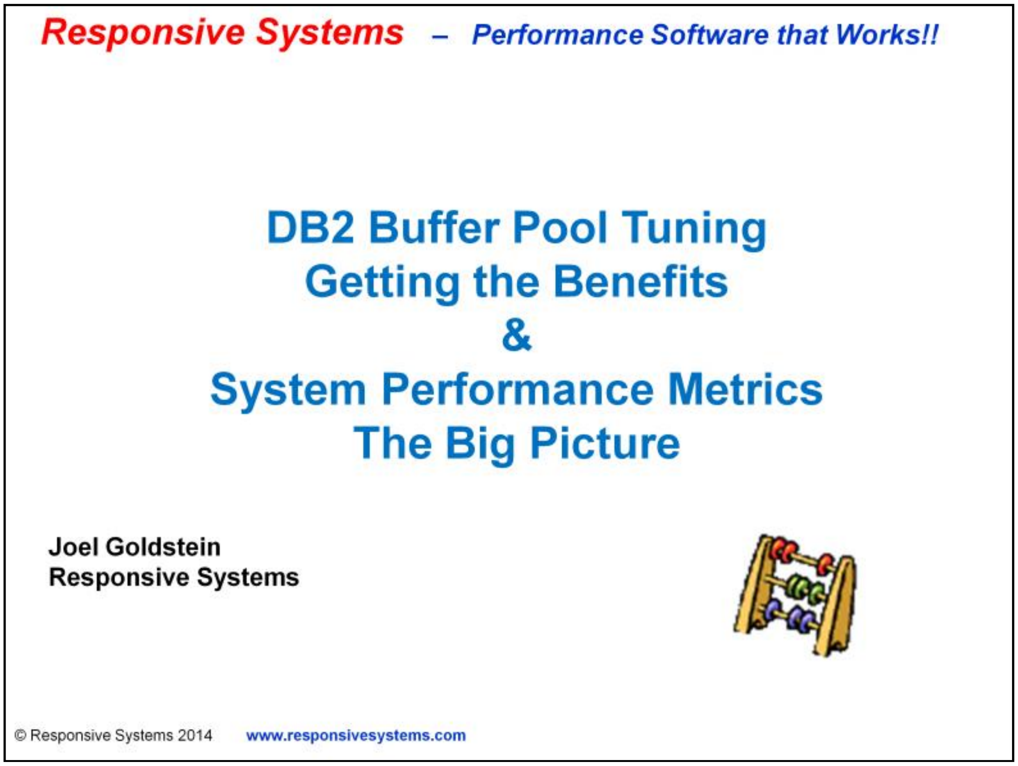
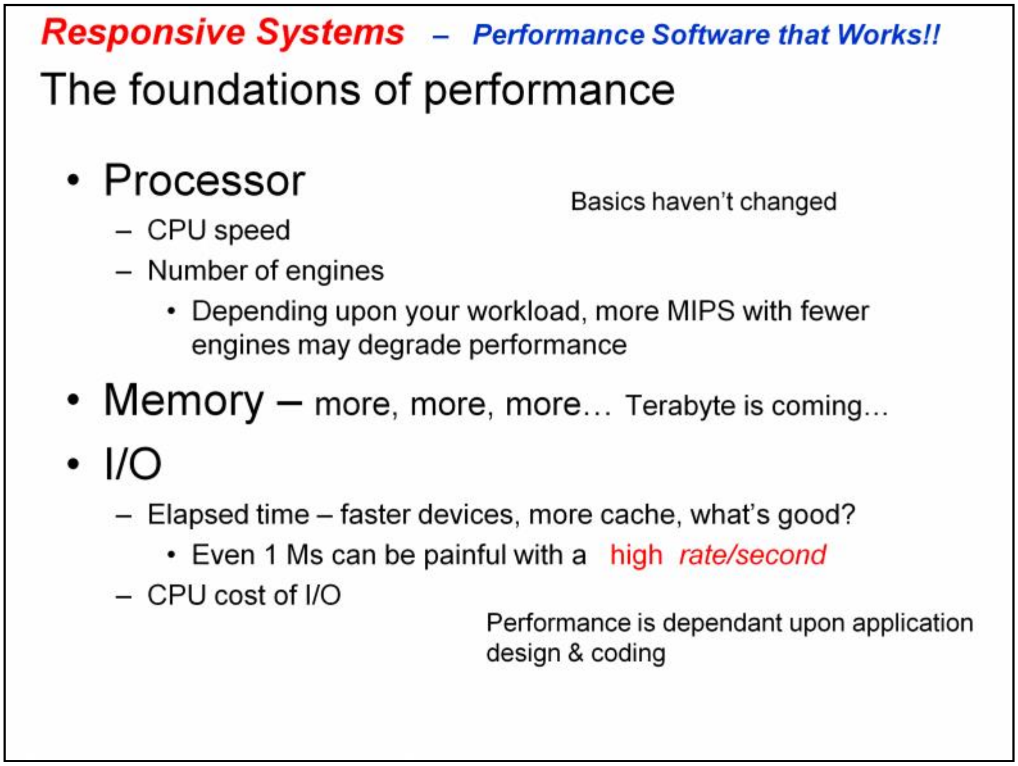


DB2 is a complex system, with a major impact upon your processing environment. There are substantial performance and instrumentation changes in versions 9, 10, 11. that must be used to measure, evaluate, and quantify the performance of the DB2 system and applications. This presentation addresses the origins and specific data available for analyzing, tuning, and tracking the performance of the system and applications, and the inter-dependencies and relationships between the various data elements. Our performance world has changed a lot since this topic was originally presented a dozen years ago. The presentation assumes the attendee has a good understanding of DB2, and does not define basic DB2 terminology.



The basics of performance have not changed within the last 50 years.

Responsive Systems – *Performance Software that Works!!*

Analysis Data

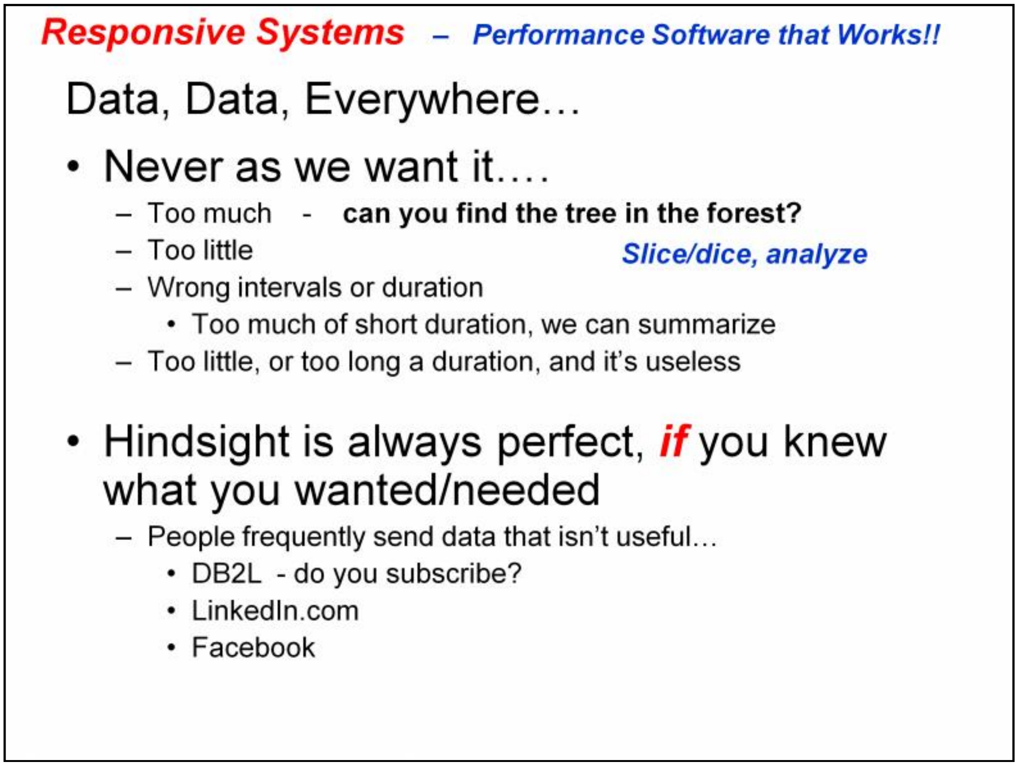
- System Data
  - z/OS data
    - RMF data, from SMF
      - CPU, memory, paging, components of dasd response, etc .....
  - DB2 performance data
    - System statistics - *one minute intervals* - *summarize*
    - Application accounting – large volume
    - Performance traces
      - SMF, GTF, IFI, various vendor products
    - Displays – Lstats, etc
      - 199 Records - **better than nothing, but limited value**
        - » Only objects > 1 IO/Sec

Data from multiple sources must be used for tuning. If you don't know what data you need, how to get, and how to analyze it, your tuning effectiveness will….

be less than management wants/needs. Producing Stats records every 60 seconds is only 1440 records, not a problem. This lets you find problems with a low level of granularity, but start from a higher level of initial analysis like 15 or 30 minutes.

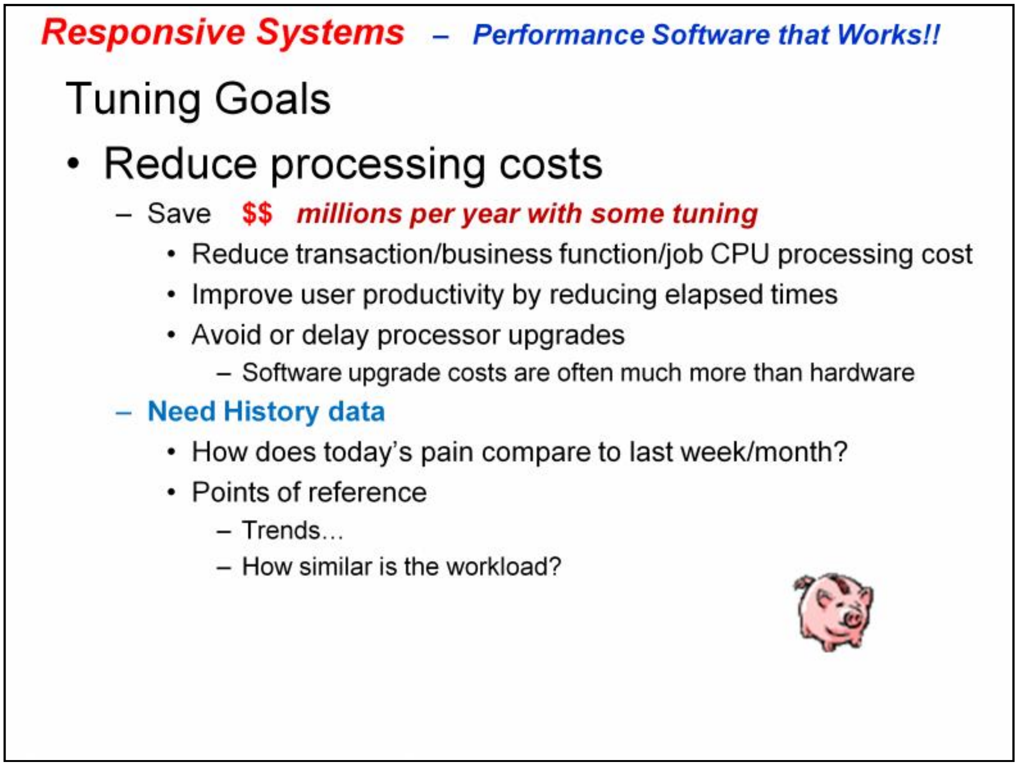
While using anything is vastly better than not tuning, it's important to understand the limitations of data. While Lstat displays, and 199 object usage records will provide some useful information, tuning from there is simply a guess. There is no validation without changing your system. Can you afford guesswork on your production system?

Making a pool larger and monitor performance won't hurt if you don't impact system paging. Move an object to the wrong pool, and you're in big trouble.



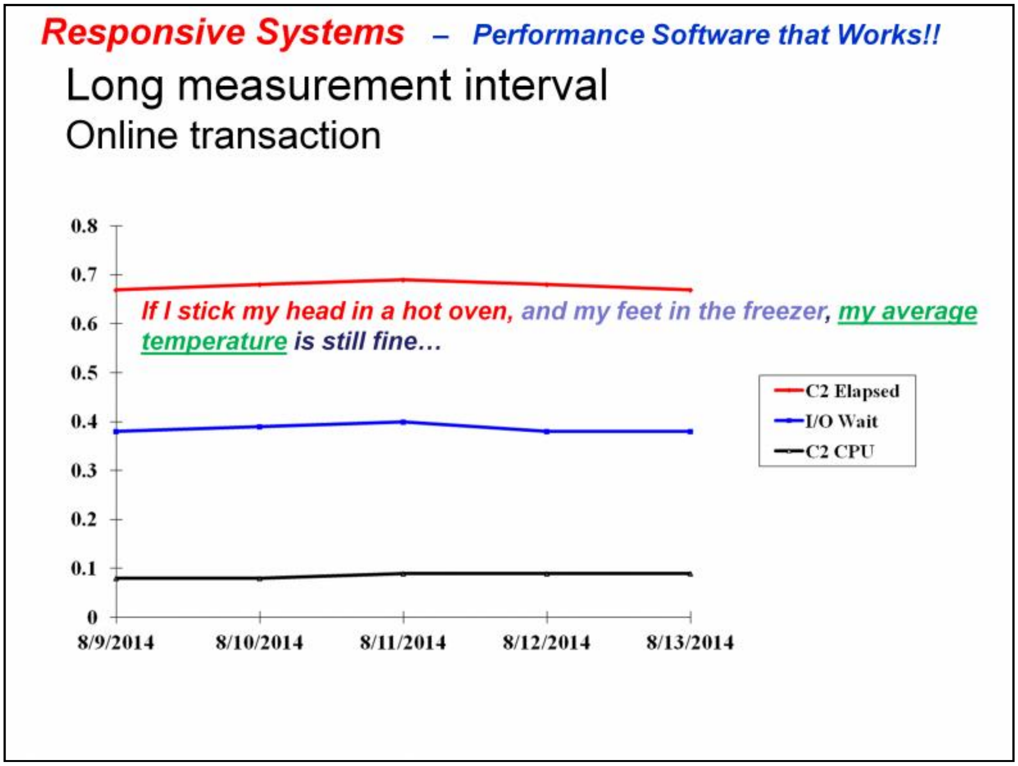
We need the ability, and *tools* to slice/dice data many different ways.

Putting data into Excel can open a whole new world of analysis.

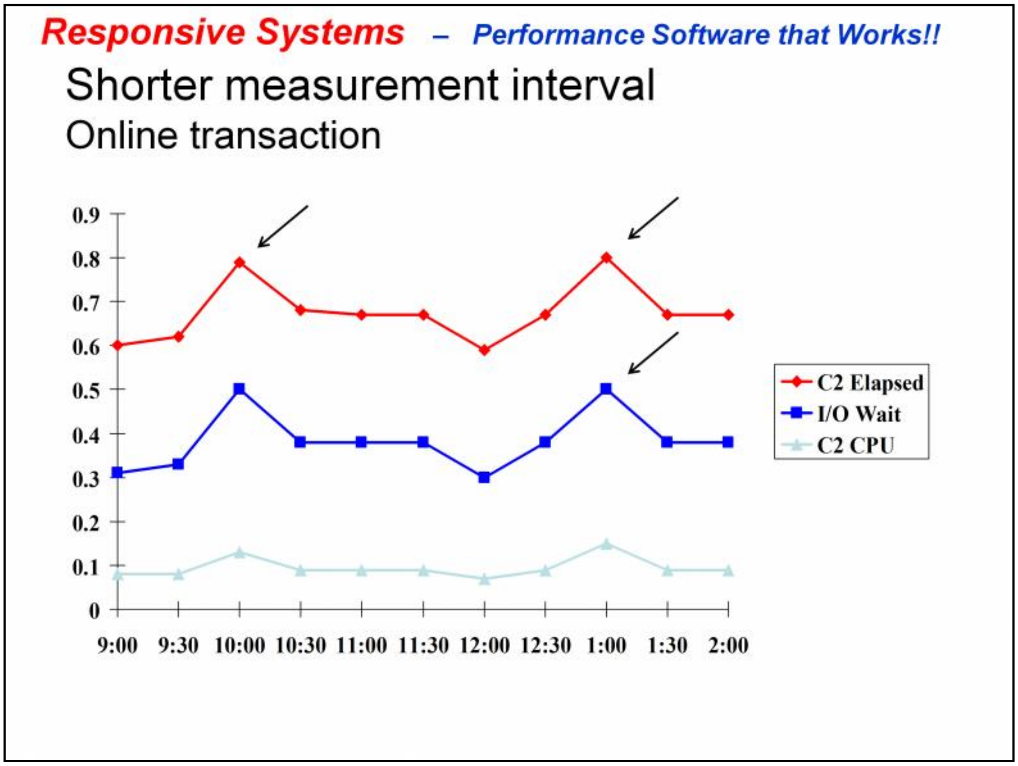


Large companies, worldwide, are wasting millions per year by ignoring tuning opportunities.

History data lets you see if this is a new problem, or an old issue that nobody complained about before. Has it always been this bad, or has it gotten worse over time?



Long measurement intervals hide performance problems by showing averages over a long period. An interval spanning a hour or two during peak load periods would be valid in most cases to get average response times. However, when periods of low system activity are factored into the data, it becomes meaningless – does not represent the times you should be most concerned with – the peak periods.



Shorter periods show the peaks and valleys. From here you should go to 5 minute periods between 9:45 – 10:15, and 12:45 – 13:15.

If necessary, home into a couple of 1 minute periods to isolate the worst (highest) points.

Then look into detailed application and system performance data to determine the cause of the response time spikes.

