

Technical Thread

Buffer Pools & DASD Subsystem Analysis & Tuning

Joel Goldstein
Responsive Systems
www.responsivesystems.com

May 14, 2000

© Copyright 2000



Abstract

This presentation addresses the architecture and functionality of buffer pools, and the DASD from different vendors. It illustrates proven approaches for pool tuning as well as explaining the thresholds, performance exceptions, and the meaning of vital values and percentages.

There have been many changes in DASD implementations over the last several years. While it's important to understand the underlying changes, it is just as important to understand the different implementations – to get beyond the salespersons hype, and to understand the approaches that will provide good performance, and those that will fail.

Bullet Points

- Buffer Pool architecture & function
- Pool thresholds, indicators, & percentages
- Proven pool tuning techniques
- DASD architectural implementations
- DASD performance perspectives and analysis



Outline

1. Buffer Pools
 - a. Structure, function, relationships
 - b. Thresholds and indicators
 - c. Calculating meaningful percentages
2. Pool tuning approaches
 - a. Resource availability
 - b. Getting the data – Statistics, Accounting, Traces
 - c. Tuning approaches and paybacks from the real world
3. DASD
 - a. How does DASD really work
 - b. Comparing the different implementations
 - c. The basics....
 - d. Where's the data? DB2, SMF records to use...
 - e. Understanding, interpreting, and using the data
4. Summary, guidelines, recommendations

Buffer Pools

- **Purpose**
 - Avoid I/Os
 - Reduce elapsed times
 - Reduce processing costs
- **We've had buffer pools in other systems and DBMSs since the early 70's**
 - Underlying implementations usually vary



DB2 has always been designed to exploit memory – keeping data in buffers, to avoid the cost and delay of I/Os.

Buffer Pools – Application Performance Perspective

- Class 2 elapsed time .20 seconds
- I/O wait .15 seconds
- The performance is poor!
 - 75% of the elapsed time is I/O wait



Performance relationships are a critical piece of any analysis and tuning project. As the above example illustrates, when the I/O wait time is a substantial percentage of the elapsed, there is a performance problem waiting to be fixed. Additional analysis by looking at the Accounting detail information will indicate how much of the is related to pool tuning issues, and how much may be caused by a poorly performing DASD environment.

Buffer Pools – Application Performance Perspective

- Class 2 elapsed time .060 seconds
- I/O wait .030 seconds
- Class 2 time is good, but the performance is poor!
 - 50% of the elapsed time is I/O wait



Even when your response time looks good, and is meeting service objectives, there may still be evident performance problems.

Improving performance by eliminating I/Os saves CPU cycles, and allows you to make better use of your processing resources.

Even though reducing the I/O rate saves CPU cycles, don't expect your processor busy rate to decrease. As you tune and eliminate I/Os, you are getting rid of a throughput constraint – the I/O wait time. This allows more real work (transactions) to process within the same elapsed time frame. So, your processor busy rate may even increase – but you are getting more value for your processing dollar,

There is also a critical issue of user/client productivity and satisfaction. Higher productivity reduces business costs and improves corporate profitability.

Buffer Pools – Current Resources

- **80 Pools** *Internally*
 - 50 – 4K 0-49
 - 10 – 8K 100-109
 - 10 – 16K 120-129
 - 10 – 32K 80-89

- **Will we see more in the future?**



Prior to Version 3 there were only four buffer pools. Now we have eighty. For those installations stuck in one pool, or only two/three pools, there is a strong message here. IBM would not have taken the time, resources, and money to create the opportunity to use more pools if it was not necessary. Just throwing massive amounts of memory at just a few pools is wasteful and won't provide the best performance for your memory dollar. You can argue that memory is inexpensive today – until it is time to write that big check for the next multi-gigabyte increment.

I expect we may see even more pools available in the future, and perhaps more different pool sizes – after we get to 64 bit addressing.

However, more is not always better. Performance must be monitored and tracked for every pool, so the more you have, the more work is required to measure/track performance.

Based on the analyses at dozens of installations, most sites can optimize in six to eight pools. Some may need a few more, up to a dozen. Personally, I have not seen any installation yet that really needs more than a dozen pools to obtain optimal performance for the available memory.

Buffer Pools – Where do they live?

- **Virtual Pools**
 - Real memory – pageable to expanded
 - Above 16M line
- **Hiper Pools (Hiper space)**
 - Expanded memory
- **VP – Data space**
 - Real/Expanded memory – can page to Dasd
 - 128 bytes/buffer in DBM1 address space
- *What does the future hold ?*

