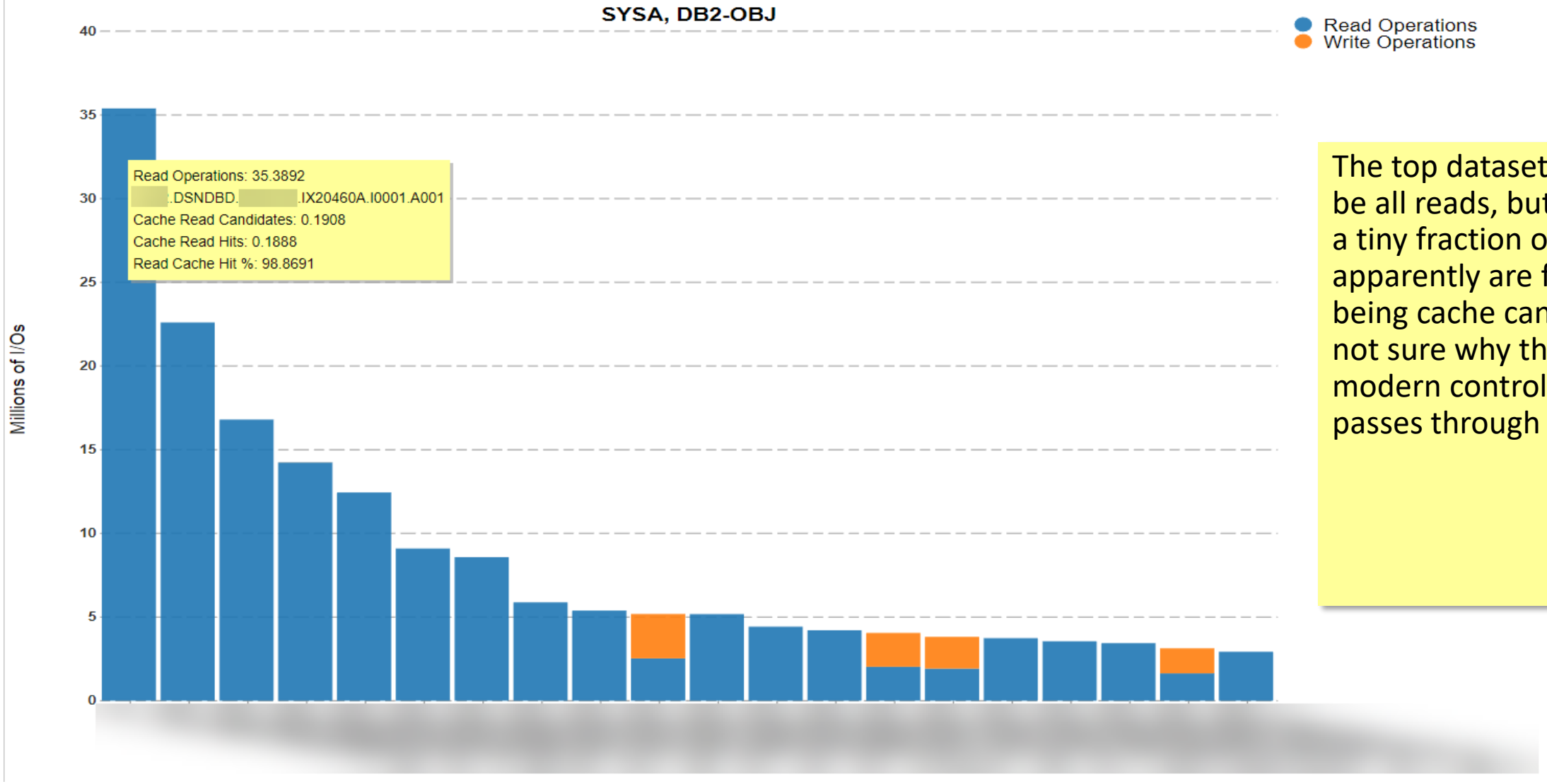


This reports looks at the total I/O over (in this case) a day from the SMF 42 records and breaks it down by reads vs. writes and by what the dataset is (probably) used for.