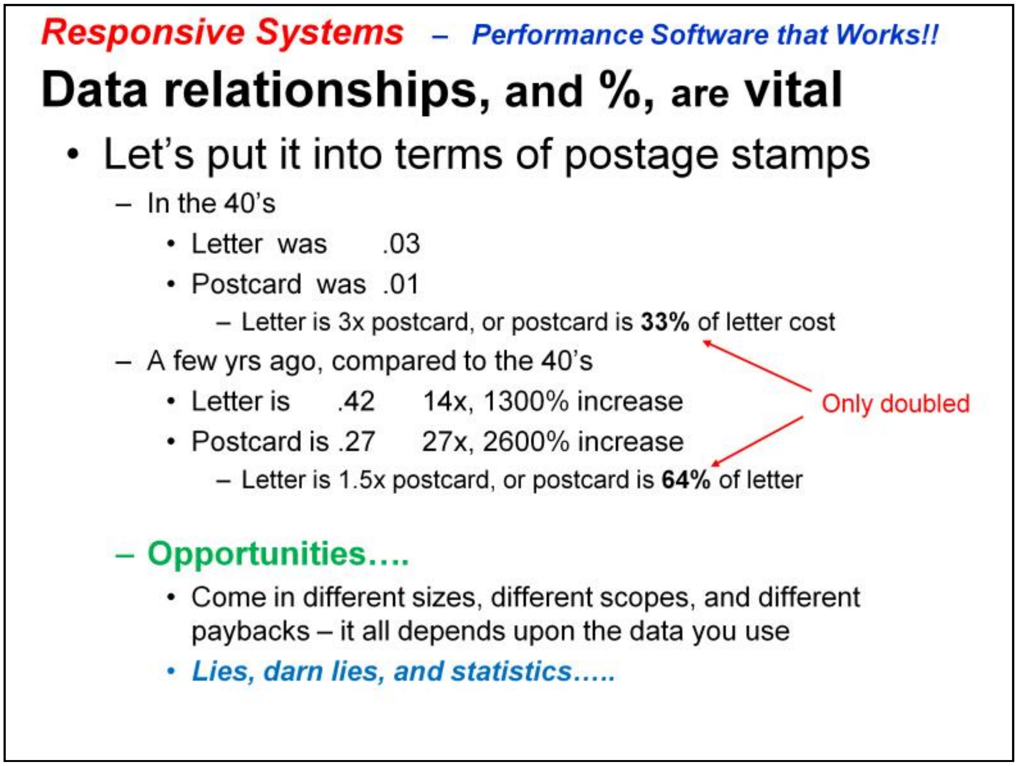
What can be done to reduce the I/O wait time issues?

Responsive Systems – *Performance Software that Works!!*

Data relationships, and %, are vital

- Sometimes the raw numbers get our attention
  - Transaction elapsed times of 5, 10, 20 seconds
- Sometimes percentages are more meaningful
  - Transaction elapsed .1 secs, .08 I/O wait
    - *I/O wait is 80% of the elapsed time*
- What other factors or events might have impacted the numbers you are looking at?
  - Elapsed times are usually < 1 sec, but spike to 10 secs for some periods
    - Who does what, to whom…?? ***z/OS, DB2, or Application?***

Eyeball method – what looks big or out of proportion to other data?



We can play many games with data, depending upon what we want to achieve, or prove.

Lies, darn lies, and statistics……….

Statistics can prove, or disprove anything, depending upon the data sample.

Responsive Systems – *Performance Software that Works!!*

The biggest CPU payback

- In your applications
  - Indexing
  - SQL coding
  - Design

CPU cost is proportional to the number of pages you process

- Where's the payback?
  - You can usually find it easily
    - If you look in the right places

- ***Document your savings***
  - **blow your own horn, let people know, but don't be obnoxious about it**

Finding and fixing one object with heavy scan activity, continually, will pay for your salary many times over.

We have several clients over the years that have cancelled their processor upgrades after some tuning…

Responsive Systems – *Performance Software that Works!!*

Where is the performance problem?

```
ELAPSED TIME DISTRIBUTION                                      CLASS 2 TIME DISTRIBUTION
----------------------------------------------------------     ------------------------------------------------
APPL  |=> 3%                                                   CPU    |===========> 23%
DB2   |============> 25%                                       NOTACC |=> 3%
SUSP  |====================================> 72%               SUSP   |====================================> 74%

AVERAGE       APPL(CL.1)  DB2 (CL.2)  IFI (CL.5)   CLASS 3 SUSPENSIONS   AVERAGE TIME  AV.EVENT   HIGHLIGHTS
------------  ----------  ----------  ----------   --------------------  ------------  --------   --------------------------
ELAPSED TIME  1:12:59.61  1:10:33.01    0.002291   LOCK/LATCH(DB2+IRLM)     12.692490    205.4K   #OCCURRENCES       :       1
 NONNESTED    1:12:59.55  1:10:32.95         N/A   SYNCHRON. I/O          46:58.120875    488.4K   #ALLIEDS           :       1
 STORED PROC    0.000000    0.000000         N/A    DATABASE I/O          46:54.918926    488.1K   #ALLIEDS DISTRIB:          0
 UDF            0.003857    0.002496         N/A    LOG WRITE I/O             3.201950    329.00   #DBATS             :       0
 TRIGGER        0.054189    0.054189         N/A   OTHER READ I/O          4:26.612175  30076.00   #DBATS DISTRIB.   :        0
                                                   OTHER WRTE I/O            39.323326   8456.00   #NO PROGRAM DATA:          0
CPU TIME      18:27.0140  16:13.0057    0.000270   SER.TASK SWTCH             9.201761   1487.00   #NORMAL TERMINAT:          1
 AGENT        18:27.0140  16:13.0057         N/A    UPDATE COMMIT             8.015826   1459.00   #ABNORMAL TERMIN:          0
  NONNESTED   18:27.0118  16:13.0040    0.000270    OPEN/CLOSE                0.000000      0.00   #CP/X PARALLEL.   :        0
  STORED PRC    0.000000    0.000000         N/A    SYSLGRNG REC              0.317184     22.00   #IO PARALLELISM   :        0
  UDF           0.001029    0.000472         N/A    EXT/DEL/DEF               0.866745      5.00   #INCREMENT. BIND:          0
  TRIGGER       0.001170    0.001170         N/A    OTHER SERVICE             0.002006      1.00   #COMMITS           :    1519
 PAR.TASKS      0.000000    0.000000         N/A   ARC.LOG(QUIES)             0.000000      0.00   #ROLLBACKS         :       0
                                                   ARC.LOG READ               0.000000      0.00   #SVPT REQUESTS     :       0
SUSPEND TIME    0.000765  52:26.1084         N/A   DRAIN LOCK                 0.000000      0.00   #SVPT RELEASE      :       0
 AGENT               N/A  52:26.1084         N/A   CLAIM RELEASE              0.000000      0.00   #SVPT ROLLBACK     :       0
 PAR.TASKS           N/A    0.000000         N/A   PAGE LATCH                 0.075009     11.00   MAX SQL CASC LVL:          2
 STORED PROC    0.000000         N/A         N/A   NOTIFY MSGS                0.000000      0.00   UPDATE/COMMIT      :  849.63
 UDF            0.000765         N/A         N/A   GLOBAL CONTENTION          0.074649     42.00   SYNCH I/O AVG.     : 0.005770
                                                   COMMIT PH1 WRITE I/O       0.000000      0.00   ROLLUP PLAN        : NO
NOT ACCOUNT.         N/A  1:53.89113         N/A   ASYNCH CF REQUESTS         0.008067     72.00
DB2 ENT/EXIT         N/A    34501.0K         N/A   TOTAL CLASS 3          52:26.108353    734.0K
EN/EX-STPROC         N/A        0.00         N/A
EN/EX-UDF            N/A        6.00         N/A      TOT4K  BPOOL ACTIVITY      AVERAGE     TOTAL
DCAPT.DESCR.         N/A         N/A    0.000000      ---------------------    --------  --------
LOG EXTRACT.         N/A         N/A    0.000000      BPOOL HIT RATIO (%)         94.87       N/A
                                                      GETPAGES                 39955.2K  39955181
                                                      GETPAGES-FAILED              0.00         0
                                                      BUFFER UPDATES            9129.3K   9129348
                                                      SYNCHRONOUS WRITE            0.00         0
                                                      SYNCHRONOUS READ           488.1K    488092
                                                      SEQ. PREFETCH REQS       49309.00     49309
                                                      LIST PREFETCH REQS           0.00         0
                                                      DYN. PREFETCH REQS         166.8K    166797
                                                      PAGES READ ASYNCHR.       1560.1K   1560126
```

While Hit Ratios aren't useful as a real performance metric, **a large % of the requested pages were found in the pool.**

Application accounting report for a batch job. Job is running too long, DBA said they had a buffer pool tuning problem, and the systems people would not tune the pools.

While pool tuning might reduce overall I/O and thus the elapsed time, the underlying issues are:

a. Death by synch I/O – possibly sort the input into the key order of the data

b. The cache on the DASD controller is too small

c. The synch I/O elapsed time is poor, that relates to b.

Speaking with the sysprogs, and analyzing system performance data, pools cannot be increased without causing the system to page.

They know they have a DASD performance issue, the cache is too small, and DASD upgrades are a few months in the future.

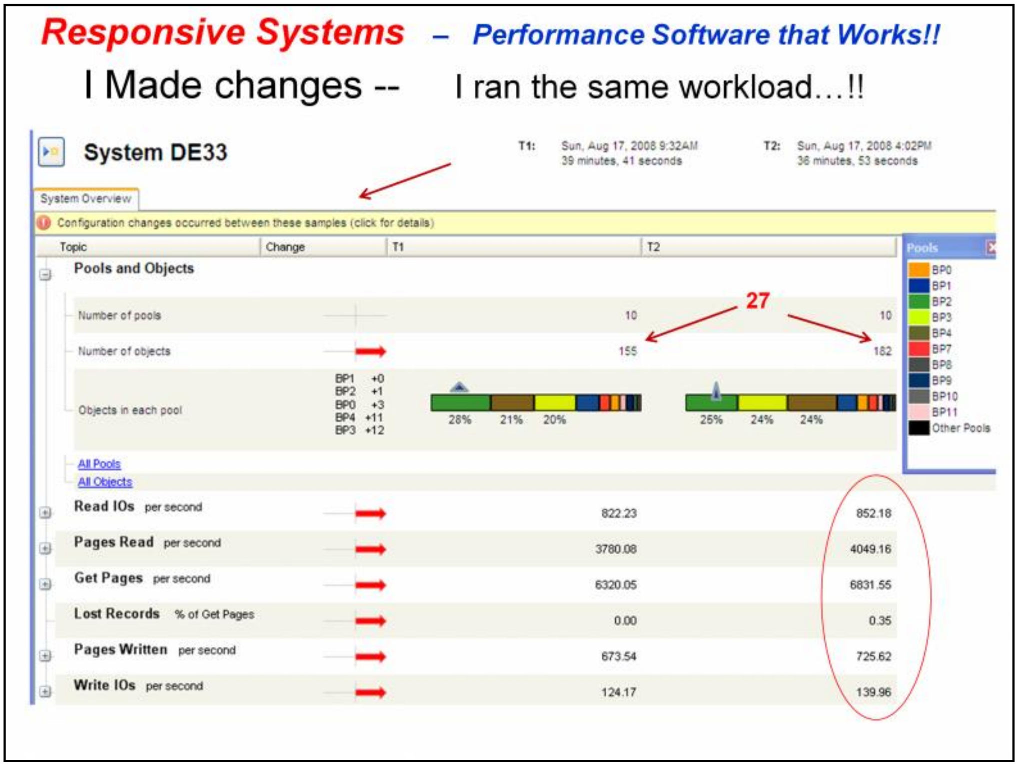
Responsive Systems – *Performance Software that Works!!*

Made tuning changes
is performance better now?

- Workload comparisons must be *reasonably* similar
  - Quite common to have significant differences of object usage, objects accessed, type of access
    - Same timeframe?
    - Length of comparison interval
  - Is your workload infinitely variable?
    - Very difficult to measure
      - Object activity inter-relationships
      - Dynamic queries?
      - Batch jobs all day?

Need multiple measurement points, both before and after

One of the most difficult problems when comparing performance, is having workloads that are reasonably similar. Unless you have a stand-alone test system, with a specific driven workload, you will never achieve an exact comparison. So the key for comparisons is – "reasonably similar".



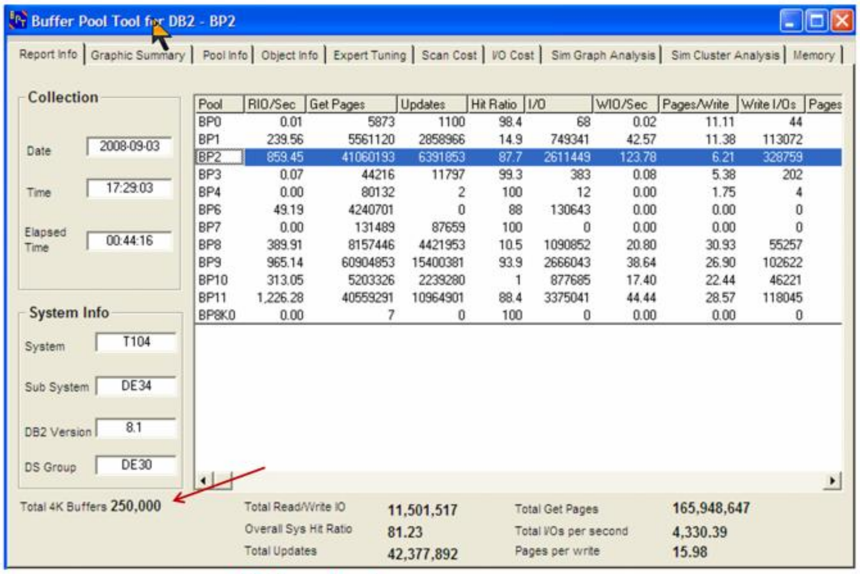
Client made pool tuning changes to measure the improvement of a batch workload. Adamant that they ran the exact same workload, and complained that the I/O rate increased.

However 27 additional (new) objects were accessed during the second performance measurement period. The Getpage rate was 10% higher, number of pages written, and write I/O was higher.

The workload in the overall system was quite different.

Responsive Systems – *Performance Software that Works!!*

Pool performance analysis – *before 1a*



Pool	RIO/Sec	Get Pages	Updates	Hit Ratio	I/O	WIO/Sec	Pages/Write	Write I/Os	Pages
BP0	0.01	5873	1100	98.4	68	0.02	11.11	44	
BP1	239.56	5561120	2858966	14.9	749341	42.57	11.38	113072	
BP2	859.45	41060193	6391853	87.7	2611449	123.78	6.21	328759	
BP3	0.07	44216	11797	99.3	383	0.08	5.38	202	
BP4	0.00	80132	2	100	12	0.00	1.75	4	
BP6	49.19	4240701	0	88	130643	0.00	0.00	0	
BP7	0.00	131489	87659	100	0	0.00	0.00	0	
BP8	389.91	8157446	4421953	10.5	1090852	20.80	30.93	55257	
BP9	965.14	60904853	15400381	93.9	2666043	38.64	26.90	102622	
BP10	313.05	5203326	2239280	1	877685	17.40	22.44	46221	
BP11	1,226.28	40559291	10964901	88.4	3375041	44.44	28.57	118045	
BP8K0	0.00	7	0	100	0	0.00	0.00	0	

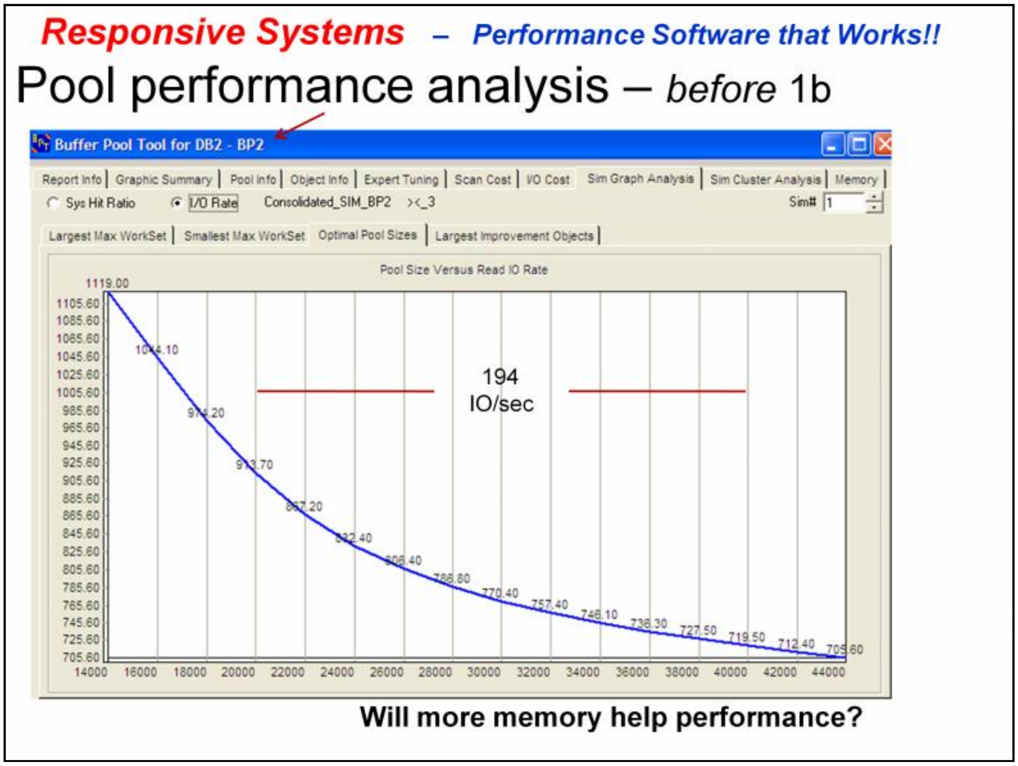
Before tuning

The following performance data is after moving to a larger, faster machine, with more memory – and other workloads.