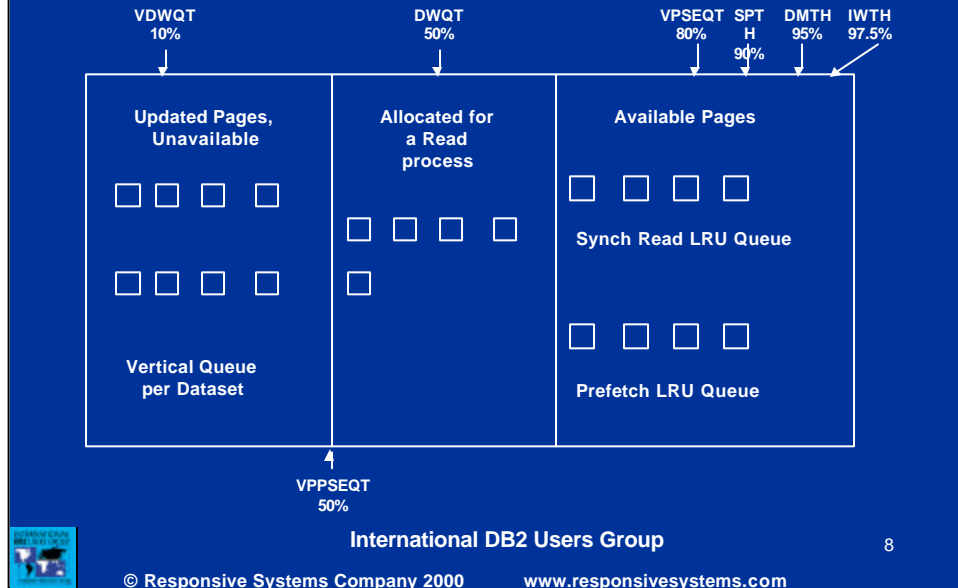


Virtual pools (VP) should be backed by real memory so there is low to no page movement by OS/390. Paging to expanded is costly, and will impact the entire processor if the rate/second gets too high. The rate can be calculated and tracked from the BP statistics – but keep in mind that this is only for the pools. The rate for the entire DBM1 address space will usually be much higher. VPs are also limited to a maximum allocation of 1.6 Gigabytes.... Then you still need room for the other large memory consumers such as the EDM Pool, RID Pool, memory sort/work space (per user), and some code. This all must fit within a 2 Gigabyte maximum for an address space.

Hiper pools (HP) live in expanded memory. Page movement between the VP and HP can be through ADMF, and can move blocks of pages at a time. When ADMF is utilized the SRB runs under an IO engine, so it is 8% less overhead than base paging that moves one page at a time using a MVPG instruction. Dirty pages cannot be moved to a HP.

A VP Dataspace can be in real storage, expanded storage, or a combination. There is a 128 byte block of memory associated with each VP/dataspace buffer, so data can be moved into real storage for processing.

Buffer Pools – General Structure



We should all be familiar with the standard thresholds. There is an occasional point of confusion when looking at some monitor or other reports – the term ‘current buffers active’. This is really the number of buffers that are NOT available, as all the buffers from the two left areas of the pool. All buffers in a pool contain data 100% of the time.

While WITH is a critical threshold, you may see some counts there – if you haven’t hit SPTH or DMTH, then ignore it, as there are two other circumstances that may cause immediate synchronous writes of pages. These are an updated page still being in the pool at a second checkpoint, and a page taken off a write queue for use by another application – and then written synchronously when the second application commits.

Excluding pools dedicated to sort/work files, the VDWQT should usually be set quite low, and should be at ZERO when the average pages/write is a low single digit like 1.7, 2.6, 3.4, etc. Setting this to zero uses an internal threshold of 40 pages as the trigger point for the asynchronous write process. Version 6 allows the actual number of pages for the threshold to be set as VDWQT (0,xxx). This will be especially useful for large pools where even a value of 1% may be hundreds of buffers. Additionally, when there are hundreds of objects in the pool, the DWQT should also be lowered.

The primary reason for lowering these thresholds is to avoid flooding the sub-system with write I/Os when DB2 takes a checkpoint.

Buffer Pools – *What really matters...*

- **SPTH, DMTH, IWTH**
 - IWTH not a concern if you haven't hit SPTH, DMTH
- **Unavailable read engine**
 - Usually an indication of DASD performance problems
- **Pageins for read/write**
 - Calculate the rate/sec as a sum across all pools
 - This is *Only the pools*, not the entire DBM1



Running out of Read/Write engines is usually an indication of a poorly performing DASD sub-system. While running out of write engines has a similar indication, it may also be reflective of having the VDWQT and DWQT thresholds set too high and DB2 checkpoints are flooding the system with write I/O requests.

The paging rate per second should be calculated and tracked as a sum across all the pools. If this is consistently greater than 10, the pools are probably over allocated for the amount of real memory on the system. This should be evaluated further, regarding the paging for the entire DBM1 address space. This information can be obtained from standard RMF reports, or your operating system programmer should be able to assist you – if they haven't already come looking for your scalp.

Buffer Pools – *What really matters...*

- **I/O rate/sec**
- **System hit ratio**
 - $(\text{Getpages} - (\text{sum of all pages read}) / \text{Getpages}) * 100$
 - Can be a negative ratio
 - Dynamic prefetch
 - Thrashing
 - Batch jobs when the CPU busy rate is high



The I/O rate/sec is the overall most important performance metric (assuming that you're not hitting any thresholds, etc). This needs to be tracked for the entire system, each pool, and all the heavily used objects in every pool.

After the I/O rate, the system hit ratio is very important. However, as you will see later in this presentation, there can often be substantial reductions in the I/O without shows much improvement for the system hit ratio.

The above formula is in the V3, V4, and V6 manuals. Somehow it got lost with V5. Some online monitors incorrectly report a GP/RIO ratio as a hit ratio. Since this completely ignore the number of pages read by prefetch functions, it often leads people to believe they have good pool performance when it may be poor.

An important caveat for this ratio: it can be a negative number. While pool thrashing can certainly cause this, it's most often caused either by dynamic prefetch reading in many pages that are never referenced by an application, or by sequential scan of a tablespace with multiple tables – such as some of the large ERP applications.

Buffer Pools - Important

- **HP Retrieval Ratio**
 - Pages Read/Pages Written
- **HP Read Failure, Write Failure**
- **Pages/Write**
 - Pool level – analyze objects in the pool
 - May vary widely
 - Most may be very low, and some may be much higher
 - VDWQT = 0
 - VDWQT = (0, xxx) **V6 option**
 - Lower DWQT when 100's of objects in the pool



A simple measure of HP efficiency is the above retrieval ratio. However, this is another situation where a low percentage may still be providing good payback.