This site is not unusual: the vast majority of the I/O is reading from DB2 objects.

Top Dataset I/O counts by Dataset Usage

Total I/Os for Study Period



The top dataset appears to be all reads, but oddly, only a tiny fraction of those apparently are flagged as being cache candidates. I'm not sure why that is. But in modern control units all I/O passes through cache.

Top Datasets by Cache Read Hits

2023-07-17

Date	Usage	DS Name	M Cache Read Hits	Cache Hit Pct	Read MiB	Allocated MiB	Read-Allocated Ratio	Read O...	Write Ops	42.6 Records	Volume
Select Fil	Select Filte	Select Filter									
2023-07-17	DB2-OBJ	.DSNDBD. .IX20460A.I0001.A001	0.189	98.869	4,201,991.004	569.841	7,373.973	35.389	0.000	54.000	2.0
2023-07-17	DB2-OBJ	.DSNDBD. IX08956B.I0001.A001	20.108	97.188	664,607.695	2,361.233	281.466	22.601	0.001	251.000	8.0
2023-07-17	DB2-OBJ	.DSNDBD. /02.TS06435.J0001.A001	1.450	98.796	1,880,165.453	569.841	3,299.457	16.798	0.000	245.000	5.0
2023-07-17	DB2-OBJ	.DSNDBD. IX00854E.I0001.A001	14.170	99.960	69,569.031	1,339.897	51.921	14.244	0.006	10.000	4.0
2023-07-17	DB2-OBJ	.DSNDBD. IX08956B.I0001.A002	11.436	97.651	288,683.320	1,159.947	248.876	12.444	0.004	241.000	8.0
2023-07-17	DB2-OBJ	.DSNDBD. .TS01452.J0001.A001	8.968	99.152	71,634.121	1,203.719	59.511	9.097			
2023-07-17	DB2-OBJ	.DSNDBD. TS07315.I0001.A001	7.206	98.403	696,717.207	4,175.324	166.865	8.580			
2023-07-17	DB2-OBJ	.DSNDBD. .TS20310.I0001.A001	5.076	96.649	135,041.418	4,720.846	28.605	5.882			
2023-07-17	DB2-OBJ	.DSNDBD. TS08957.J0001.A001	4.991	99.302	228,975.305	4,721.657	48.495	5.390			
2023-07-17	DB2-OBJ	.DSNDBD. TS00854.I0001.A014	3.724	89.910	198,572.117	1,957.563	101.438	5.179			
2023-07-17	DB2-OBJ	.DSNDBD. .TS01451.J0001.A001	1.365	94.870	332,869.598	306.401	1,086.384	4.428			
2023-07-17	DB2-OBJ	.DSNDBD. TS07315.I0001.A001	3.588	99.894	499,237.086	4,377.969	114.034	4.210			
2023-07-17	DB2-OBJ	.DSNDBD. TS07315.J0001.A001	3.121	99.978	417,840.731	4,341.493	96.244	3.551			
2023-07-17	DB2-OBJ	.DSNDBD. .TS07292.I0001.A001	2.965	99.891	77,346.273	4,722.468	16.378	3.444			
2023-07-17	DB2-OBJ	.DSNDBD. .TS17613.I0001.A001	0.156	94.461	345,553.938	284.516	1,214.534	2.930			
2023-07-17	DB2-OBJ	.DSNDBD. .TS07292.J0001.A002	2.255	99.791	71,282.477	4,721.657	15.097	2.738			
2023-07-17	DB2-OBJ	.DSNDBD. TS02809.J0001.A009	2.170	89.721	77,474.367	3,465.251	22.358	2.688			
2023-07-17	DB2-OBJ	.DSNDBD. TS02813.J0001.A004	2.085	95.299	63,996.520	3,465.252	18.468	2.609			
2023-07-17	DB2-OBJ	.DSNDBD. .TS06562.I0001.A001	2.316	96.014	37,477.902	1,738.704	21.555	2.499			
2023-07-17	DB2-OBJ	.DSNDBD. TS17820.I0001.A001	1.968	92.237	73,247.258	2,691.953	27.210	2.451			
2023-07-17	DB2-OBJ	.DSNDBD. .TS17613.J0001.A001	0.062	93.959	295,867.445	284.516	1,039.899	2.446			

This table report joins the SMF 42 data with the DCOLLECT data to get the total allocated size (summed across multiple volumes if necessary) of the datasets.