High overall I/O rate, spread across several pools

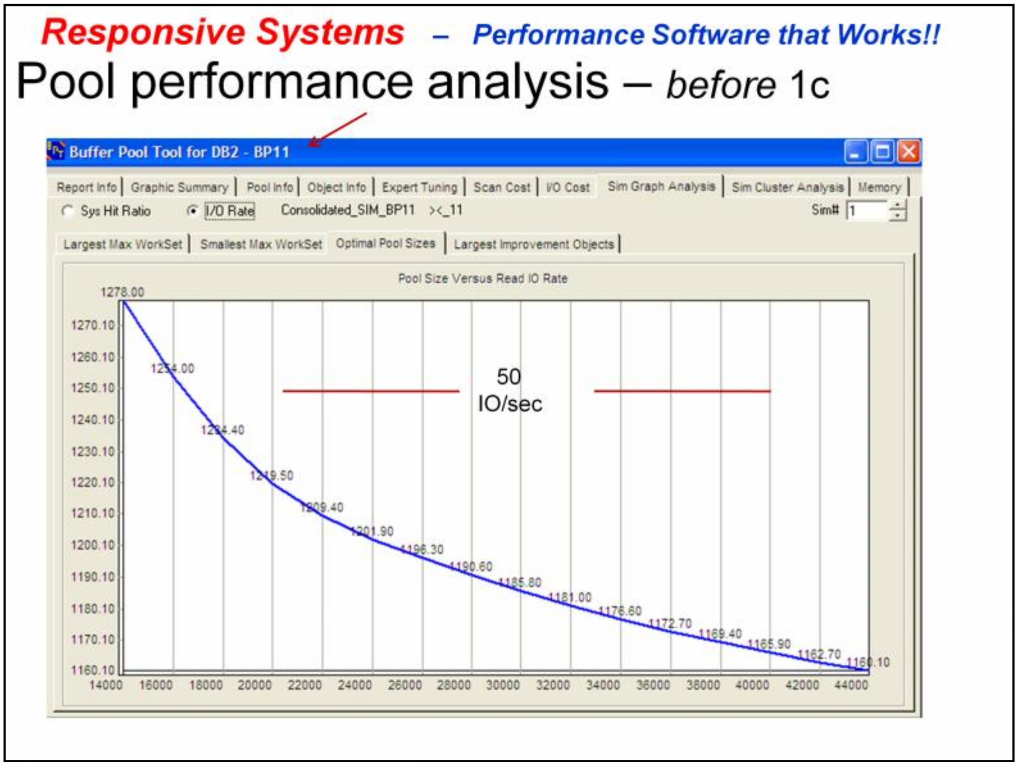
Before tuning, making some pools larger

The above and following slides titled "before" look at the current performance, and the "predicted" improvements of pool size increases



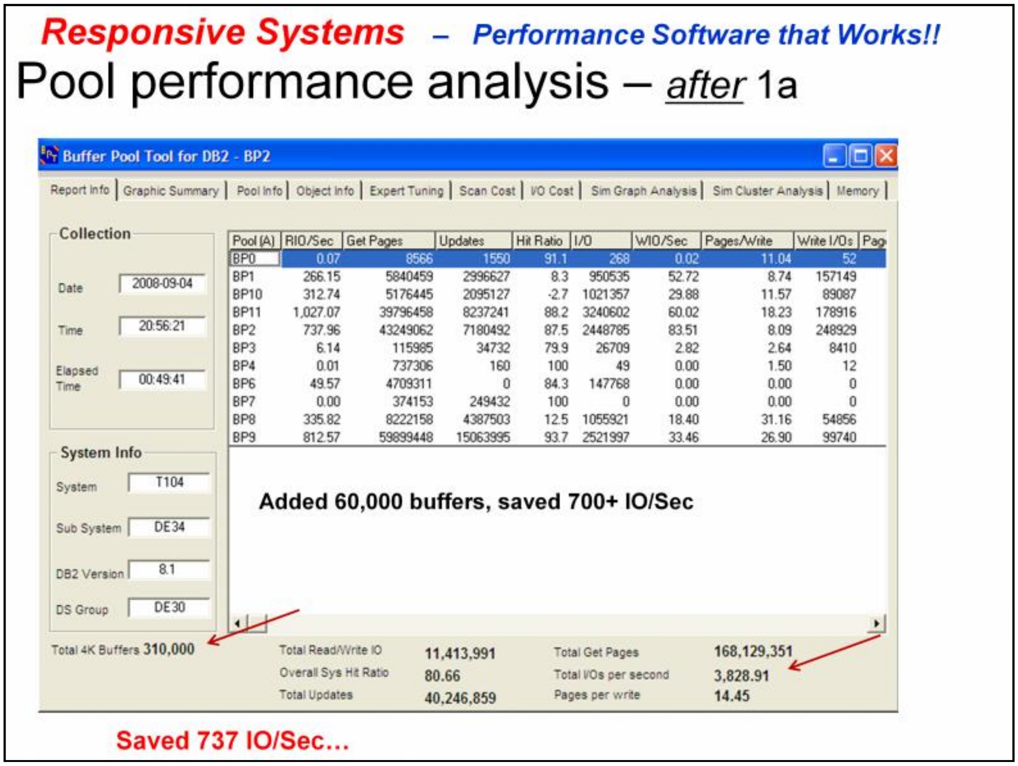
Predicted saving by doubling BP2 from 20,000 to 40,000 buffers, a bit less than 200 I/O second

Before tuning, making some pools larger….



Predicted saving by doubling BP11 from 20,000 to 40,000 buffers, about 50 I/O sec, 1165 at 40,000.

Before tuning, making some pools larger….



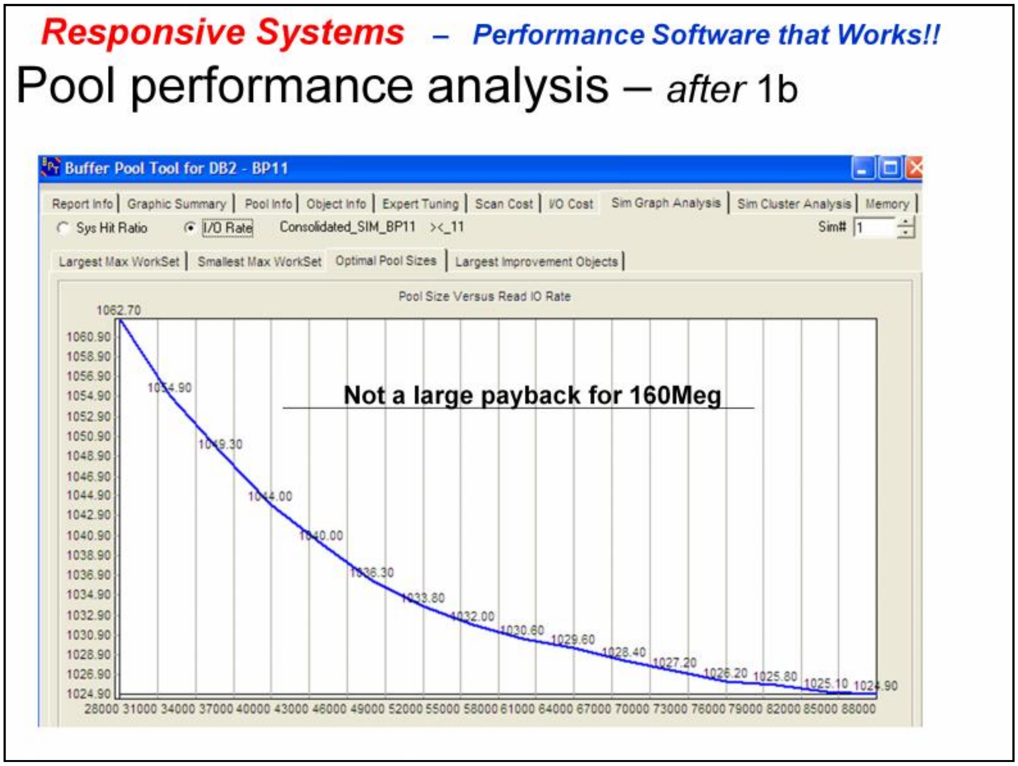
After the pools were enlarged…. After base pool tuning.

High overall I/O rate, spread across several pools

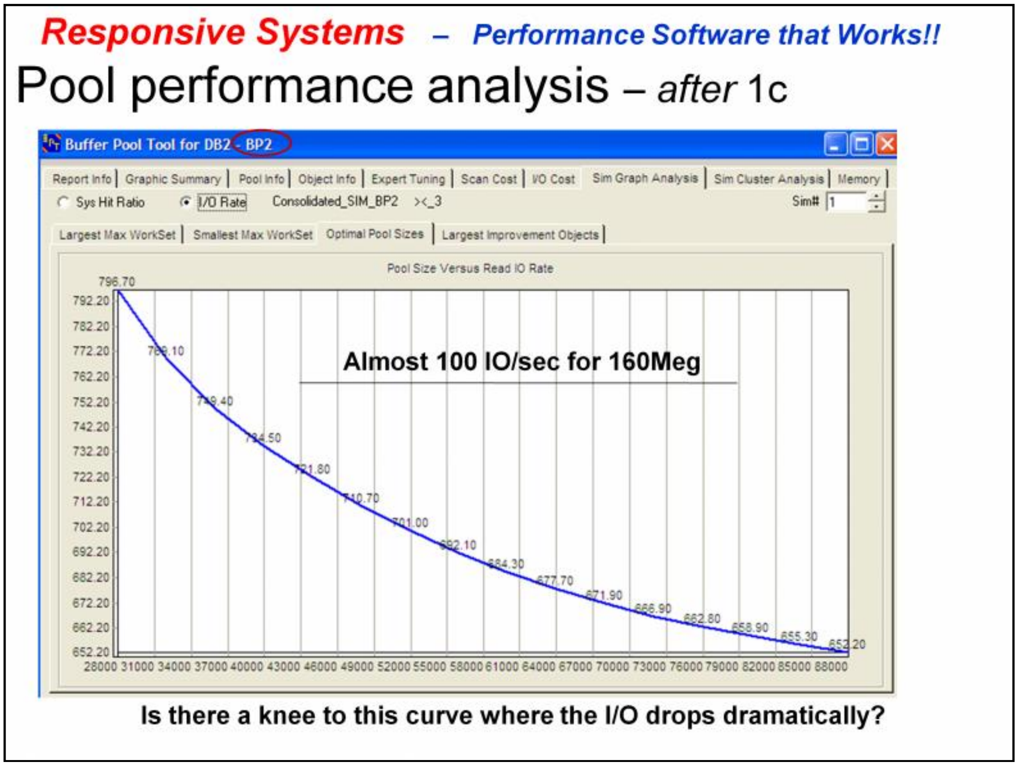
The I/O rate dropped for both BP2 and BP9 that were increased.

We predicted 720 I/O second at 40,000 buffers, we got 737.

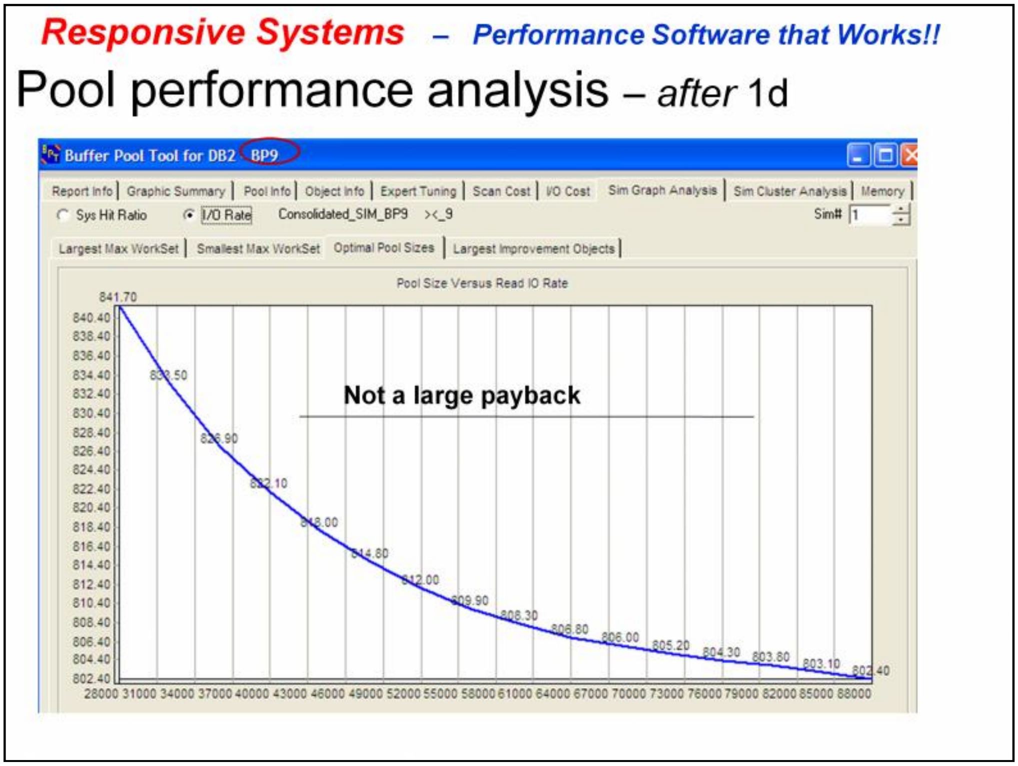
There are still variations in the workload.



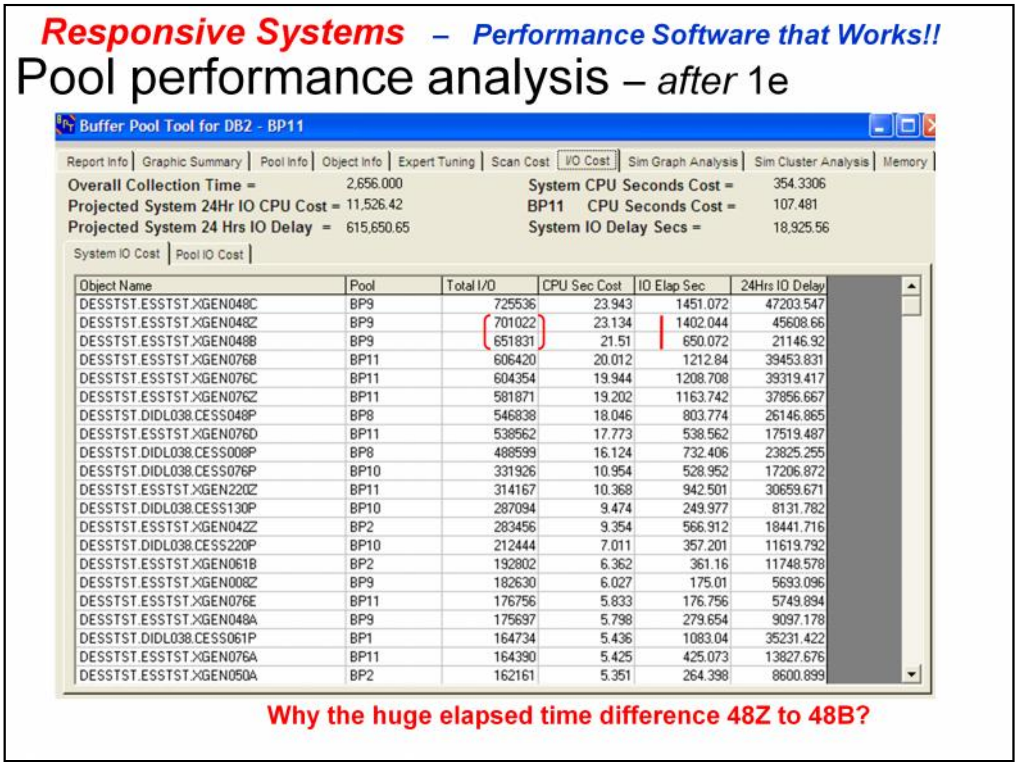
Additional doubling BP11 from 40,000 to 80,000 buffers will save about 20 I/O second. Not a large payback, but later slide will imply that we might do better because more pages are found in the DASD cache..



Doubling BP2 from 40,000 to 80,000 saves almost 100 I/O second. There might be a bit more to be gained, but the curve is flattening rapidly.

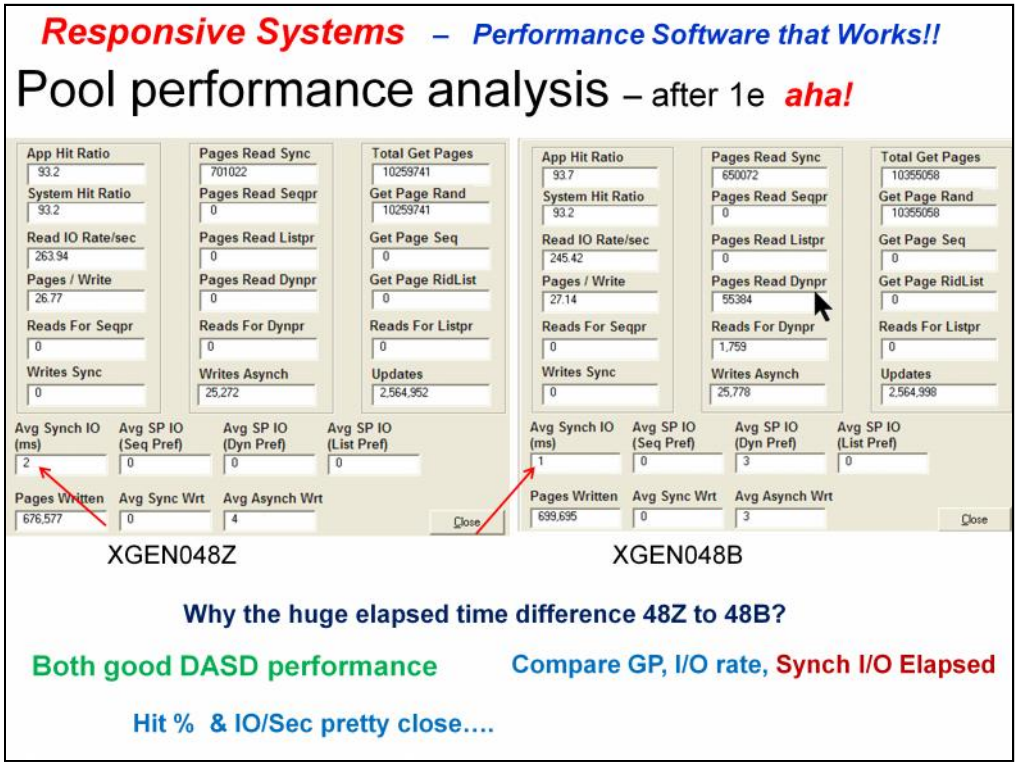


Doubling BP9 from 40,000 to 80,000 saves about 20 I/O second. Not a large payback, and the curve flattens as we give it more memory.