Read and Write failures are an indication that the operating system has stolen pages away from the HP. This should be easily determined from your online monitor, or DB2 statistics data. If adequate memory is available for HPs, and your goal is to maintain online transaction response times, then you should consider setting CASTOUT=N.

If you're not in a datasharing environment, and the number of pages/write at the pool level is a consistently low number, then VDWQT should be set to zero.

Buffer Pools – Pages/Write

Online Day - SYSB

Pool	I/O	Get Pages	Updates	Hit Ratio	I/O Sec	Pages/Write
BP0	30258	432028	10183	61.6	4.21	1.86
BP1	4041	439899	397446	98.1	.56	25.83
BP2	18996	582361	55341	76.6	2.64	3.36
BP3	16524	1515545	85233	96.9	2.3	2.15
BP4	89357	2020372	1544	80.7	12.42	1.16
BP5	2006	746983	992	99.5	.28	1.23
BP6	95820	921084	10177	65.5	13.32	5.32
BP7	4126	151683	1138	57.9	.57	1.9

Batch Workload - SYSB

Pool	I/O	Get Pages	Updates	Hit Ratio	I/O Sec	Pages/Write
BP0	558	6296	1307	72.2	.08	2.98
BP1	105073	1074730	0	24.4	14.59	0
BP2	30383	730124	0	61.6	4.22	0
BP3	104648	515369	363455	80.5	14.53	24.63
BP5	515769	2362737	95083	29	71.63	28.44
BP6	244828	2142172	98021	26.4	34	10.86



International DB2 Users Group

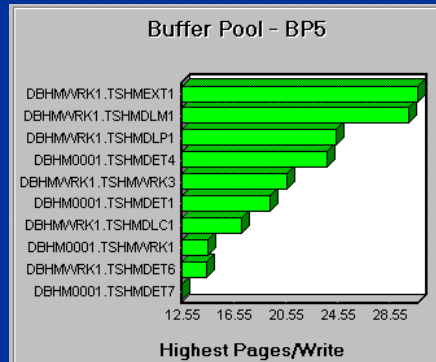
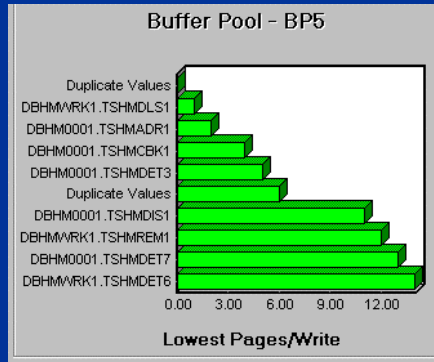
12

© Responsive Systems Company 2000 www.responsivesystems.com

The pool information above highlights the difference in workloads and performance characteristics for online versus batch at one installation. Widely varying workloads can benefit from changing pool sizes and thresholds.

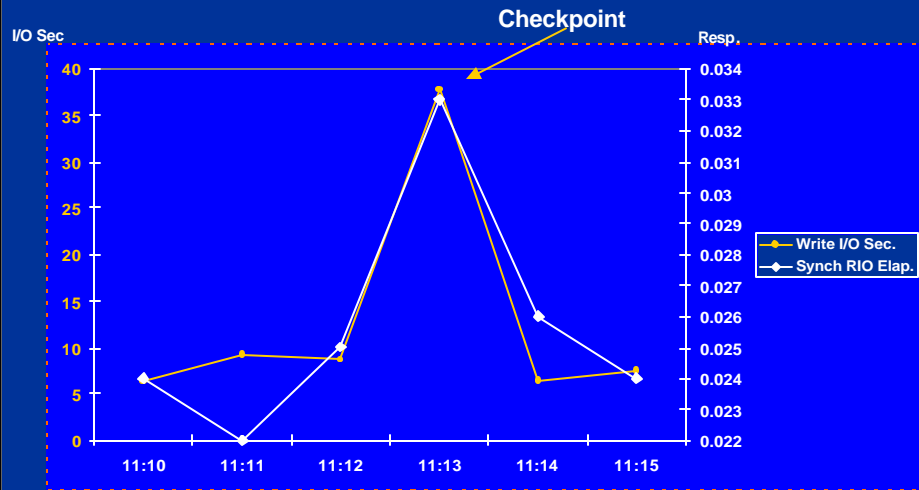
This can be accomplished in many ways such as automated operator software, or even by front and back ending the batch workload with jobs that alter pool sizes and thresholds.

Buffer Pools – Pages/Write



Looking at the ‘top ten’ objects in various categories can provide some unique information about how objects are used. Notice the *duplicate values top bar on the left*. This indicates that two or more objects had the identical very low value.