Note there's little write activity and a number of these datasets are only a few GB.

Even if they can't all go into memory, probably some can, saving 10s of millions of I/Os.

What to do about busy datasets

Db2

- Basically: pin the objects in a buffer pool
- Make BP big enough to hold the entire object(s)
  - Db2 systems with 100s of GBs of buffer pools are increasingly common
- Optionally set PGSTEAL(NONE)
  - Indicates to Db2 you believe the BP is big enough to hold all of the object(s) in the BP
    - Doesn't mean that Db2 won't steal pages from it if need be
    - Doesn't mean that the pool is read-only
  - Db2 will use async prefetch to pre-load the objects on first reference
  - Note: Don't use PGSTEAL(NONE) and FRAMESIZE(2G) together.
    - NONE & 2G will be treated as LRU & 2G. Use NONE & 1M instead!
- Remember to page-fix your production BPs (at least, maybe dev/test too)
  - CPU reduction for every I/O to/from the BP

Db2 – Group Buffer Pools

- Rule of thumb for GBP size is sum(local BPs) / 3
  - Goal is to avoid directory entry reclaims
- BPs with very little update activity may not need as much
  - Might also consider GBPCACHE(NO) for such
    - GBP will only be used for cross-invalidation; writes will suffer though
- Other idea: use GBP storage instead of LBP storage
  - Instead of really large LBPs, use really large GBP
  - Saves on the amount of memory you need overall
  - Set with GBPCACHE ALL on the object level
    - Pages will be copied to GBP as they're read regardless of inter-system read/write interest
  - Benefit similar to zHyperLink without actually having to implement zHyperLink
    - Probably actually a little better since DB2 will check the GBP anyways

VSAM Buffering

- There are 4 types of buffer pool management for VSAM:
  - NSR - Nonshared Resource
  - LSR – Local Shared Resource
  - GSR – Global Shared Resource (no longer used)
  - RLS – Record-Level Sharing
- Set by the open, not part of the VSAM dataset definition
- See Chapters 4-6 of VSAM Demystified Redbook
  - https://www.redbooks.ibm.com/abstracts/sg246105.html

VSAM – NSR

- By default batch access with be Nonshared Resources (NSR)
  - Basically tune the buffers via BUFNI, BUFND, et. al. on JCL DD's AMP parameter
- Direct access
  - Data buffers: 1 per "string" (concurrent request)
  - Index Buffers: enough to hold the entire index set (# index records – # data CAs)
- Sequential access
  - Data buffers: up to number of CIs in a CA (probably just use this)
  - Index Buffers: minimal/default fine
- Skip sequential generally works closer to Direct Access as VSAM won't do read-ahead
- Generally: over-specifying buffers has very minimal memory impact in today's systems, but won't provide additional benefits

VSAM NSR and SMB

- System Managed Buffering – let the system figure it out (sort of)
- Dataset must be SMS-managed and extended-format
- Specify ACCBIAS on the AMP parameter or via RECORD_ACCESS_BIAS on the data class
  - SYSTEM takes into consideration the storage class bias
  - Or directly set yourself via DW, DO, SO, SW
- Can provide better performance than plain NSR but seems similarly arcane
  - Maybe more so if you let it rely on the SC/DC of the dataset?

VSAM – LSR

- Primarily used by CICS and IMS but (old) Batch LSR Subsystem extends it to batch work too
  - Can be beneficial to batch in specific circumstances
  - Can work with datasets that are non SMS-managed
- Best for random access that re-references data multiple times
  - Does not do read-ahead for sequential access patterns
- Can defer writes
- The controlling address space creates one or more buffer pools that are shared amongst multiple VSAM files.