Object Name	Pool	Total I/O	CPU Sec Cost	IO Elap Sec	24Hrs IO Delay
DESSTST.ESSTST.XGEN048C	BP9	725536	23.943	1451.072	47203.547
DESSTST.ESSTST.XGEN048Z	BP9	701022	23.134	1402.044	45608.66
DESSTST.ESSTST.XGEN048B	BP9	651831	21.51	650.072	21146.92
DESSTST.ESSTST.XGEN076B	BP11	606420	20.012	1212.84	39453.831
DESSTST.ESSTST.XGEN076C	BP11	604354	19.944	1208.708	39319.417
DESSTST.ESSTST.XGEN076Z	BP11	581871	19.202	1163.742	37856.667
DESSTST.DIDL038.CESS048P	BP8	546838	18.046	803.774	26146.865
DESSTST.ESSTST.XGEN076D	BP11	538562	17.773	538.562	17519.487
DESSTST.DIDL038.CESS008P	BP8	488599	16.124	732.406	23825.255
DESSTST.DIDL038.CESS076P	BP10	331926	10.954	528.952	17206.872
DESSTST.ESSTST.XGEN220Z	BP11	314167	10.368	942.501	30659.671
DESSTST.DIDL038.CESS130P	BP10	287094	9.474	249.977	8131.782
DESSTST.ESSTST.XGEN042Z	BP2	283456	9.354	566.912	18441.716
DESSTST.DIDL038.CESS220P	BP10	212444	7.011	357.201	11619.792
DESSTST.ESSTST.XGEN061B	BP2	192802	6.362	361.16	11748.578
DESSTST.ESSTST.XGEN008Z	BP9	182630	6.027	175.01	5693.096
DESSTST.ESSTST.XGEN076E	BP11	176756	5.833	176.756	5749.894
DESSTST.ESSTST.XGEN048A	BP9	175697	5.798	279.654	9097.178
DESSTST.DIDL038.CESS061P	BP1	164734	5.436	1083.04	35231.422
DESSTST.ESSTST.XGEN076A	BP11	164390	5.425	425.073	13827.676
DESSTST.ESSTST.XGEN050A	BP2	162161	5.351	264.398	8600.899

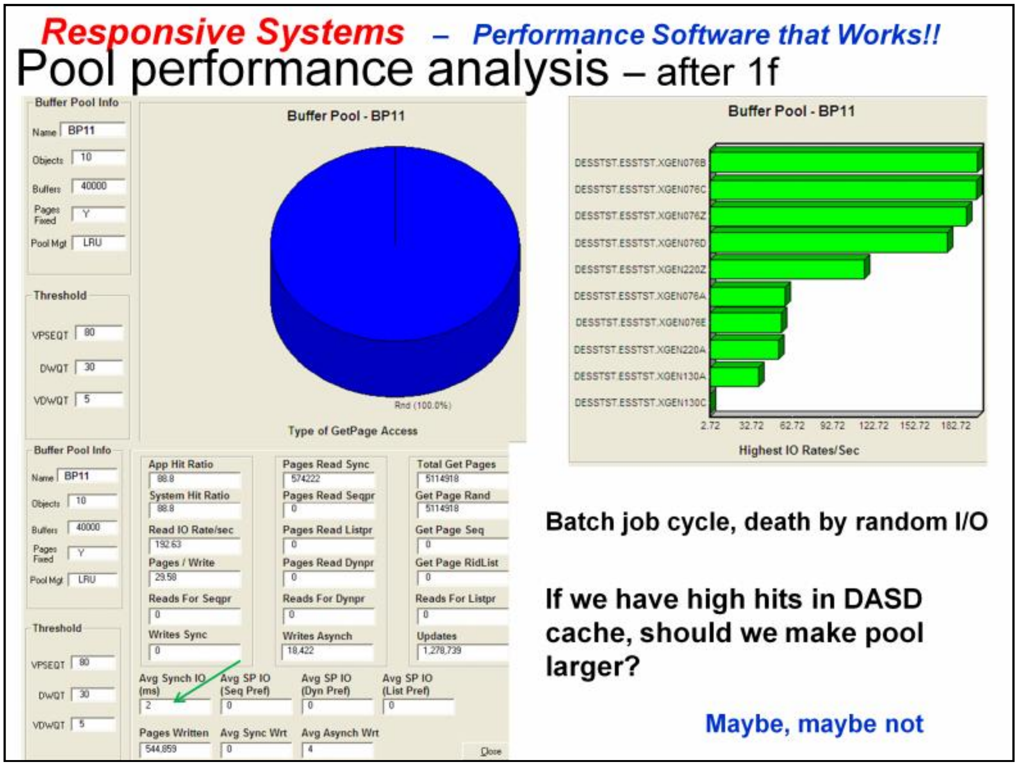
This illustrates the objects with the highest I/O rates across the system, shows how many CPU seconds of CPU are caused by the I/O, and the application elapsed time effects of the I/O. The difference in IO is less than 10%, but more than twice the IO Elapsed seconds. The underlying reason for the huge difference between the IO elapsed times, is because of the DASD response times, as you will see on the next slide.



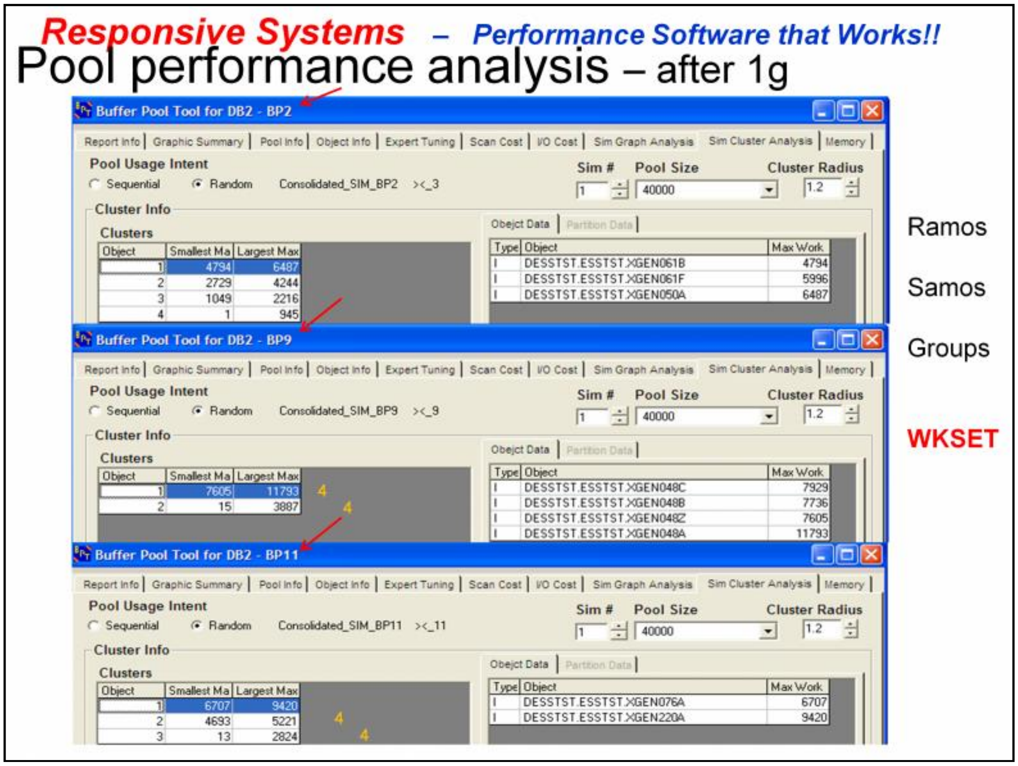
48B has a higher number of GP, a lower I/O rate, and lower Sync I/O times.

48B found more pages in the pool, and more pages in the DASD cache.

Synch IO is noticeably lower



BP11 is 100% random access, we can see the top 10 I/O objects in the pool, and then see the performance characteristics of each object. Overall avg. synch I/O times of 2 Ms, is very good, means almost all pages are found in the DASD cache. That also tells us that making the pool larger will find more pages in the BP, and reduce the I/O rate, and save CPU cycles.



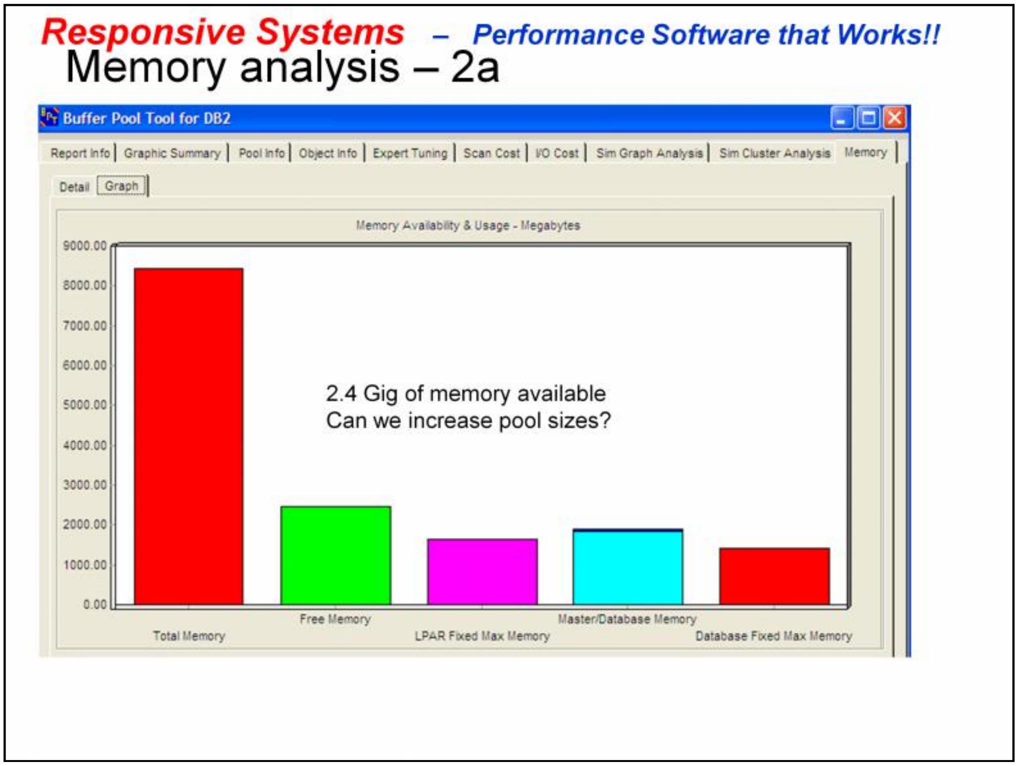
Working set size is often a crucial factor used for splitting objects into pools.

The general methodology, is Ramos/Samos – (randomly accessed, mostly), and (sequentially accessed mostly); then within those criteria, very large working set objects from the rest. The goal is to separate objects that monopolize a pool.

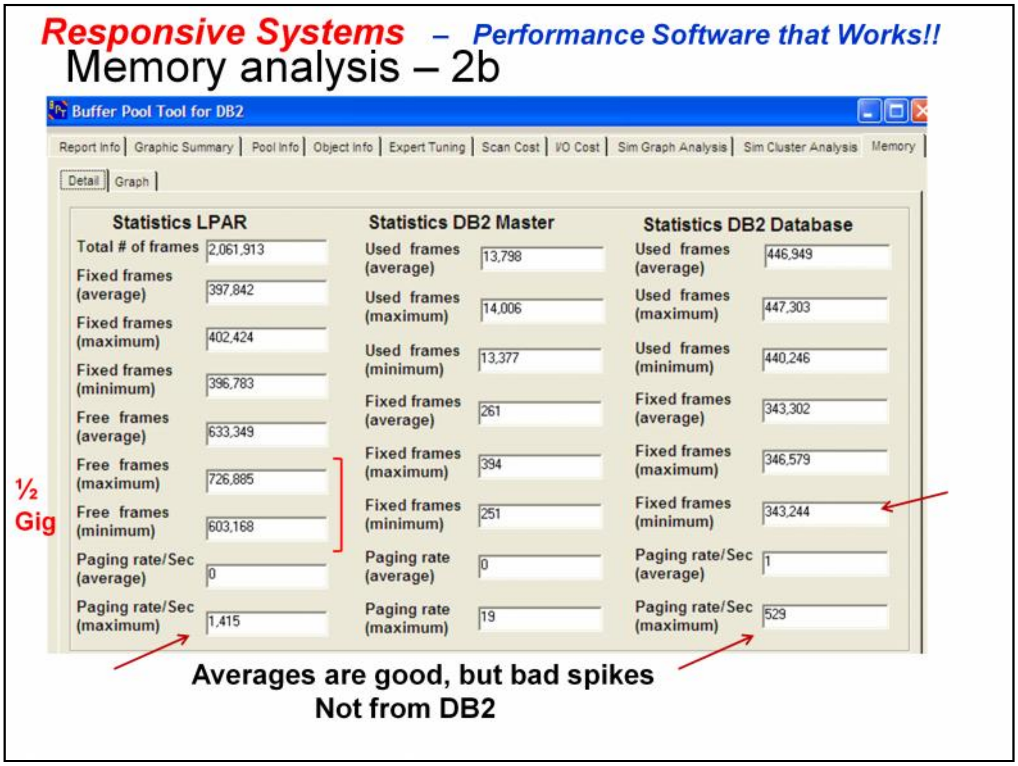
The first obstacle we encounter in this overall workload, is that it's almost all random.

Based on the working set sizes in each pool, there probably isn't much opportunity for gain by splitting out objects.

BP9 is a maybe, based on 4 objects in each cluster.

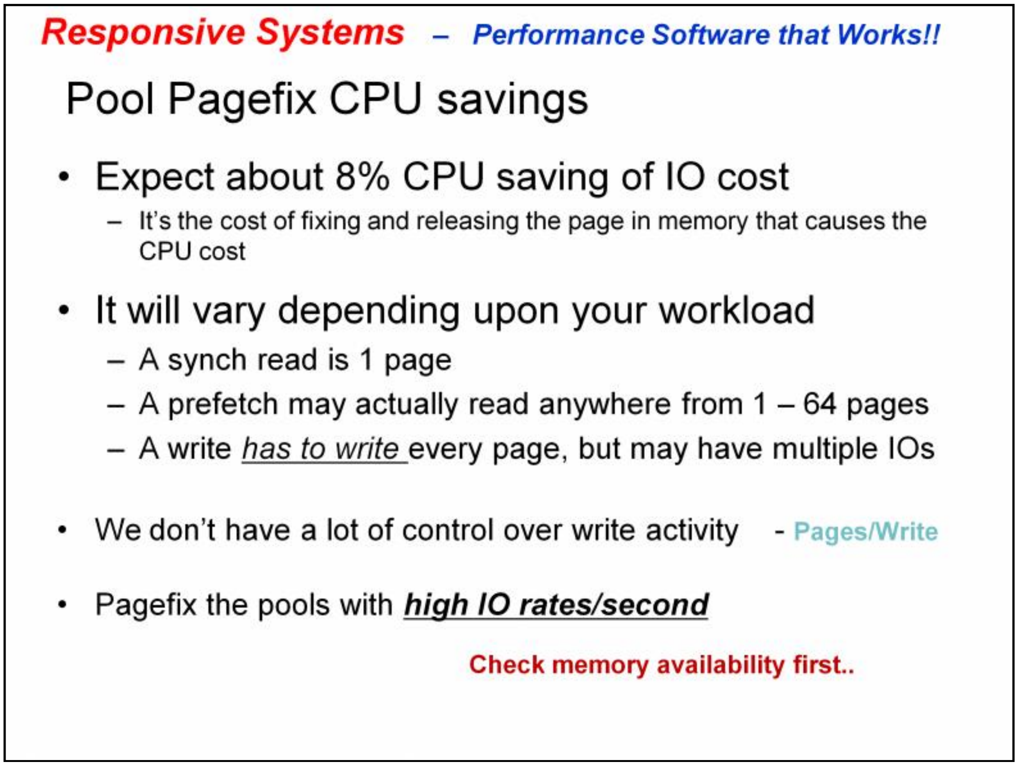


There is 2 Gig of memory available on the LPAR, can we increase any pools?



Even with 2.4 Gig available, the system has shown paging activity over the monitored interval. Pool increases might hurt overall performance.

A frame is 4K.



The number of pages actually read or written will determine the saving.

If a prefetch "can" read 64 pages…. If the pool is pagefixed, an actual read of 64 pages, will "save" more than a prefetch that only has to read 10…

Responsive Systems – ***Performance Software that Works!!***

Pool Pagefix CPU savings

- Additional consideration
  - Size of the pool(s)
- Two pools with 500+ IO/Sec
  - One has 50,000 buffers
  - The other has 150,000 buffers
- Which would you Pagefix?
  - Biggest bang for the buck…?
  - How much avg **free** memory do you have?