Buffer Pools – VDWQT=10



International DB2 Users Group

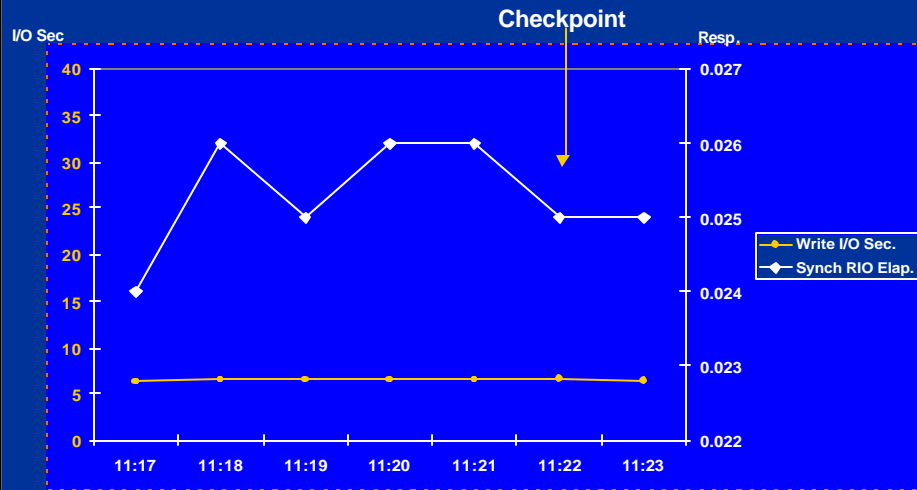
14

© Responsive Systems Company 2000

www.responsivesystems.com

Effect of heavy write workload caused by a DB2 checkpoint.

Buffer Pools – VDWQT=0



International DB2 Users Group

15

© Responsive Systems Company 2000

www.responsivesystems.com

After the VDWQT was set to zero to trigger asynchronous writes at 40 pages per object, there is not huge burst or write activity when a checkpoint takes place.

Please note the slight change in scale on the Y2 axis compared to the previous graph.

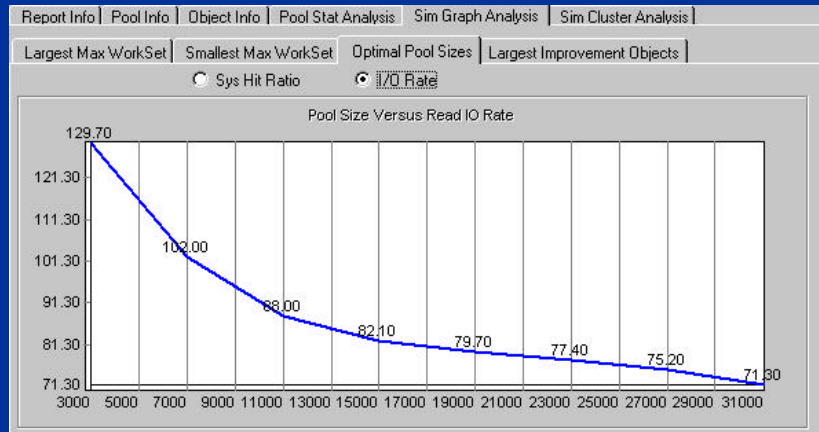
Buffer Pools – Writing out the updates

- **Pages/Write**
- **DB2 can write 32 Pages I/O...**
- **Dequeue up to 128 Pages per Dataset, scheduled for one write engine**
- **Limit of 150 CIs per write**
 - Sorted into Ascending RBA order first
 - VDWQT of 0 schedules a write at 40 pages



Let's look at write activity. DB2 can write 32 pages per I/O, and 64 for utilities. However, there is an internal limiting threshold of 150 CIs for any one write. The intent of this threshold is to (hopefully) have all the pages written on one cylinder so device arm movement is not necessary. So, when you see a consistently low number of pages/write, it means that the pages updated are far apart. Lowering the VDWQT to zero does not show any appreciable increase in the total write activity. Like wise, raising it rarely shows any improvement – and may really hurt at checkpoint time. Even considering the fact that most writes today are to a non-volatile control unit cache, there may still be massive contention at the logical volume level when datasets are poorly placed.

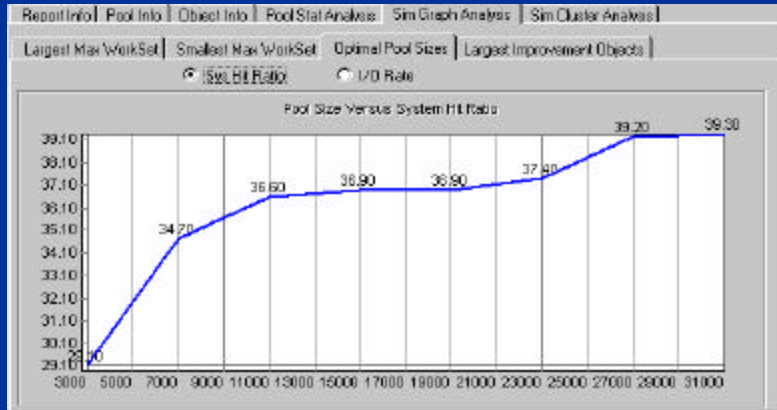
Buffer Pools – I/O Rate/Second



The overall I/O rate per second, for the system and each pool is the primary performance criteria. Assuming adequate memory is available to increase pool size without increasing the OS/390 system paging rate, this pool continues to show good performance improvement as the size is increased right through 31,000 buffers. Note that that the increase is greater between 27,000 and 31,000 than between 23,000 and 27,000.

This is not unusual. There are various thresholds that come into play based on the working set sizes of the objects so a performance curve may flatten, then show another good improvement at some markedly larger size. However, this may not always be the case, and each pool and its objects behave differently. With any pool and combination of objects, there is a point of no further improvement, and additional memory is just wasted.

Buffer Pools – System Hit Ratio



The System Hit Ratio is the second most important performance criteria. While critical to track, note that it is relatively flat between 11,000 and 19,000, show minimal gain up to 23,000 and then is totally flat after 27,000.

This illustrates that the I/O performance curve is both more important, and ultimately more indicative of real performance and performance gains.