VSAM - RLS

- Allows concurrent access at the record level to VSAM from across the sysplex while maintaining file integrity
- Managed by the SMSVSAM address space
  - Buffers managed by SMSVSAM shared by all address spaces (on same LPAR)
- Can eliminate CICS FOR regions and that inherent function shipping
- Files must be SMS-managed
- Requires a coupling facility
  - Lock and cache in CF
- For recoverable datasets batch can only access in read-only mode
- In a lot of ways, VSAM RLS processing looks like DB2
- In many cases can result in performance improvement and I/O reduction compared to LSR, but is situation-dependent

VSAM KSDS from CICS

- Also consider CICS-maintained data table
  - Basically the file is cached in a data space
  - No application changes needed
  - See: https://www.ibm.com/docs/en/cics-ts/5.6?topic=sets-overview-shared-data-tables
- Note file can not be using VSAM RLS
- User-maintained data tables may also be fine for read-only files

Application note

- If you can make application changes, even better than buffering the data is avoiding the database
- Third party products (e.g. tableBASE) can help with this
- There may be lookups that your application could cache itself or maybe don't need to do at all
  - Removing execution of unnecessary code always a performance win!

Catalog Caching

- ISC – In-Storage Cache (default)
- CDSC – Catalog Data Space Cache (VLF-managed dataspace)
  - LRU cache
  - Defined in COFVLFxx, default storage used is only 16MB (max 2GB)
- ECS – Enhanced Catalog Sharing
  - Better for shared catalogs, avoids expensive VVDS volume reservations
  - Works best with CDSC (ISC is completely flushed for each update!)
  - Requires CF
- RLS – Record Level Sharing
  - Reduced contention (record-level)
  - Generally expect better performance than ECS with CDSC
  - Potentially can use larger cache sizes
  - Requires CF

HSM CDSes

- Use VSAM RLS: CDSSNR=RLS in the HSM proclib
- Can improves overall HSM performance
- Size your RLS buffers appropriately, but generously if you have the memory for it

But Scott: I don't have a CF!

- But you could!
- Best practice is of course to run the CF LPAR on a dedicated ICF engine, but can be run on GP
  - We have customers doing this in production (in some cases on surprisingly small systems!)
  - Wouldn't use this for high-volume data-sharing, but could be fine for getting RLS up and running
    - How fine depends on … details
- Very little CPU consumption expected on the CF LPAR
  - Shouldn't directly/noticeably impact your software costs
    (Any additional LPAR may cause increased cache contention)
- If you're a "sysplex in a box" having a single CF may be quite acceptable

Sequential Files – QSAM/BSAM

- I haven't actually seen a significant concern about sequential files since the 90s, but…
- Consider compression to decrease I/Os
  - Newer compression options since the 90s are potentially even better
- Consider raising BUFNO from QSAM default of 5 to at least 10
- Use system-determined block sizes (usually c. half-track)
- Striping was very useful with ESCON and non-RAID disk; maybe less so today

Sorting

- Sort products can make good use of memory for in-memory sorts
- This has been true for decades
  - Make sure you have your parms right: use available memory, don't cause problems
- New SORTL facility on z15 allows even more improvement
- DFSORT exploitation called "IBM Z Sort" https://www.ibm.com/support/pages/ibm-z-sort-and-dfsort-considerations
  - Requires memory >= 70% of dataset size, 200% is recommended planning number
  - IBM test of 44GB sort resulted ~50% reduction in ET and ~40% reduction in CPU vs. in-memory sort without ZSORT
  - But, somewhat oddly, using SORTWK was actually about the same CPU as ZSORT, although it took almost 9x longer (341 seconds vs 39 seconds)
    - So performance savings, but not really CPU savings compared to doing I/O
- I haven't seen a SyncSort benchmark

Summary

- Many (but not all) systems are memory-rich today
  - And if you're not, maybe you should be?
- Take advantage of that memory to avoid I/O to
  - Improve performance
  - (Potentially) reduce CPU and thus (potentially) reduce costs
- SMF has a plethora of data to help you find your I/O
  - SMF 42 records has a good level of detail
- Once you've found your I/O, avoid it by keeping the data in memory

Questions??