Responsive Systems – *Performance Software that Works!!*

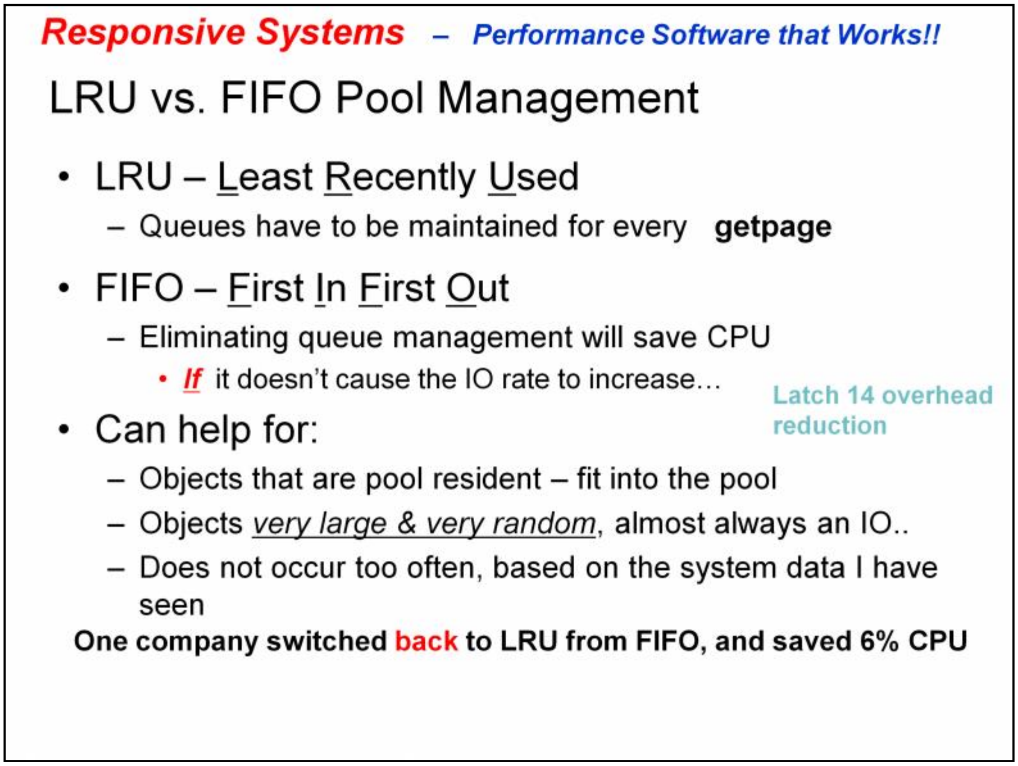
Pool Pagefix CPU savings

DATE	SRB CPUTM DBM1	Prefetch Read I/Os	Total Sync Read I/O's	IOCOUNT	CPU Seconds DBM1	Prefetch CPU Cost per IO CPU Seconds	% Saving
28-Apr-08	0:54:04.31	91,066,400	41,435,109	132,501,509	3244	0.000035622	Base
29-Apr-08	0:57:23.67	98,631,004	41,811,306	140,442,310	3443	0.000034908	
Pools Pagefixed							
1-May-08	0:56:42.66	106,118,184	43,969,479	150,087,663	3402	0.000032059	10.00%
2-May-08	0:56:27.99	97,573,050	45,343,642	142,916,692			
7-May-08	0:53:48.67	90,425,961	42,672,240	133,098,201	3228	0.000035698	-0.21%
8-May-08	0:57:34.25	92,418,870	44,804,753	137,223,623			
12-May-08	1:02:57.66	134,339,741	47,452,124	181,791,865	3677	0.000027371	23.16%
13-May-08	1:09:30.06	157,009,921	47,800,608	204,810,529	4080	0.000025986	27.05%
15-May-08	1:01:08.25	128,107,975	44,278,455	172,386,430			
16-May-08	1:05:45.95	129,716,219	45,207,337	174,923,556	3945	0.000030413	14.63%
19-May-08	0:58:54.02	121,134,552	37,964,597	159,099,149			
20-May-08	1:05:05.63	130,500,306	44,076,187	174,576,493	3905	0.000029923	16.00%
21-May-08	1:02:24.31	123,246,902	41,336,231	164,583,133			
9-Jun-08	0:48:25.30	89,988,804	37,248,998	127,237,802	2905	0.000032282	9.38%
10-Jun-08	0:49:14.31	93,629,950	36,642,766	130,272,716	2954	0.000031550	11.43%
11-Jun-08	0:48:18.86	92,112,926	37,892,212	130,005,138	2898	0.000031461	11.87%
12-Jun-08	0:51:20.76	95,807,846	39,655,823	135,463,669	3080	0.000032148	9.75%

Prefetch is not the only user of SRB CPU in DBM1, is is the predominant cause.

Synch IO is charged to the application TCB

We normally estimate about 8% at the application level.



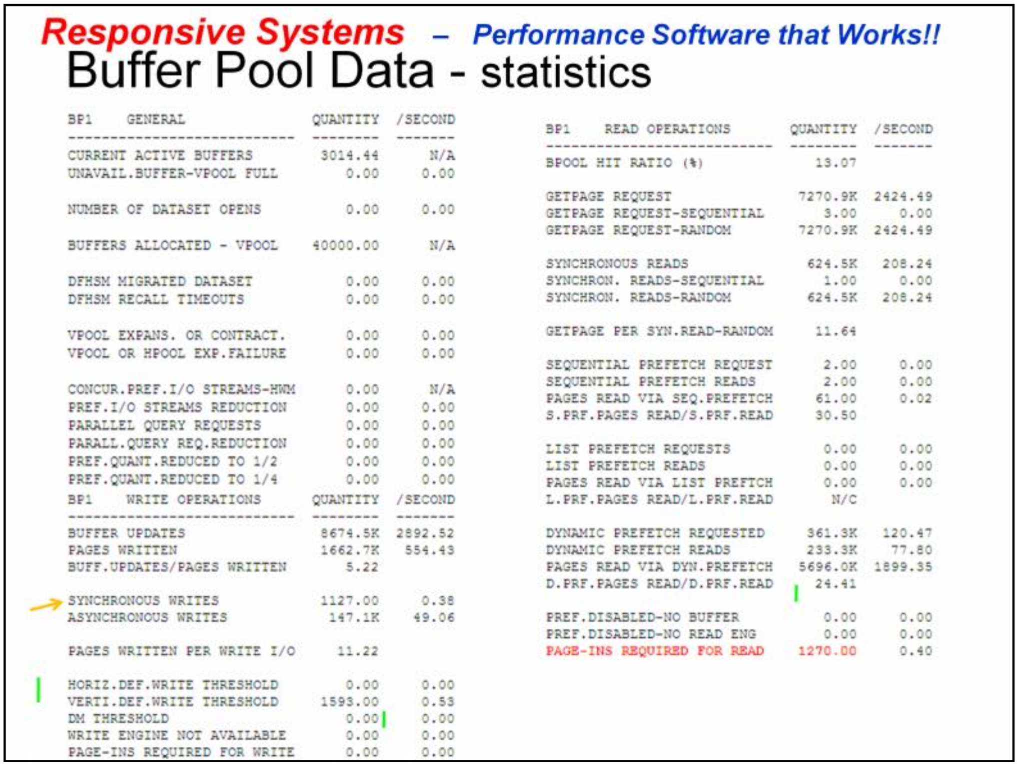
The number of pages actually read or written will determine the saving.

If a prefetch "can" read 64 pages…. If the pool is pagefixed, an actual read of 64 pages, will "save" more than a prefetch that only has to read 10…

The key to using FIFO is either:

a. Objects have no IO

b. Almost totally random, and rarely find a page in the pool – always do an IO



Responsive Systems – Performance Software that Works!!

Buffer Pool Data - statistics

BP1 GENERAL	QUANTITY	/SECOND
CURRENT ACTIVE BUFFERS	3014.44	N/A
UNAVAIL.BUFFER-VPOOL FULL	0.00	0.00
NUMBER OF DATASET OPENS	0.00	0.00
BUFFERS ALLOCATED - VPOOL	40000.00	N/A
DFHSM MIGRATED DATASET	0.00	0.00
DFHSM RECALL TIMEOUTS	0.00	0.00
VPOOL EXPANS. OR CONTRACT.	0.00	0.00
VPOOL OR HPOOL EXP.FAILURE	0.00	0.00
CONCUR.PREF.I/O STREAMS-HWM	0.00	N/A
PREF.I/O STREAMS REDUCTION	0.00	0.00
PARALLEL QUERY REQUESTS	0.00	0.00
PARALL.QUERY REQ.REDUCTION	0.00	0.00
PREF.QUANT.REDUCED TO 1/2	0.00	0.00
PREF.QUANT.REDUCED TO 1/4	0.00	0.00

BP1 WRITE OPERATIONS	QUANTITY	/SECOND
BUFFER UPDATES	8674.5K	2892.52
PAGES WRITTEN	1662.7K	554.43
BUFF.UPDATES/PAGES WRITTEN	5.22	
SYNCHRONOUS WRITES	1127.00	0.38
ASYNCHRONOUS WRITES	147.1K	49.06
PAGES WRITTEN PER WRITE I/O	11.22	
HORIZ.DEF.WRITE THRESHOLD	0.00	0.00
VERTI.DEF.WRITE THRESHOLD	1593.00	0.53
DM THRESHOLD	0.00	0.00
WRITE ENGINE NOT AVAILABLE	0.00	0.00
PAGE-INS REQUIRED FOR WRITE	0.00	0.00

BP1 READ OPERATIONS	QUANTITY	/SECOND
BPOOL HIT RATIO (%)	13.07	
GETPAGE REQUEST	7270.9K	2424.49
GETPAGE REQUEST-SEQUENTIAL	3.00	0.00
GETPAGE REQUEST-RANDOM	7270.9K	2424.49
SYNCHRONOUS READS	624.5K	208.24
SYNCHRON. READS-SEQUENTIAL	1.00	0.00
SYNCHRON. READS-RANDOM	624.5K	208.24
GETPAGE PER SYN.READ-RANDOM	11.64	
SEQUENTIAL PREFETCH REQUEST	2.00	0.00
SEQUENTIAL PREFETCH READS	2.00	0.00
PAGES READ VIA SEQ.PREFETCH	61.00	0.02
S.PRF.PAGES READ/S.PRF.READ	30.50	
LIST PREFETCH REQUESTS	0.00	0.00
LIST PREFETCH READS	0.00	0.00
PAGES READ VIA LIST PREFTCH	0.00	0.00
L.PRF.PAGES READ/L.PRF.READ	N/C	
DYNAMIC PREFETCH REQUESTED	361.3K	120.47
DYNAMIC PREFETCH READS	233.3K	77.80
PAGES READ VIA DYN.PREFETCH	5696.0K	1899.35
D.PRF.PAGES READ/D.PRF.READ	24.41	
PREF.DISABLED-NO BUFFER	0.00	0.00
PREF.DISABLED-NO READ ENG	0.00	0.00
PAGE-INS REQUIRED FOR READ	1270.00	0.40

Paging indicates that the overall system memory requirements are too high. Paging impacts the entire LPAR, not only DB2.

There are Synch writes, but the DM threshold has not been reached, so this isn't a problem.

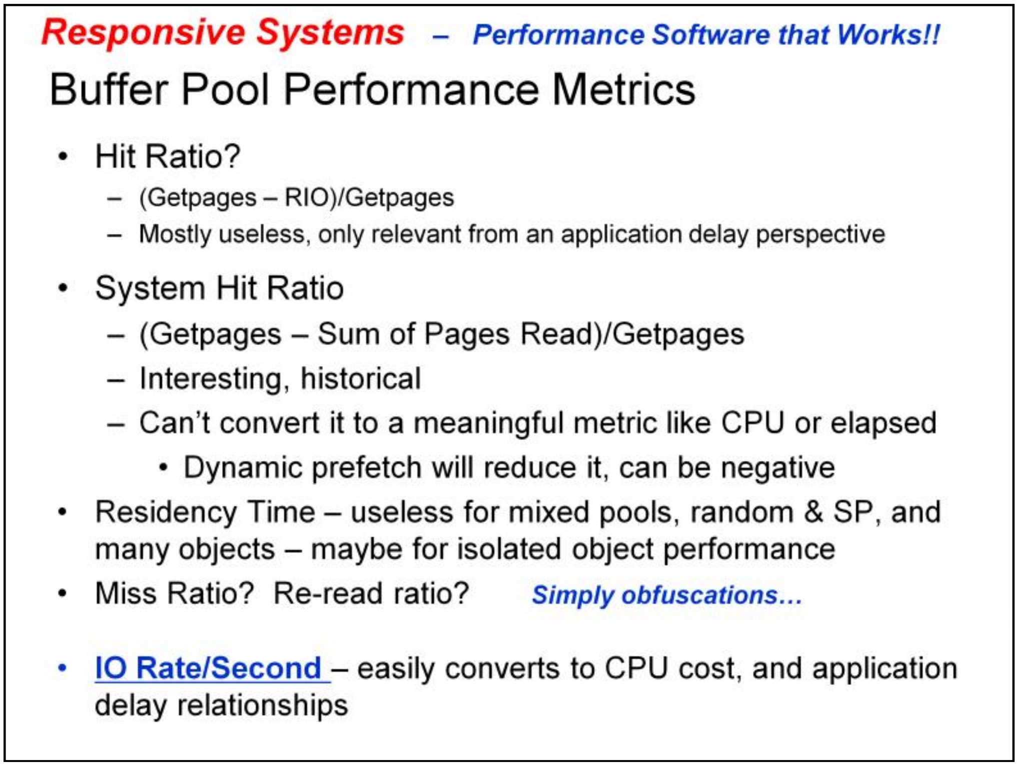
Frequently Mis-Understood

Current Buffers Active

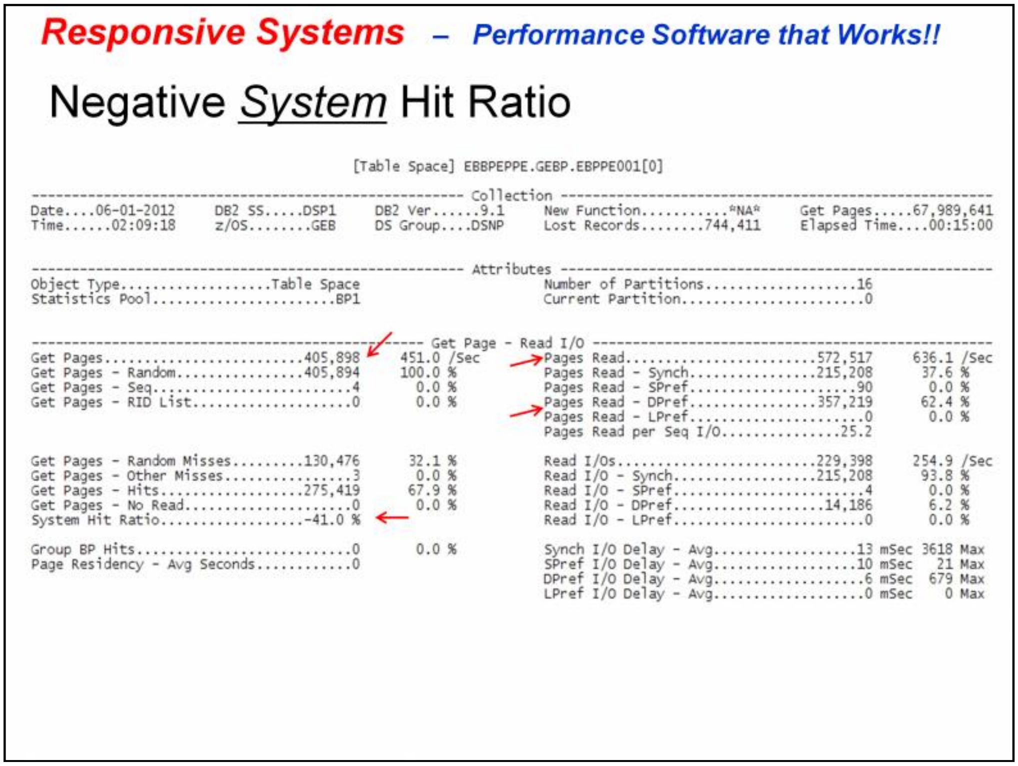
I have 150,000 buffers in the pool, but I'm only using 1763 of them.

Should I make the pool smaller?

The naming of this field is misleading. It is the number of buffers that are NOT available – pages that have been updated but not written yet, and pages on a read or write queue.



It has been proven of dozens of performance studies that the IO rate/second is the only metric that can be converted into measurements that really matter to clients – CPU costs, and specific application delay factors.