Buffer Pools – Buffer Management

➤ LRU

- Default
- Should be used for most situations

➤ FIFO

- Slight decrease of overhead
 - No queue or chain management
- When an object can fit/live in a pool
- Large random – very low/no re-reference



Buffer Pool – Tuning Methodology

- Mostly Random – **RAMOS**
 - Small/Medium Working sets
 - Large Working sets
 - Very Large & Very Random
 - Never have a good Hit %
 - May have high I/O rate
 - Indexes

- Mostly Sequential – **SAMOS**
 - Small/Medium Working sets
 - May be able to get a decent Hit %
 - Large Working sets
 - Never get a decent Hit %
 - Does not need a large pool



A proven pool tuning methodology is the proper grouping of objects based on access type and working set size (number of resident pages in the pool). Object catalog statistics indicating the number of physical pages are not useful for this approach.

An object may have a million physical pages, but the important thing is how many you re-reference within a few minutes – and the impact this object reference pattern has on other objects in the pool and vice versa.

DASD

- Still rotating brown platters
- Dramatically smaller
 - 2.5"
- Dramatically faster
 - 10,000 RPM
- Some cache at the device level



There have been dramatic advances in DASD technology over the last few years, just as processor technology. The PCs on your desktop today may have more processing power and disk storage than some of the smaller mainframes a decade ago.

However, most of the basics of DASD performance remain the same. Quite startlingly, much of the actual performance is worse today than it was a decade ago.

Quite simply, because people (incorrectly) think it's not important. That the new whiz-bang device from the vendor du jour will provide the silver bullet. WRONG! Remember, they are there to SELL you something. A Mercedes salesperson will happily overcharge you far more than a Kia salesperson.... Because they can.

The basics have not changed!

SMS can do nice job for you if it's set up properly. Just 'give it all' to SMS and you die ! Sure we don't have time to manage thousands of datasets – and you don't have to. There is some reasonably small number of performance critical objects. Manage the placement of these, let SMS have the rest and you can run quite well.

DASD – Today's World

- **RAID Implementations**
- **Gigabytes of Cache**
 - Depends on the vendor
- **Intelligent Cache Management**
- **Read Ahead – prestage based on access & cache hit**
 - Track
 - Cylinder



DASD - Raid

- **Raid 0**
 - Stripes data across all disks, no redundancy or parity
- **Raid 1**
 - Mirrors data across multiple disks
 - Internal access to either disk (data) reduces contention
- **Raid 3 & Raid 4**
 - Stripes data across multiple drives, writes parity to dedicated drive
- **Raid 5**
 - Stripes data and parity at block level across all drives in an array



Most (not all) of the top performance devices use a Raid5 implementation. We've all heard about the RAID write penalty, and it's quite real. However, from the application and Db2 perspective, we rarely see it because the write is usually to fast non-volatile memory on the control unit.

The multi-gigabyte caches on these units is the only thing that may provide decent performance. The reality is that a cache miss usually has worse performance than an older 3390... if/when the cache management logic does not succeed in pre-staging the data into the cache.

DASD - Raid

- Raid 0
- Raid 1
- Raid 3 & Raid 4
- Raid 5

Reliability



Performance



International DB2 Users Group

24

© Responsive Systems Company 2000 www.responsivesystems.com

Just think about the opportunities of having data striped across multiple disks... the opportunity for multiple RPS misses.

DASD – EMC Symmetrix

- 16 Paths between controller & DASD
- 4000 I/O Sec, expect 3000
- 64 Gig of Cache (Max), Intelligent Algorithms
- RAID 0,1, RAID-S (RAID 4 or RAID 5)... in same Box
- Does not truly 'stripe', so a Logical Volume (or multiples) may be on one Physical



Understanding the logical device to internal device mapping is critical for success. Somebody in your installation knows what this is, and it is to your real benefit to obtain this knowledge.

DASD – HDS 7700

- 8 Paths between controller & DASD
- 8 Gig Cache at Controller Level
- 2400 I/O Sec max, expect 14-1600
- Random Reads have Priority over Sequential
- RAID write contention issues
- Splitting of Cache for Read/Write
 - Critical 'Must Write' scenario...
- No control over where data is written



Note the large performance improvements from this device to the 7700E on the next slide.

DASD – HDS 7700E

- 32 Paths between controller & DASD
- 16 Gig Cache (Max)
- Faster SCSI Disks
- 8000 I/O Sec max, expect 6000



Four times the number of paths.

Twice the cache.

Much faster disks

All this doubles the performance.

DASD – IBM RAMAC

- RAMAC 1
- RAMAC 2
- RAMAC 3
- RAMAC RVA
- RAMAC SVA



We've had many incarnations of Ramac. As an interesting aside, the first IBM disk in the 60's was also called Ramac. Today we can get more data storage on one disk drive than most entire data centers had in the 70's.

DASD – IBM RAMAC RVA

- 4 Paths between controller & DASD
- 1200 I/O Sec max, expect 8-900
- Has heavy write/contention problems...
- RAID 5
- Data Compression up to 6:1 (but wastes space)
- Does not 're-write' Changed data



Not many paths, little cache, and low performance.... And it was 'state of the art' about a decade ago.

DASD – IBM RAMAC RVA2T

- 8 Paths between controller & DASD
- 2400 I/O Sec max, expect 14-1600
- Has heavy write/contention problems...
- RAID 5
- Data Compression up to 6:1 (but wastes space)
- Does not 're-write' Changed data



Doubled the paths, and the throughput doubled. Maybe we're onto something here.