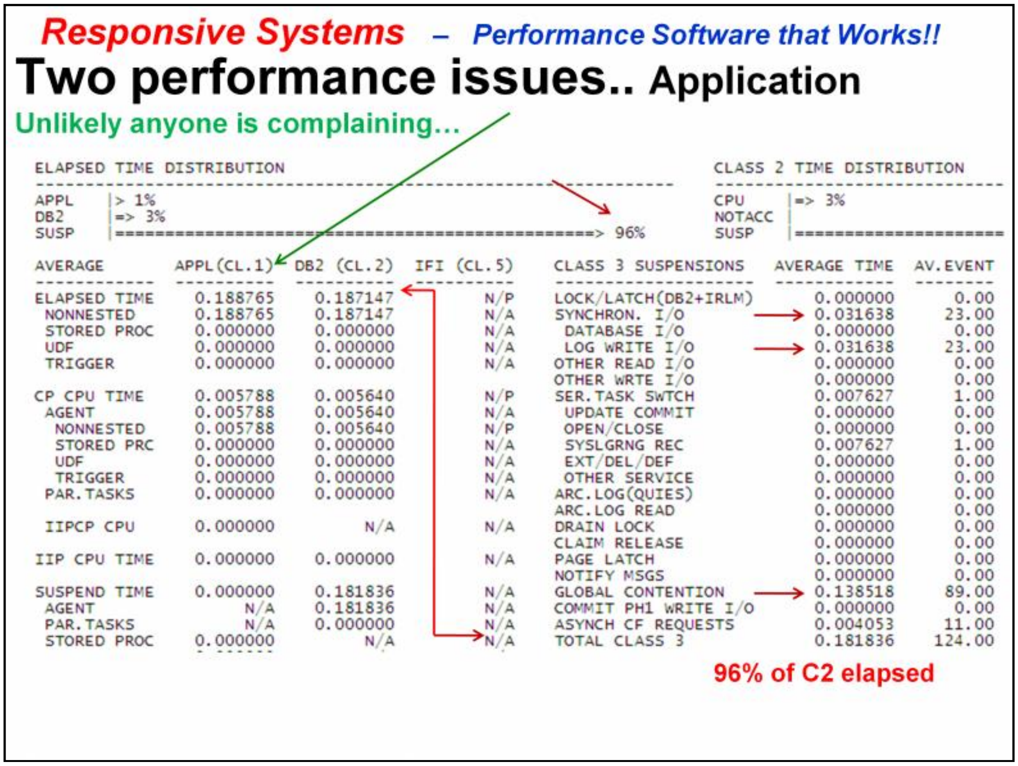


Dynamic prefetch is reading in thousands of pages that are never accessed by the application.

i.e. no getpage request issued for them.

So we see negative hit ratios at object levels and, of course, at the overall pool level.

Pool ratio can still be positive, although some objects are reducing the overall pool hit ration.




```
ELAPSED TIME DISTRIBUTION                                        CLASS 2 TIME DISTRIBUTION
----------------------------------------------------------------  --------------------------
APPL  |> 1%                                                        CPU    |=> 3%
DB2   |=> 3%                                                       NOTACC |
SUSP  |=================================================> 96%      SUSP   |====================

AVERAGE        APPL(CL.1)  DB2 (CL.2)  IFI (CL.5)   CLASS 3 SUSPENSIONS    AVERAGE TIME  AV.EVENT
------------   ----------  ----------  ----------   --------------------   ------------  --------
ELAPSED TIME     0.188765    0.187147         N/P   LOCK/LATCH(DB2+IRLM)       0.000000      0.00
 NONNESTED       0.188765    0.187147         N/A   SYNCHRON. I/O              0.031638     23.00
 STORED PROC     0.000000    0.000000         N/A    DATABASE I/O              0.000000      0.00
 UDF             0.000000    0.000000         N/A    LOG WRITE I/O             0.031638     23.00
 TRIGGER         0.000000    0.000000         N/A   OTHER READ I/O             0.000000      0.00
                                                    OTHER WRTE I/O             0.000000      0.00
CP CPU TIME      0.005788    0.005640         N/P   SER.TASK SWTCH             0.007627      1.00
 AGENT           0.005788    0.005640         N/A    UPDATE COMMIT             0.000000      0.00
  NONNESTED      0.005788    0.005640         N/P    OPEN/CLOSE                0.000000      0.00
  STORED PRC     0.000000    0.000000         N/A    SYSLGRNG REC              0.007627      1.00
  UDF            0.000000    0.000000         N/A    EXT/DEL/DEF               0.000000      0.00
  TRIGGER        0.000000    0.000000         N/A    OTHER SERVICE             0.000000      0.00
 PAR.TASKS       0.000000    0.000000         N/A   ARC.LOG(QUIES)             0.000000      0.00
                                                    ARC.LOG READ               0.000000      0.00
 IIPCP CPU       0.000000         N/A         N/A   DRAIN LOCK                 0.000000      0.00
                                                    CLAIM RELEASE              0.000000      0.00
IIP CPU TIME     0.000000    0.000000         N/A   PAGE LATCH                 0.000000      0.00
                                                    NOTIFY MSGS                0.000000      0.00
SUSPEND TIME     0.000000    0.181836         N/A   GLOBAL CONTENTION          0.138518     89.00
 AGENT                N/A    0.181836         N/A   COMMIT PH1 WRITE I/O       0.000000      0.00
 PAR.TASKS            N/A    0.000000         N/A   ASYNCH CF REQUESTS         0.004053     11.00
 STORED PROC     0.000000         N/A         N/A   TOTAL CLASS 3              0.181836    124.00
```

Logging delays, and cross-system contention with another member in the data sharing group.

While the elapsed time is good, 96% is wait time.   This is one of little item often overlooked during performance analysis.

Because the elapsed time is good, nobody looks for the delay problems.

Synch IO avg isn't bad, but on the high side for todays IO subsystems.

Responsive Systems – ***Performance Software that Works!!***

Performance problem, emailed

Yesterday, we had a major incident in Production when we adjusted the Bufferpool sizes. At first, all went well. We resized one BP, waited for 15 minutes, no problem, resized the next and so on. After the fifth, all of a sudden, the whole system went nearly dead. Only with a high privileged user, we could log-on and undo the sizing. As soon as one of the Bufferpools was at its original size, within a few seconds, the system recovered and all works fine since then.

We detected no paging activity, just CPU at 100%, which means that more than 3000 MIPS were in use while we usually have peaks up to 2200 MIPS and an average usage of much less. We have assigned 42 GB of RAM to the LPAR, only 50% of it is really used and the resizing of the Bufferpools should have been from about 10 GB to 15 GB.

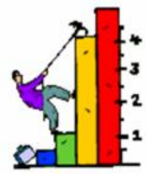
I had no idea why DB2 could be able to use 6 processors with highest priority at 100%......

So, DB2 just ate your system. However, with the memory usage mentioned, and *NO* paging activity, it's unlikely that the pool increases caused the problem.

The fact that it went away after pools were decreased, is probably incidental luck.

Performance problem, emailed

- We asked some basic questions, and asked for a DB2 Statistics report
- The next data was an RMF report
  - No paging activity, what-so-ever, nothing
  - All 6 engines very busy
  - Several devices with heavy I/O volume
    - Not paging or swap volumes



- *Took them a week* to get and send a DB2 Statistics report
  - So how important was problem resolutuon?
    - We see this same issue weekly on DB2L….

The only time I ever saw DB2 completely eat a machine was during a huge backout, and that was more than a decade ago..

Responsive Systems – *Performance Software that Works!!*

Performance problem, emailed

SUBSYSTEM SERVICES	QUANTITY
IDENTIFY	6189.00
CREATE THREAD	1464.6K
SIGNON	1455.4K
TERMINATE	1473.9K
ROLLBACK	23038.00
COMMIT PHASE 1	1452.5K
COMMIT PHASE 2	322.1K
READ ONLY COMMIT	1232.0K
UNITS OF RECOVERY INDOUBT	0.00
UNITS OF REC.INDBT RESOLVED	0.00
SYNCHS(SINGLE PHASE COMMIT)	367.4K
QUEUED AT CREATE THREAD	0.00
SUBSYSTEM ALLIED MEMORY EOT	2251.00
SUBSYSTEM ALLIED MEMORY EOM	0.00
SYSTEM EVENT CHECKPOINT	381.00
HIGH WATER MARK IDBACK	73.00
HIGH WATER MARK IDFORE	28.00
HIGH WATER MARK CTHREAD	130.00

LOG ACTIVITY	QUANTITY	/SECOND
READS SATISFIED-OUTPUT BUFF	67691.3K	817.53
READS SATISFIED-OUTP.BUF(%)	38.30	
READS SATISFIED-ACTIVE LOG	109.1M	1317.22
READS SATISFIED-ACTV.LOG(%)	61.70	
READS SATISFIED-ARCHIVE LOG	0.00	0.00
READS SATISFIED-ARCH.LOG(%)	0.00	
TAPE VOLUME CONTENTION WAIT	0.00	0.00
READ DELAYED-UNAVAIL.RESOUR	0.00	0.00
ARCHIVE LOG READ ALLOCATION	0.00	0.00
ARCHIVE LOG WRITE ALLOCAT.	304.00	0.00
CONTR.INTERV.OFFLOADED-ARCH	9138.2K	110.37
LOOK-AHEAD MOUNT ATTEMPTED	0.00	0.00
LOOK-AHEAD MOUNT SUCCESSFUL	0.00	0.00
UNAVAILABLE OUTPUT LOG BUFF	11.00	0.00
OUTPUT LOG BUFFER PAGED IN	0.00	0.00
LOG RECORDS CREATED	156.7M	1892.84
LOG CI CREATED	9143.0K	110.42
LOG WRITE I/O REQ (LOG1&2)	4285.9K	51.76
LOG CI WRITTEN (LOG1&2)	20273.8K	244.85
LOG RATE FOR 1 LOG (MB)	N/A	0.48
LOG WRITE SUSPENDED	1321.9K	15.96

What happens during a backout?

Synch IO backout, logging increases 50%.., checkpoints, filling logs, archiving, etc

A lot of rollbacks, and a huge number of reads from the Active Log. The read rate for the active logs is 1317/Second. That's a very high I/O rate. Note the number of checkpoints, archive log allocations and Cis

Responsive Systems – *Performance Software that Works!!*

Buffer Pool Part1

BP1 GENERAL	QUANTITY	/SECOND	/THREAD	/COMMIT
CURRENT ACTIVE BUFFERS	1037.85	N/A	N/A	N/A
UNAVAIL.BUFFER-VPOOL FULL	0.00	0.00	0.00	0.00
NUMBER OF DATASET OPENS	3217.00	0.04	0.00	0.00
BUFFERS ALLOCATED - VPOOL	232.3K	N/A	N/A	N/A
DFHSM MIGRATED DATASET	0.00	0.00	0.00	0.00
DFHSM RECALL TIMEOUTS	0.00	0.00	0.00	0.00
VPOOL EXPANS. OR CONTRACT.	2.00	0.00	0.00	0.00
VPOOL OR HPOOL EXP.FAILURE	0.00	0.00	0.00	0.00
CONCUR.PREF.I/O STREAMS-HWM	0.00	N/A	N/A	N/A
PREF.I/O STREAMS REDUCTION	0.00	0.00	0.00	0.00
PARALLEL QUERY REQUESTS	0.00	0.00	0.00	0.00
PARALL.QUERY REQ.REDUCTION	0.00	0.00	0.00	0.00
PREF.QUANT.REDUCED TO 1/2	0.00	0.00	0.00	0.00
PREF.QUANT.REDUCED TO 1/4	0.00	0.00	0.00	0.00

BP1 READ OPERATIONS	QUANTITY	/SECOND
BPOOL HIT RATIO (%)	53.81	
GETPAGE REQUEST	447.0M	5398.69
GETPAGE REQUEST-SEQUENTIAL	110.8M	1337.65
GETPAGE REQUEST-RANDOM	336.3M	4061.04
SYNCHRONOUS READS	20658.6K	249.50
SYNCHRON. READS-SEQUENTIAL	4244.1K	51.26
SYNCHRON. READS-RANDOM	16414.5K	198.24
GETPAGE PER SYN.READ-RANDOM	20.49	
SEQUENTIAL PREFETCH REQUEST	2438.7K	29.45
SEQUENTIAL PREFETCH READS	1492.0K	18.02
PAGES READ VIA SEQ.PREFETCH	46269.7K	558.81
S.PRF.PAGES READ/S.PRF.READ	31.01	
LIST PREFETCH REQUESTS	2793.2K	33.73
LIST PREFETCH READS	290.5K	3.51
PAGES READ VIA LIST PREFTCH	5613.4K	67.79
L.PRF.PAGES READ/L.PRF.READ	19.33	
DYNAMIC PREFETCH REQUESTED	20125.4K	243.06
DYNAMIC PREFETCH READS	8848.1K	106.86
PAGES READ VIA DYN.PREFETCH	133.9M	1617.50
D.PRF.PAGES READ/D.PRF.READ	15.14	
PREF.DISABLED-NO BUFFER	0.00	0.00
PREF.DISABLED-NO READ ENG	25475.00	0.31
PAGE-INS REQUIRED FOR READ	0.00	0.00

What are the problems in this data? What happens when you hit this threshold?

Dataset opens, Prefetch Disabled-No Read Engine.

Responsive Systems – *Performance Software that Works!!*

Buffer Pool Part2

BP1 WRITE OPERATIONS	QUANTITY	/SECOND
BUFFER UPDATES	57165.1K	690.40
PAGES WRITTEN	7549.6K	91.18
BUFF.UPDATES/PAGES WRITTEN	7.57	
SYNCHRONOUS WRITES	17486.00	0.21
ASYNCHRONOUS WRITES	2356.3K	28.46
PAGES WRITTEN PER WRITE I/O	3.18	
HORIZ.DEF.WRITE THRESHOLD	25819.00	0.31
VERTI.DEF.WRITE THRESHOLD	5658.00	0.07
DM THRESHOLD	0.00	0.00
WRITE ENGINE NOT AVAILABLE	47.00	0.00
PAGE-INS REQUIRED FOR WRITE	0.00	0.00

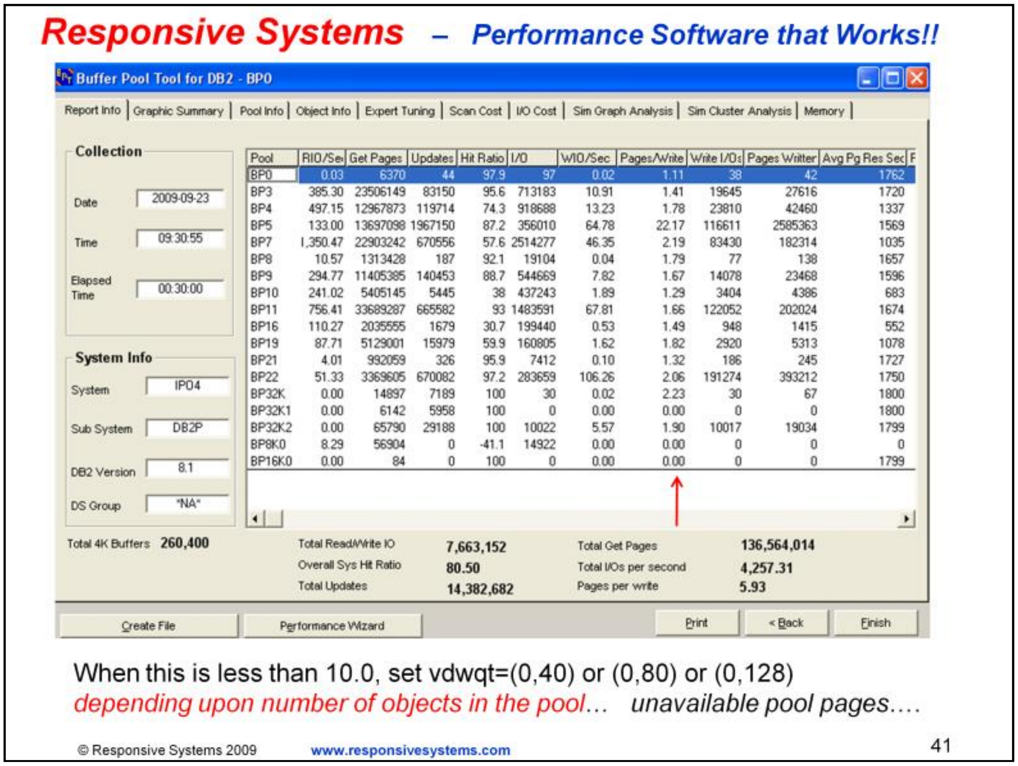
Low, but normal…

Reverse this relationship

Fix this problem…

Probable cause of the problems here?

Write Engine Not Available. DWQT is high, VDWQT is low – we should see the opposite. Pages/Write is low single digit, set VDWQT=(0,140)



Pool	RIO/Se	Get Pages	Updates	Hit Ratio	I/O	WIO/Sec	Pages/Write	Write I/Os	Pages Writter	Avg Pg Res Sec
BP0	0.03	6370	44	97.9	97	0.02	1.11	38	42	1762
BP3	385.30	23506149	83150	95.6	713183	10.91	1.41	19645	27616	1720
BP4	497.15	12967873	119714	74.3	918688	13.23	1.78	23810	42460	1337
BP5	133.00	13697098	1967150	87.2	356010	64.78	22.17	116611	2585363	1569
BP7	1,350.47	22903242	670556	57.6	2514277	46.35	2.19	83430	182314	1035
BP8	10.57	1313428	187	92.1	19104	0.04	1.79	77	138	1657
BP9	294.77	11405385	140453	88.7	544669	7.82	1.67	14078	23468	1596
BP10	241.02	5405145	5445	38	437243	1.89	1.29	3404	4386	683
BP11	756.41	33689287	665582	93	1483591	67.81	1.66	122052	202024	1674
BP16	110.27	2035555	1679	30.7	199440	0.53	1.49	948	1415	552
BP19	87.71	5129001	15979	59.9	160805	1.62	1.82	2920	5313	1078
BP21	4.01	992059	326	95.9	7412	0.10	1.32	186	245	1727
BP22	51.33	3369605	670082	97.2	283659	106.26	2.06	191274	393212	1750
BP32K	0.00	14897	7189	100	30	0.02	2.23	30	67	1800
BP32K1	0.00	6142	5958	100	0	0.00	0.00	0	0	1800
BP32K2	0.00	65790	29188	100	10022	5.57	1.90	10017	19034	1799
BP8K0	8.29	56904	0	-41.1	14922	0.00	0.00	0	0	0
BP16K0	0.00	84	0	100	0	0.00	0.00	0	0	1799

When a pool has a lot of objects, hundreds or thousands, lower is better. Think about the number of pages that might be updated in the pool, across all the objects, and have to be written at checkpoint.

Responsive Systems – Performance Software that Works!!