DASD – IBM RAMAC SVA

- 16 Paths between controller & DASD
- 4000 I/O Sec max, expect 3000
- Has heavy write/contention problems...
- RAID 5
- Data Compression up to 6:1 (but wastes space)
- Does not 're-write' Changed data



Doubled the paths again, and throughput is almost doubled. But we lost about 16% from a direct doubling from the 2T. We still need more paths, but we also need something else for that next big leap....

DASD – IBM SHARK, 2105, ESS

- 32 Escon channels
- 16 Gig of cache
- SCSI 10,000 RPM disks, 18 Gig each
 - 36 Gig option, but 7200 RPM
- Parallel Access Volume (PAV) eliminates IOSQ problem
- Multiple Allegiance eliminates PEND problems
- IO Rates with 70% cache hit
 - Depending on the channel & disk configuration, may support up to 8500 I/Os/Sec before significant degradation
- User benchmarks show dramatic improvement over previous RAMAC-3 devices, and significant improvement over RVA devices.



Four PCI buses have a theoretically internal bandwidth of 800 Meg/Sec. 384Mb of nonvolatile memory is available to improve heavy write performance.

Can support up to 11.2 Terabytes of storage

While PAV and MA remove many of the concerns regarding dataset placement, some users have still encountered varying degrees of performance degradation with very large data volumes. The IBM documentation states that Indexes should still be separated from data, but it is not necessary to worry about the placement of partitions relative to heavy parallel I/O processes.

While this may ultimately prove true, it seems to fly in the face of logic – from my perspective. If I don't have to worry about the location of partitions, then why Indexes? We need some more real user experiences in heavy production environments. Hopefully these will become available over the next few months.

Keep in mind that IBM has run all their standard benchmarks And your systems and performance are very different.

The IBM plans for the future are to add more cache, and increase the performance of the internal engines.

DASD – What happens when you do it wrong...

View of All DASD Lines 1 to 7 of 212

DevNo	Volume	Response	Cache hit %	%Busy
092C	ZTDH34	49.8 ms	Cache hit % low	4.4
0979	ZPDHLB	36.8 ms	Cache hit % low	8.4
023B	MVSHPI	36.7 ms		.6
096B	ZTCH21	32.0 ms	Cache hit % low	.2
027B	MVSHPI3	31.0 ms		.7
023C	MVSHPI2	30.8 ms		.6
0970	ZTDH75	29.6 ms	Cache hit % low	3.4




Using an OS/390 system monitor to look at DASD performance for the entire LPAR.

Looks like we have a few problems.

DASD – What happens when you do it wrong...

xdsk	Number of Devices =	961	Number of Samples =	145							
	Volser	LCU	Rate Util%	Resp =	IOSQ +	Pend +	Conn +	Disc	RESV%	#Alloc	
+ 025E	ZTDH15	000	.0	.0	33.6	.0	.3	12.6	20.6	.0	19.0
+ 027B	ZPSH10	000	.0	.0	30.6	.0	.5	1.1	29.0	.0	13.4
+ 0920	ZTLH06	00E	.0	.2	65.9	.0	.6	4.0	61.2	.0	1.0
+ 096B	ZTCH21	00F	.0	.1	46.2	.0	1.1	45.1	.0	.0	.0
+ 096B	ZPDHL9	00F	.9	4.3	46.9	.0	1.2	7.0	38.6	.0	3.0
+ 096D	ZTDH86	00F	.2	.9	38.5	.0	1.1	9.8	27.5	.0	4.0
+ 0970	ZTDH75	00F	.0	.0	51.4	.0	1.0	8.4	42.0	.0	13.0
+ 0979	ZPDHL8	00F	4.2	<u>19.9</u>	67.6	19.7	1.1	4.3	42.4	.0	3.0
+ 097A	ZTDH81	00F	.0	.1	35.7	.0	3.0	1.6	31.1	.0	12.0
+ 097B	ZTDH82	00F	.0	.1	85.0	.0	.7	21.7	62.5	.0	15.0

	↑	↑	↑	↑	↑
	A	B	C	D	E



International DB2 Users Group 34

© Responsive Systems Company 2000 www.responsivesystems.com

A - shows the overall poor performance across all logical devices.

B – IOSQ shows a severe queuing problem for device ZPDHL8. IOQ is caused by dataset placement problems. However, note that there are only 3 open datasets in use on this volume

C – pend is channel path contention, and it may be caused by interference from other LPARs accessing data on this volume.. As in shared DASD

D – connect is data transfer. Long transfer times are usual for prefetch operations

E – disconnect indicates a cache miss, and the data had to be read from disk

DASD – Destroying a *State of the Art* Device...

DATA BASE	TABLE SPACE	P	I/O COUNT	I/O %	MAX IOWAIT	AVG IOWAIT					
							--ms--	--ms--			
							0	20	40	60	80
XCBES01P	ESIS47I1		78	0.0	84	14		***			
XCBES01P	ESIS47TS		15	0.0	55	18		****			
XCBES01P	ESIS48I1		43	0.0	106	20		*****			
XCBES01P	ESIS49I1		39	0.0	59	12		***			
XCBES01P	ESIS50I1		10426	6.2	17037	10		**			
XCBES01P	ESIS50TS		24354	14.5	146424	209		*****			
XCBES01P	ESIS51I1		3	0.0	3	5					
XCBES01P	ESIS52I1		8534	5.1	3693	14		***			
XCBES01P	ESIS52TS		2443	1.5	4720	19		****			
XCBES01P	ESIS53I1		3	0.0	11	8		**			
XCBES01P	ESIS54I1		127	0.1	133	15		***			

146.4 Secs = almost 2 ½ minutes



Continuing to look at performance, we look at one volume to see what datasets are there. One dataset has the highest I/O count and the worst performance. Consider that that the maximum response time is almost 2 ½ minutes, with an average of ¼ second.

So – this was a ‘state of the art’ device from a well known vendor. There is really nothing wrong with the device or the technology. We can destroy the performance of anything by using it improperly, by not using common sense.

This is a corporate *management* problem... caused in part by the ‘bean counters’.

There were two management edicts at work here that destroyed performance, and impacted the productivity of users worldwide:

- a. All logical devices must be at least 80% full – we don’t want to waste any disk space
- b. All datasets are completely managed by SMS.

DASD – SMF 42-6 Analysis

Volume	IO Intensity	IO Chit%	IO Resp	IO Conn	IO Pend	IO Disc	IO Queue	IO Count	Cache Cand	Cache Hits	IO/Sec	Cache Ratio	Write Cand	Write Hits	Write Ratio
DD9132	0.7387	88	9	1	0	2	6	664793	623358	583737	82.7321	93	1019	937	91
DD9166	1.8011	91	58	7	0	15	36	251533	232784	229934	31.5346	98	3602	3570	99
DD9108	1.2290	31	34	19	0	7	8	195187	82755	60314	36.1457	72	9575	9503	99
DD9187	0.5488	84	23	3	0	13	7	193262	185556	163146	23.8595	87	1396	1365	97
DD939K	0.2180	86	10	3	0	3	4	176599	154979	151875	21.8023	97	2562	2455	95
DD939F	0.3919	83	18	3	0	9	6	176345	155476	146933	21.7710	94	3784	3629	95
DD9202	0.3827	85	18	3	0	7	8	172218	152399	146484	21.2615	96	2970	2872	96
DD9118	0.4845	87	24	3	0	17	4	163520	160736	141486	20.1877	88	46	46	100
DD9144	0.4796	71	26	2	0	14	10	149404	137046	106686	18.4449	77	1554	1534	98
DD9146	0.3555	94	21	4	0	12	5	137134	133231	129126	16.9301	96	165	162	98
DD9127	0.3637	64	24	5	0	13	6	122733	109179	78563	15.1522	71	2400	2377	99
DD9149	0.4565	87	27	3	0	17	7	121729	110663	105309	16.9068	95	1021	991	97



The SMF 42-6 records provide the only means of really tracking performance at the DB2 dataset level. Interval reporting must be specified in the SMFPRMxx module for the records to be useful. If this is not specified, then records are only produced when a dataset is de-allocated.... So you will not see activity for your most active datasets.

The analysis of many thousands of records can be either very difficult, or quite simple depending on the tools you have available.

Above is a list of active volumes sorted in order of decreasing I/O activity. I/O Intensity (resp * I/O rate/sec) is often used to highlight the volumes or datasets that should be most interesting, and get the most attention.

However, that approach is not always the best. When you have the ability to immediately re-order the data on different criteria, other information may be more relevant to your tuning exercise. There are several volumes that have a higher I/O Intensity than DD9132 – and that is caused by significantly higher response times, even though the I/Os are much lower.

So, the most beneficial approach is to look at the number of I/Os, and the calculated IO Cache Hit percentage (CHIT%). Now, if you can get a very high cache hit rate in the control unit, then you ‘may’ be able to tune at the buffer pool level, and eliminate much of the I/O.

The information we don’t have, and can’t get, may reduce some of the pool tuning opportunity. We don’t know anything about pre-stage activity of data into the controller cache, and this is certainly increasing (to some degree) the number of cache hits.

DASD – SMF 42-6 Analysis

Datasets on volume DD9132:

Database	Object	File	Partitio	IO Intensity	IO Hit%	IO Resp	IO Conn	IO Pend	IO Disc	IO Queue	IO Count	Cache Cand	Cache Hits	IO/Sec	Cache Ratio
DSKUPROD	SSK	I0001	A001	0.7277	88	9	2	0	2	5	654962	614021	578294	80.8595	94
DAPACFX0	IPAYREQ1	I0001	A001	0.0087	27	30	1	0	22	7	2338	2337	622	0.2886	26
DSUACFX0	ISHUUPCD	I0001	A001	0.0125	66	14	2	0	7	5	7210	6729	4735	0.8901	70
DAPACFX0	IPREQIT0	I0001	A012	0.0007	22	30	1	0	25	4	190	186	41	0.0235	22
DMRACFX0	SMRISFE	I0001	A019	0.0000	50	14	1	0	10	3	2	2	1	0.0022	50
DAPACFX0	IVNDINVO	I0001	A010	0.0001	0	38	1	0	28	9	12	12	0	0.0033	0
DFMACFX1	SPMRSMS	I0001	A010	0.0013	64	29	5	0	9	15	39	31	25	0.0433	80
DSKUPROD	SSKSTMTH	I0001	A021	0.0002	48	30	1	0	22	7	40	40	19	0.0074	47



So, we have eight datasets on the volume, and one of them has 99% of the activity.

Now it is necessary to back to the buffer pool this object is assigned to, determine what type of access it receives, its working set, and determine if we can tune the pool(s) to reduce the I/O.