Buffer Pool DB2 V9

BP20 GENERAL	QUANTITY	/SECOND	/THREAD	/COMMIT
CURRENT ACTIVE BUFFERS	100.44	N/A	N/A	N/A
UNAVAIL.BUFFER-VPOOL FULL	0.00	0.00	0.00	0.00
NUMBER OF DATASET OPENS	19795.00	0.23	0.01	0.00
BUFFERS ALLOCATED - VPOOL	90000.00	N/A	N/A	N/A
DFHSM MIGRATED DATASET	0.00	0.00	0.00	0.00
DFHSM RECALL TIMEOUTS	0.00	0.00	0.00	0.00
VPOOL EXPANS. OR CONTRACT.	0.00	0.00	0.00	0.00
VPOOL OR HPOOL EXP.FAILURE	0.00	0.00	0.00	0.00
CONCUR.PREF.I/O STREAMS-HWM	251.00	N/A	N/A	N/A
PREF.I/O STREAMS REDUCTION	0.00	0.00	0.00	0.00
PARALLEL QUERY REQUESTS	88.00	0.00	0.00	0.00
PARALL.QUERY REQ.REDUCTION	0.00	0.00	0.00	0.00
PREF.QUANT.REDUCED TO 1/2	0.00	0.00	0.00	0.00
PREF.QUANT.REDUCED TO 1/4	0.00	0.00	0.00	0.00

BP20 READ OPERATIONS	QUANTITY	/SECOND
BPOOL HIT RATIO (%)	59.14	
GETPAGE REQUEST	234.1M	2715.21
GETPAGE REQUEST-SEQUENTIAL	104.6M	1213.33
GETPAGE REQUEST-RANDOM	129.5M	1501.88
SYNCHRONOUS READS	10878.4K	126.17
SYNCHRON. READS-SEQUENTIAL	29762.00	0.35
SYNCHRON. READS-RANDOM	10848.6K	125.82
GETPAGE PER SYN.READ-RANDOM	11.94	
SEQUENTIAL PREFETCH REQUEST	842.0K	9.77
SEQUENTIAL PREFETCH READS	788.9K	9.15
PAGES READ VIA SEQ.PREFETCH	50206.6K	582.31
S.PRF.PAGES READ/S.PRF.READ	63.64	
LIST PREFETCH REQUESTS	370.6K	4.30
LIST PREFETCH READS	41866.00	0.49
PAGES READ VIA LIST PREFTCH	1276.8K	14.81
L.PRF.PAGES READ/L.PRF.READ	30.50	
DYNAMIC PREFETCH REQUESTED	2916.9K	33.83
DYNAMIC PREFETCH READS	1139.7K	13.22
PAGES READ VIA DYN.PREFETCH	33286.0K	386.06
D.PRF.PAGES READ/D.PRF.READ	29.21	
PREF.DISABLED-NO BUFFER	0.00	0.00
PREF.DISABLED-NO READ ENG	0.00	0.00
PAGE-INS REQUIRED FOR READ	0.00	0.00

What important piece of information is missing ??

Note Pages for Prefetch read…

Time -- elapsed time for the statistics data !!!

Responsive Systems – *Performance Software that Works!!*

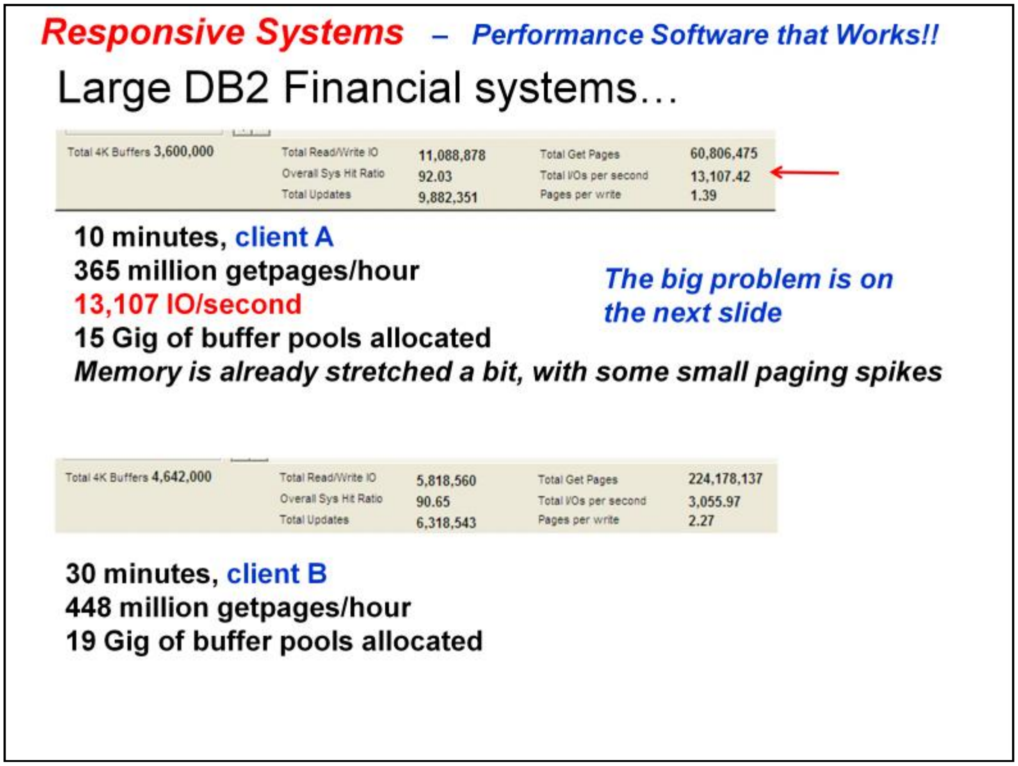
Buffer Pool DB2 V9

BP20 WRITE OPERATIONS	QUANTITY	/SECOND
BUFFER UPDATES	18540.8K	215.04
PAGES WRITTEN	1014.4K	11.77
BUFF.UPDATES/PAGES WRITTEN	18.28	
SYNCHRONOUS WRITES	8935.00	0.10
ASYNCHRONOUS WRITES	218.9K	2.54
PAGES WRITTEN PER WRITE I/O	4.45	
HORIZ.DEF.WRITE THRESHOLD	227.00	0.00
VERTI.DEF.WRITE THRESHOLD	184.4K	2.14
DM THRESHOLD	0.00	0.00
WRITE ENGINE NOT AVAILABLE	0.00	0.00
PAGE-INS REQUIRED FOR WRITE	0.00	0.00

This is what we like to see as a relationship… mostly VDWQT triggering

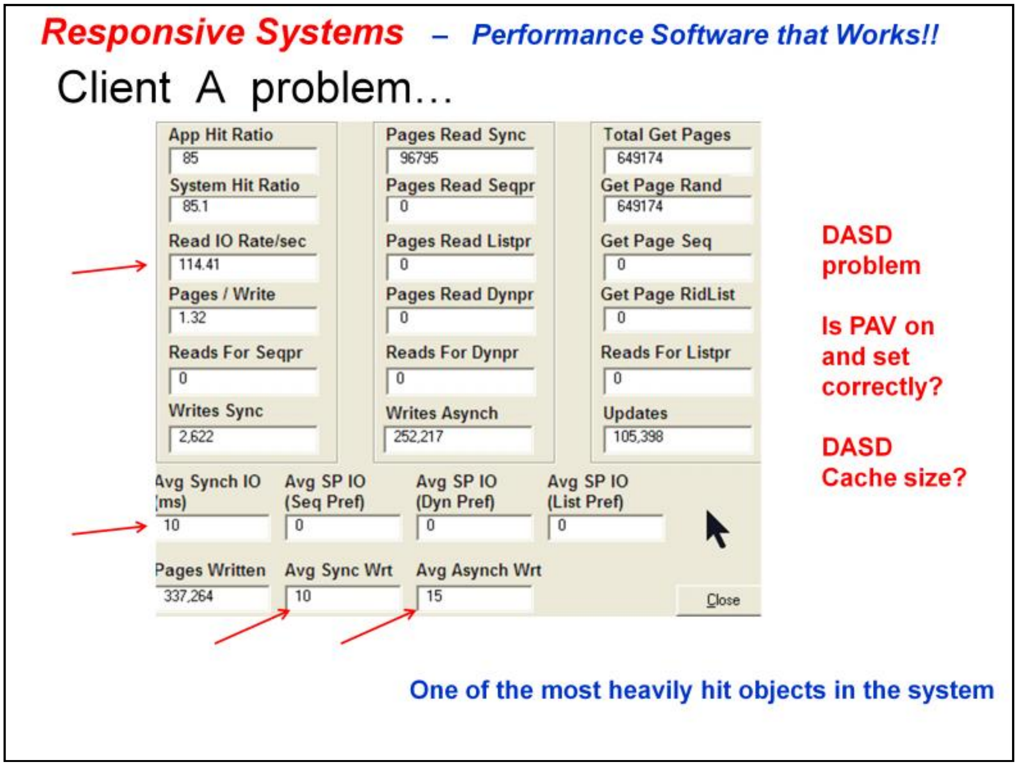
High numbers for VDWQT vs. DWQT

Almost all writes are triggered by VDWQT.



Which sounds bigger – 500 million, or ½ billion?

Systems are constantly getting larger, and have amazing throughput levels.

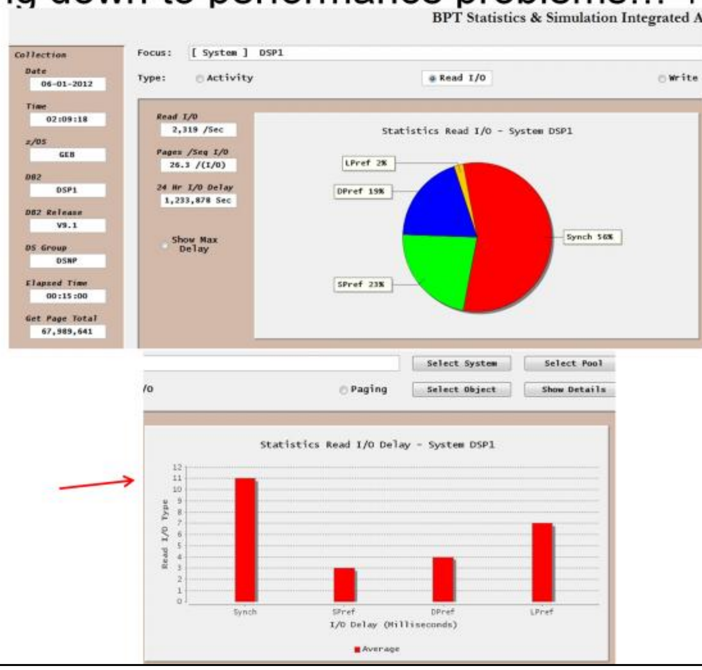


The client's huge performance problem is poor DASD performance.
10 Ms Synch IO times, with a very high IO rate/second
Synch write times are also poor.

DASD cache is severely under sized for the workload.

Responsive Systems – ***Performance Software that Works!!***

Drilling down to performance problems… 1



Responsive Systems – *Performance Software that Works!!*

Drilling down to performance problems… 2

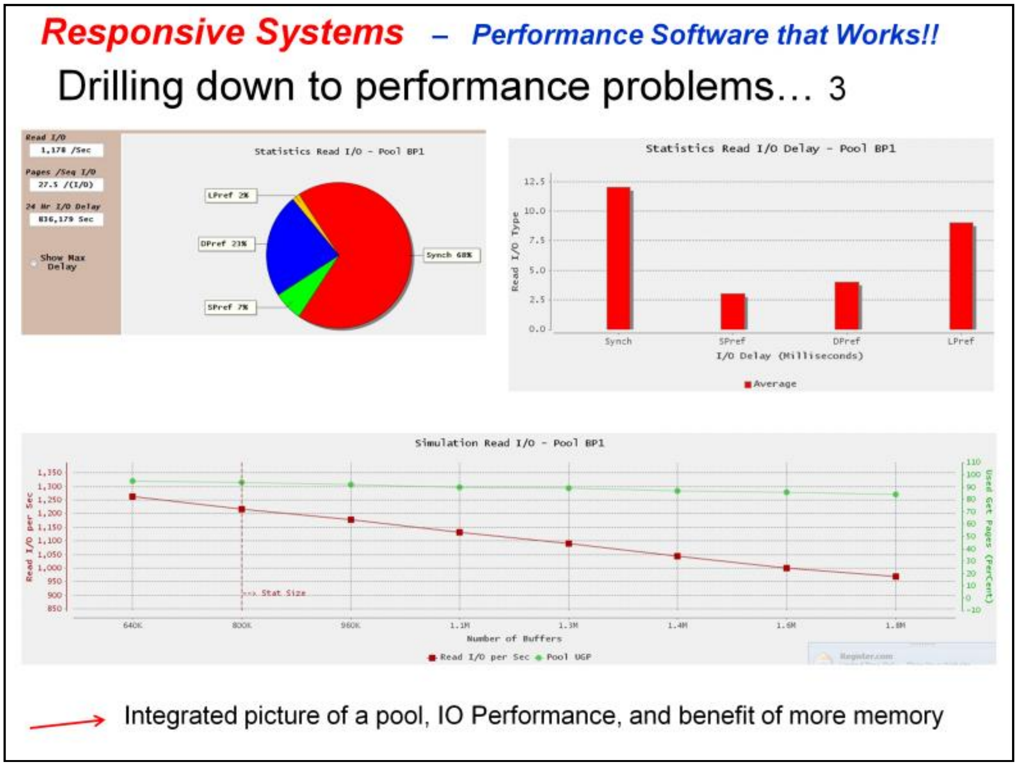
Name	Size	PGFIX	PGSTEAL	VPSEQT	Sim. Reports Valid [NV]	Activity Get Page /Sec	Read I/O I/O /Sec	Read I/O Pages /Seq-I/O
BP0	15,000	No	LRU	25 %	0 [1]	198 /Sec	1 /Sec	12.4 / I/O
BP1	800,000	Yes	LRU	10 %	1 [0]	20,660 /Sec	1,178 /Sec	26.1 / I/O
BP2	900,000	Yes	LRU	20 %	1 [0]	34,649 /Sec	545 /Sec	24.7 / I/O
BP3	300,000	No	LRU	85 %	1 [0]	2,502 /Sec	9 /Sec	7.4 / I/O
BP4	300,000	No	LRU	50 %	0 [1]	833 /Sec	3 /Sec	28.3 / I/O
BP5	500,000	Yes	LRU	40 %	1 [0]	13,299 /Sec	365 /Sec	37.9 / I/O
BP6	800,000	Yes	LRU	20 %	0 [1]	706 /Sec	31 /Sec	26.2 / I/O
BP7	900,000	No	LRU	20 %	0 [1]	548 /Sec	2 /Sec	7.7 / I/O
BP8	10,000	No	LRU	20 %	0 [1]	3 /Sec	1 /Sec	19.2 / I/O
BP9	700,000	No	LRU	65 %	0 [1]	570 /Sec	0 /Sec	28.2 / I/O
BP10	10,000	No	LRU	60 %	0 [1]	0 /Sec	0 /Sec	0.0 / I/O
BP11	10,000	No	LRU	10 %	0 [1]	23 /Sec	0 /Sec	0.0 / I/O
BP8K0	1,000	No	LRU	80 %	0 [1]	3 /Sec	1 /Sec	13.2 / I/O
BP16K0	80,000	No	LRU	10 %	1 [0]	241 /Sec	21 /Sec	3.1 / I/O
BP32K	62,000	No	LRU	40 %	1 [0]	1,295 /Sec	164 /Sec	6.5 / I/O

The client's huge performance problem is poor DASD performance.

10 Ms Synch IO times, with a very high IO rate/second

Synch write times are also poor.

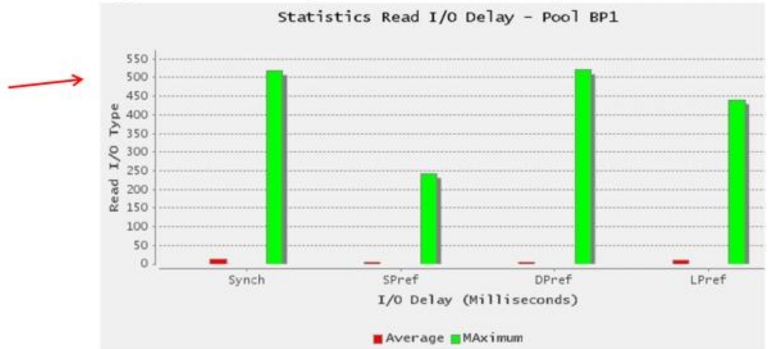
DASD cache is severely under sized for the workload.



Here we can see the reduced IO rate/second if the pool size is increased.

Responsive Systems – Performance Software that Works!!

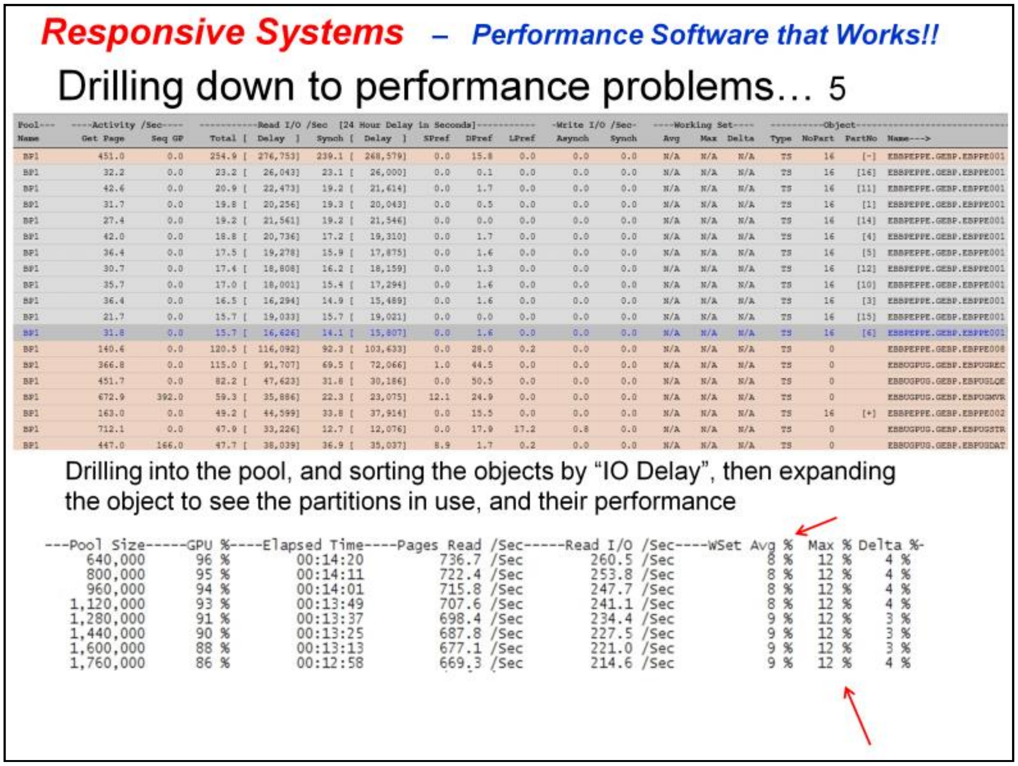
Drilling down to performance problems… 4



Not unusual to see sporadic "huge" IO waits…. As a max value

These huge Max IO waits are not unusual -- I see them in many systems.

You get a dasd controller cache miss, and your data is striped across several spinning disks in a Raid array….



Responsive Systems – *Performance Software that Works!!*

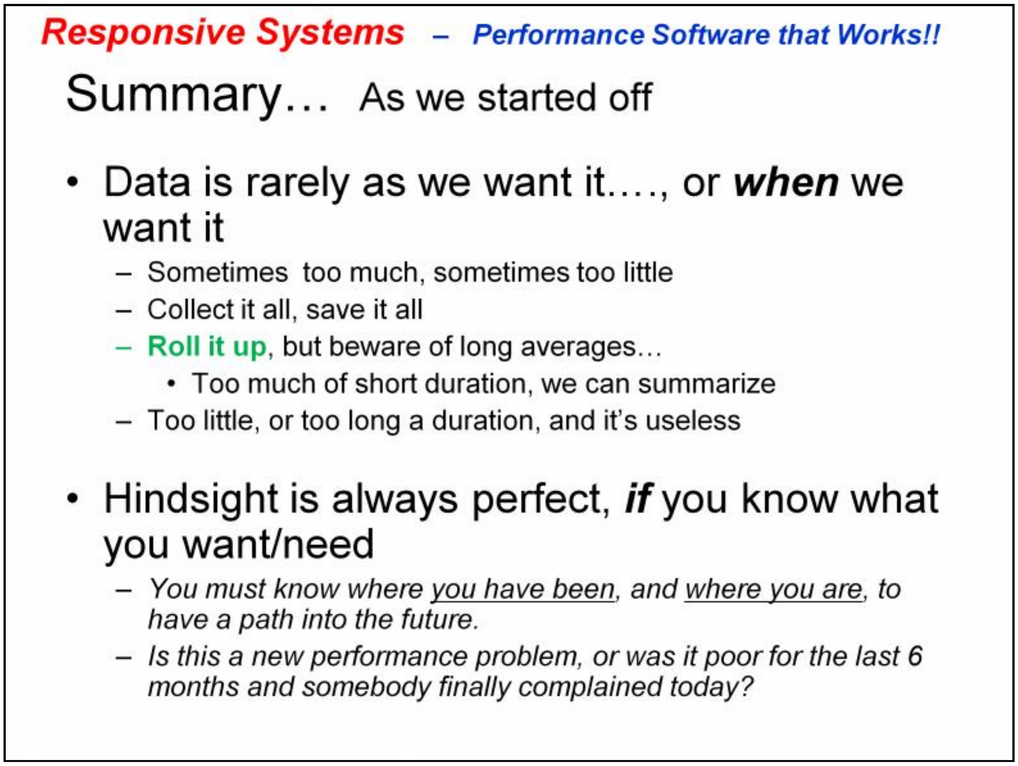
Drilling down to performance problems… 5

Pool--- Name	----Activity /Sec---- Get Page	Seq GP	------------Read I/O /Sec [24 Hour Delay in Seconds]------------ Total [Delay]		Synch [Delay]		SPref	DPref	LPref	-Write I/O /Sec- Asynch	Synch	----Working Set---- Avg	Max	Delta	----------Object------------------------------ Type	NoPart	PartNo	Name--->
BP1	451.0	0.0	254.9 [	276,753]	239.1 [	268,579]	0.0	15.8	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[-]	EBBPEPPE.GEBP.EBPPE001
BP1	32.2	0.0	23.2 [	26,043]	23.1 [	26,000]	0.0	0.1	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[16]	EBBPEPPE.GEBP.EBPPE001
BP1	42.6	0.0	20.9 [	22,473]	19.2 [	21,614]	0.0	1.7	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[11]	EBBPEPPE.GEBP.EBPPE001
BP1	31.7	0.0	19.8 [	20,256]	19.3 [	20,043]	0.0	0.5	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[1]	EBBPEPPE.GEBP.EBPPE001
BP1	27.4	0.0	19.2 [	21,561]	19.2 [	21,546]	0.0	0.0	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[14]	EBBPEPPE.GEBP.EBPPE001
BP1	42.0	0.0	18.8 [	20,736]	17.2 [	19,310]	0.0	1.7	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[4]	EBBPEPPE.GEBP.EBPPE001
BP1	36.4	0.0	17.5 [	19,278]	15.9 [	17,875]	0.0	1.6	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[5]	EBBPEPPE.GEBP.EBPPE001
BP1	30.7	0.0	17.4 [	18,808]	16.2 [	18,159]	0.0	1.3	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[12]	EBBPEPPE.GEBP.EBPPE001
BP1	35.7	0.0	17.0 [	18,001]	15.4 [	17,294]	0.0	1.6	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[10]	EBBPEPPE.GEBP.EBPPE001
BP1	36.4	0.0	16.5 [	16,294]	14.9 [	15,489]	0.0	1.6	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[3]	EBBPEPPE.GEBP.EBPPE001
BP1	21.7	0.0	15.7 [	19,033]	15.7 [	19,021]	0.0	0.0	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[15]	EBBPEPPE.GEBP.EBPPE001
BP1	31.8	0.0	15.7 [	16,626]	14.1 [	15,807]	0.0	1.6	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[6]	EBBPEPPE.GEBP.EBPPE001
BP1	140.6	0.0	120.5 [	116,092]	92.3 [	103,633]	0.0	28.0	0.2	0.0	0.0	N/A	N/A	N/A	TS	0		EBBPEPPE.GEBP.EBPPE008
BP1	366.8	0.0	115.0 [	91,707]	69.5 [	72,066]	1.0	44.5	0.0	0.0	0.0	N/A	N/A	N/A	TS	0		EBBUGPUG.GEBP.EBPUGREC
BP1	451.7	0.0	82.2 [	47,623]	31.8 [	30,186]	0.0	50.5	0.0	0.0	0.0	N/A	N/A	N/A	TS	0		EBBUGPUG.GEBP.EBPUGLQE
BP1	672.9	392.0	59.3 [	35,886]	22.3 [	23,075]	12.1	24.9	0.0	0.0	0.0	N/A	N/A	N/A	TS	0		EBBUGPUG.GEBP.EBPUGMVR
BP1	163.0	0.0	49.2 [	44,599]	33.8 [	37,914]	0.0	15.5	0.0	0.0	0.0	N/A	N/A	N/A	TS	16	[+]	EBBPEPPE.GEBP.EBPPE002
BP1	712.1	0.0	47.9 [	33,226]	12.7 [	12,076]	0.0	17.9	17.2	0.8	0.0	N/A	N/A	N/A	TS	0		EBBUGPUG.GEBP.EBPUGSTR
BP1	447.0	166.0	47.7 [	38,039]	36.9 [	35,037]	8.9	1.7	0.2	0.0	0.0	N/A	N/A	N/A	TS	0		EBBUGPUG.GEBP.EBPUGDAT

Drilling into the pool, and sorting the objects by "IO Delay", then expanding the object to see the partitions in use, and their performance

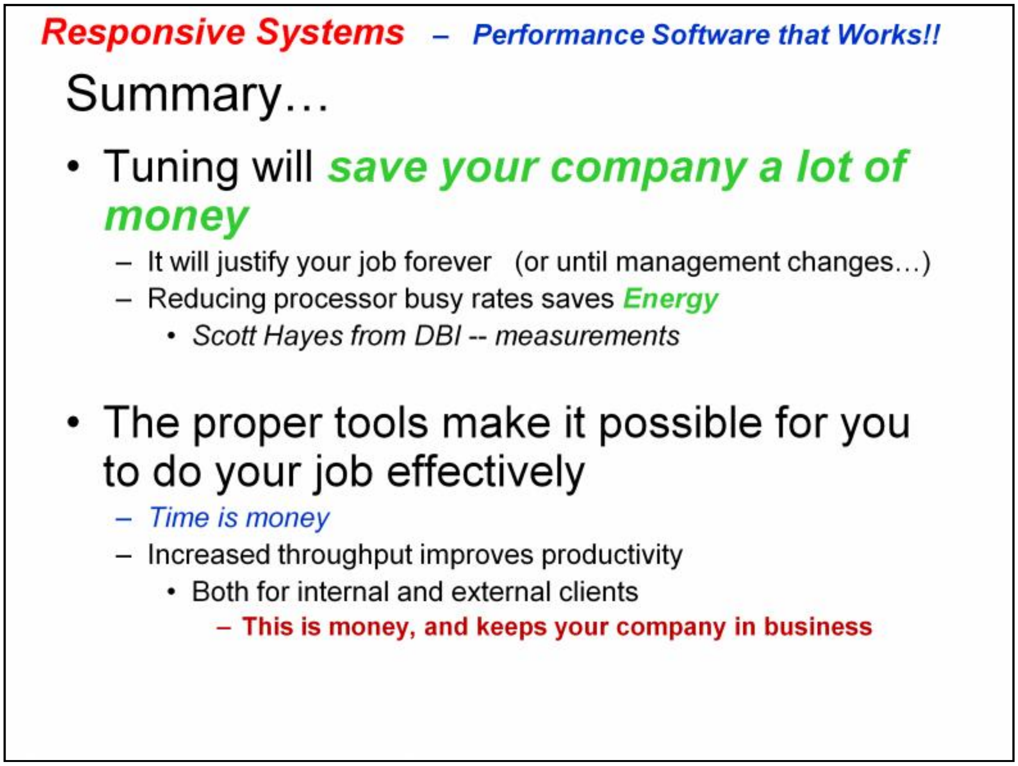
```
---Pool Size-----GPU %----Elapsed Time----Pages Read /Sec-----Read I/O /Sec----WSet Avg %  Max % Delta %-
     640,000       96 %       00:14:20          736.7 /Sec         260.5 /Sec              8 %   12 %    4 %
     800,000       95 %       00:14:11          722.4 /Sec         253.8 /Sec              8 %   12 %    4 %
     960,000       94 %       00:14:01          715.8 /Sec         247.7 /Sec              8 %   12 %    4 %
   1,120,000       93 %       00:13:49          707.6 /Sec         241.1 /Sec              8 %   12 %    4 %
   1,280,000       91 %       00:13:37          698.4 /Sec         234.4 /Sec              9 %   12 %    3 %
   1,440,000       90 %       00:13:25          687.8 /Sec         227.5 /Sec              9 %   12 %    3 %
   1,600,000       88 %       00:13:13          677.1 /Sec         221.0 /Sec              9 %   12 %    3 %
   1,760,000       86 %       00:12:58          669.3 /Sec         214.6 /Sec              9 %   12 %    4 %
```

Working set size represented as a % of the pool size – so it grows as the pool increases, thus the IO rate/sec decreases



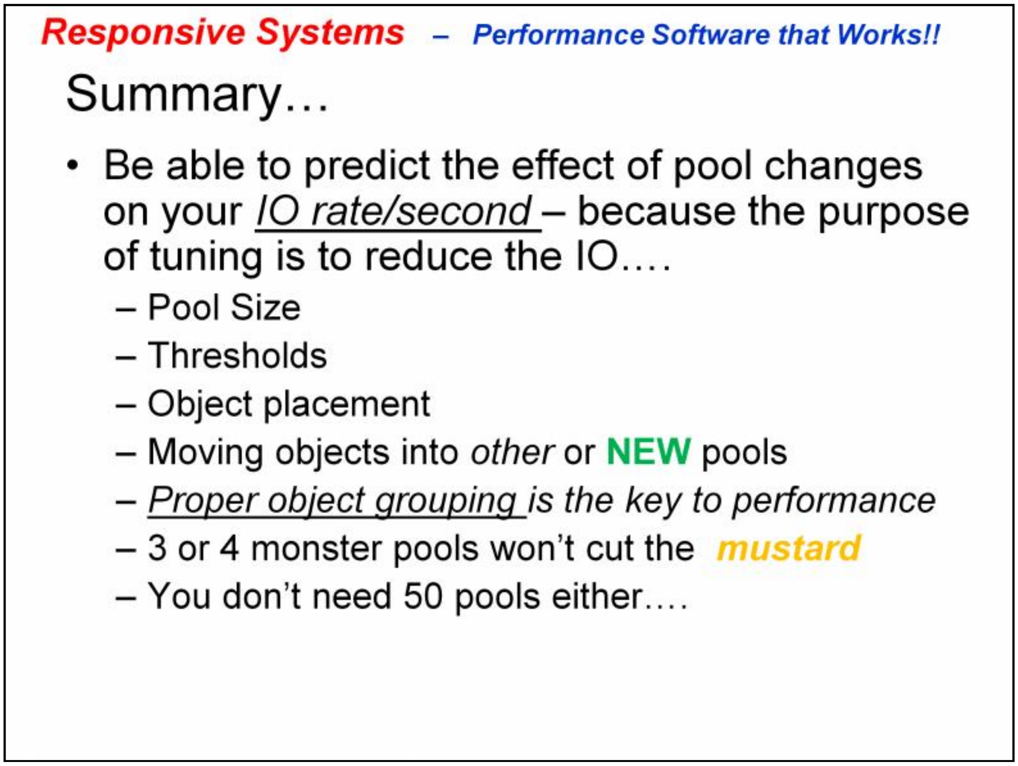
We need the ability, and *tools* get data, and to slice/dice data many different ways. There are never enough tools.

HISTORY data…. Shows you where you were…

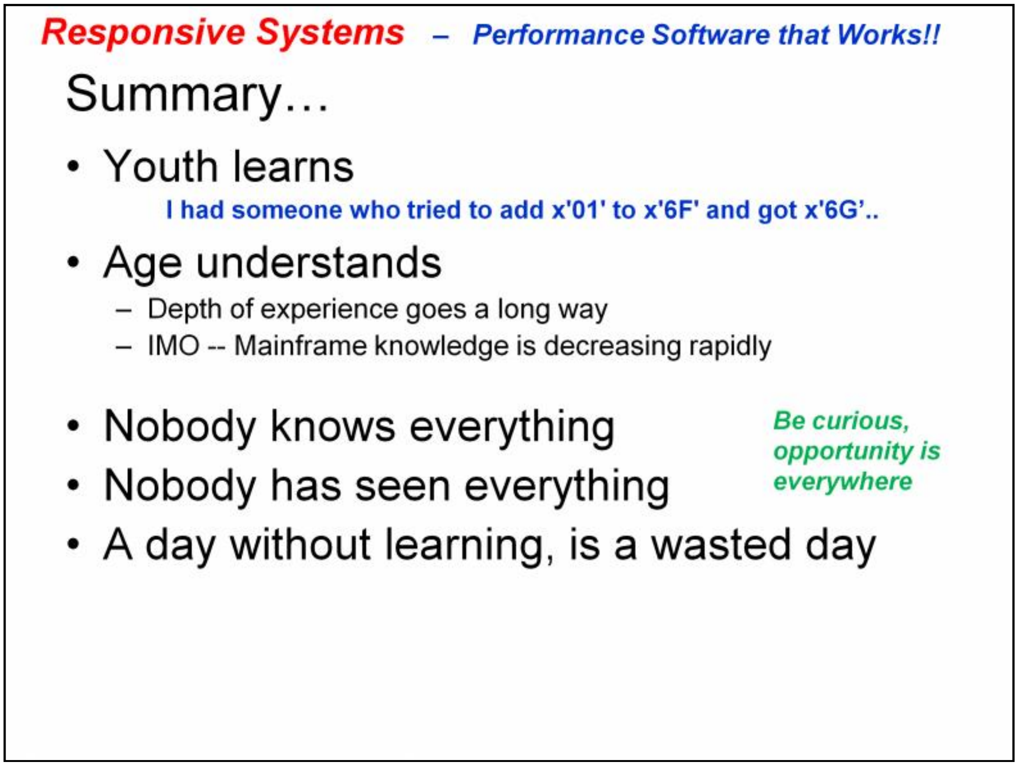


You must know which tools you need, which tools are available – and the different pieces of performance perspective available from each tool.

Some tools may be redundant with others, may not add much, and consume CPU cycles for little benefit.



You must know which tools you need, which tools are available – and the different pieces of performance perspective available from each tool.



Finding something new, the cause of a problem, that sudden flash of insight – is a wonderful thing!!

Responsive Systems – ***Performance Software that Works!!***

I WENT TO A BOOKSTORE AND ASKED THE SALESWOMAN, "WHERE'S THE SELF- HELP SECTION?"

SHE SAID IF SHE TOLD ME, IT WOULD DEFEAT THE PURPOSE.

Thank you for coming

joel@responsivesystems.com
www.responsivesystems.com
732-972-1261

Responsive Systems – ***Performance Software that Works!!***

Joel Goldstein
Responsive Systems
joel@responsivesystems.com

Thank you for attending today !

The Performance Wizard is here, he lives in the Buffer Pool Tool !!!