DASD – SMF 42-6 Analysis

Volume	I/O Intensity	I/O Chit%	I/O Resp	I/O Conn	I/O Pend	I/O Disc	I/O Queue	I/O Count	Cache Cand	Cache Hits	I/O/Sec	Cache Ratio	Write Cand	Write Hits	Write Ratio
DD9166	1.8011	91	58	7	0	15	36	251533	232784	229934	31.5346	98	3602	3570	99
DD9108	1.2290	31	34	19	0	7	8	195187	82755	60314	36.1457	72	9575	9503	99
DD9132	0.7387	88	9	1	0	2	6	664793	623358	583737	82.7321	93	1019	937	91
DD9152	0.7082	62	20	3	0	12	5	63734	48280	39801	35.4078	82	1330	1311	98
DD939G	0.5945	71	17	3	0	9	5	94426	74391	67078	34.9726	90	4176	4101	98
DD9106	0.5859	48	24	4	0	14	6	43943	36898	20994	24.4128	56	2153	2150	99
DD9187	0.5488	84	23	3	0	13	7	193262	185556	163146	23.8595	87	1396	1365	97
DD9138	0.5403	85	19	3	0	13	3	102369	89531	87052	28.4358	97	299	293	97
DD9118	0.4845	87	24	3	0	17	4	163520	160736	141486	20.1877	88	46	46	100
DD939A	0.4798	74	20	3	0	11	6	43186	38436	32048	23.9922	83	1865	1778	95
DD9144	0.4796	71	26	2	0	14	10	149404	137046	106686	18.4449	77	1554	1534	98
DD9183	0.4609	78	55	4	0	16	35	67882	63968	52864	8.3805	82	427	423	99
DD9149	0.4565	87	27	3	0	17	7	121729	110663	105309	16.9068	95	1021	991	97
DD939F	0.3919	83	18	3	0	9	6	176345	155476	146933	21.7710	94	3784	3629	95
DD9202	0.3827	85	18	3	0	7	8	172218	152399	146484	21.2615	96	2970	2872	96
DD9127	0.3637	64	24	5	0	13	6	122733	109179	78563	15.1522	71	2400	2377	99



Another quick view of the same performance data, sorted by I/O Intensity. Look at volume DD9166. More than 1/2 of the response time is IOQ. Perhaps moving a dataset off this volume will improve performance to a more acceptable level.

DASD – SMF 42-6 Analysis

Datasets - All																
Database	Object	File	Partit	Volume	IO Intensity	IO Chk%	IO Resp	IO Conn	IO Pend	IO Disc	IO Queue	IO Count	Cache Card	Cache Hit	IO/Sec	Cache Ratio
DRCADPX0	SRECEV	10001	A001	DD9166	2.2214	92	72	21	0	1	50	24906	23171	22661	30.6526	99
DIFACPX0	STFNSTR	10001	A001	DD9108	0.7921	25	43	19	0	6	18	14905	3930	3757	18.4204	95
DSKUPRO0	SSK	10001	A001	DD9132	0.7277	68	9	2	0	2	5	65492	61402	578294	80.8595	94
DETACPX0	SFOTRST	10001	A001	DD9155	0.3325	78	37	5	0	18	14	72782	6498	5683	8.9654	87
DITACPX0	SUPC	10001	A001	DD9127	0.2537	69	19	2	0	12	6	109174	99549	74437	13.3648	74
DSNADPX0	SASNHDR	10001	A001	DD9138	0.2216	86	19	3	0	13	2	99728	87998	86222	12.3121	97
DWRACPX0	INWKSLS0	10001	A007	DD9243	0.2148	26	27	13	0	2	12	7161	2142	1841	7.9567	85
DPDADPX0	SPURCHS	10001	A001	DD9137	0.2037	68	19	4	0	8	7	88869	74000	57177	10.7293	77

Datasets on volume DD9166

Database	Object	File	Partit	Volume	IO Intensity	IO Chk%	IO Resp	IO Conn	IO Pend	IO Disc	IO Queue	IO Count	Cache Card	Cache Hit	IO/Sec	Cache Ratio
DRCADPX0	SRECEV	10001	A001	DD9166	2.2214	92	72	21	0	1	50	24906	23171	22661	30.6526	99
DAPADPX0	IMITEM0	10001	A009		0.0090	8	63	1	0	27	35	1160	1160	98	0.1432	8
DRCADPX0	IPKITEM0	10001	A009		0.0024	54	44	1	0	14	29	440	436	237	0.0543	55
DPDADPX0	SMESTD5	10001	A013		0.0002	82	10	1	0	4	5	22	22	18	0.0244	81
DAPADPX0	SYNDINV	10001	A011		0.0001	0	55	1	0	23	31	5	5	0	0.0019	0



Using a slightly different view of the data, and looking at DD9166 we can see which objects are on the volume, and the performance of each. Unfortunately there is no obvious silver bullet here. One dataset has 99% of the activity, and the long Conn time indicates that there is heavy sequential prefetch activity.

There ‘may’ be some minor improvements possible, and more analysis is necessary to determine if this workload is relatively consistent over many periods, and across multiple days.

Moving the datasets that are 2nd and 3rd in activity may provide some gain. Since they have very low cache hit ratios, and very long Disc times, they really hurt the performance and tie up the volume when they happen.

The major problem is the IOQ time, which means the device is busy servicing an I/O request, so other I/Os must queue and wait until the executing I/O completes.

The new Shark technology, using PAV should be able to eliminate most, if not all of the IOQ time.

Summary

- Buffer Pool Tuning is your greatest system performance opportunity and payback
- RAMOS/SAMOS object grouping has improved performance at many dozens of companies worldwide
- DASD performance is generally poor and getting worse, not better
 - Lack of understanding
 - Ignoring the basics
- The data is available to find your problems and improve performance



QUESTIONS?

- There are many other performance presentations and white papers on the web site.

