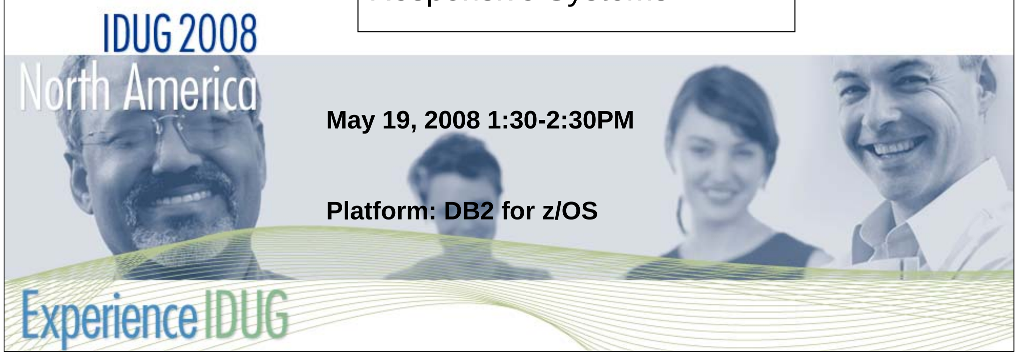
Session: A03

DB2 System Performance, the Basics...

...and a Bit, to a Lot More

Joel Goldstein
Responsive Systems



Abstract

This presentation addresses the approaches, values, thresholds, triggers, and relationships of performance data analysis. Performance monitoring, analysis, and tuning should be a pro-active and continuous process; however, this often gets neglected because people are too busy with other work, or they are not sure what data is really important and which data quickly shows that there are problems - when nobody is complaining. Some basic thresholds and data provide quick insights about problems both at the application level, and the system level. Charts and graphs illustrate the surprising CPU and dollar costs of sequential prefetch and IOs.

This presentation *does <u>not</u> address SQL coding or application design considerations.*

Major Bullet Points

- Sources of data
- Key performance thresholds and indicators
  - Do you have problems?
- Rules of Thumb
  - Good numbers, bad numbers, or ???
  - Can't dot every ***i*** or cross every ***t*** in this presentation….
- Using and applying data for analysis
- Performance examples from the real world

2

<u>Outline:</u>

1. DB2 Data Sources

- **a. Application Accounting Data**
- **b. System Statistics Data**
- **c. DB2 performance trace data**

2. The important performance variables and indicators:

- **a. Variables and relationships**
- **b. Inter-relationships between Statistics and Application data**
- **c. How do you know when you have a problem, and just how bad is it?**

3. Top down tuning approach

a. Tuning from available DB2 data sources

- **b. How much can you tune - the payback potential**
- **c. How long will it take?**

5. Bottom up approach

- **a. Data interpretation**
- **b. Determining where the benefits are…**

6. Summary, Guidelines and recommendations

We care about performance because..

- Competitiveness in the marketplace
- Providing service to your customers
  - Both internal & external
- Reducing costs/avoiding costs
  - Tune, or buy hardware
    - Tuning is a free payback for the money already invested
- Time is money
  - The system is slow today... *I can fix that…*

Or maybe you're just a computer geek like me..

3

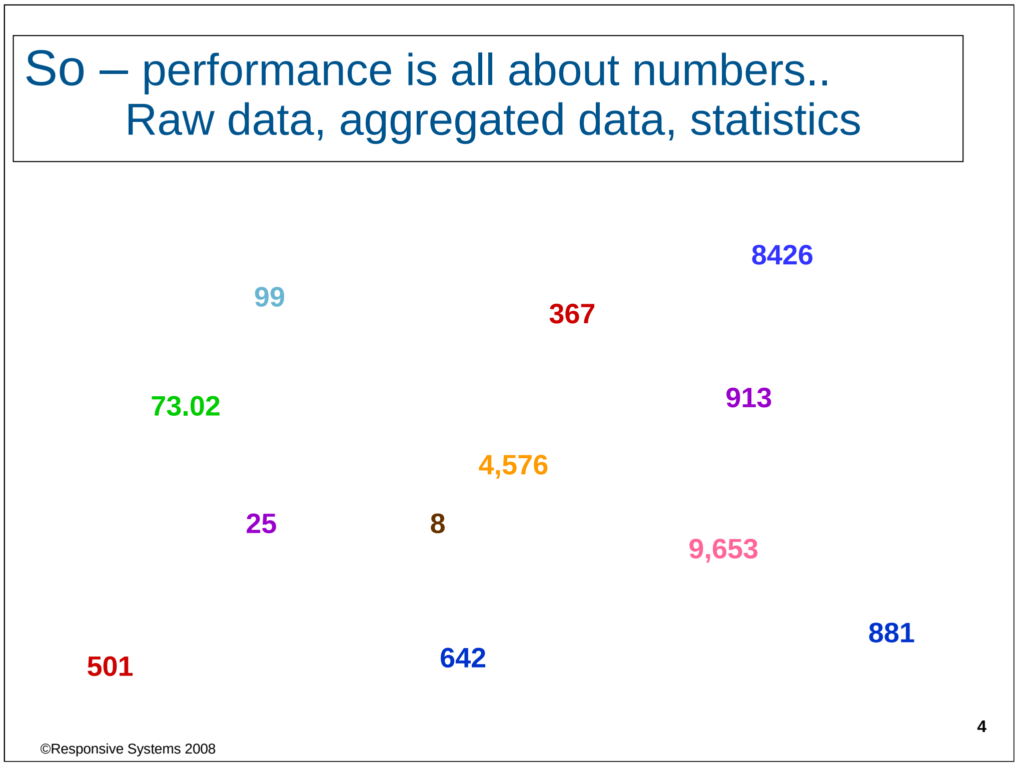
In today's competitive economic environment we can't be complacent about performance. Information is critical to the success of your company, and the ability to both retain existing customers and get new ones, may determine long term growth. The enterprises that can deliver information quickly will grow and thrive - those that cannot, may either cease to exist or will be swallowed by other more aggressive corporations.

Ever increasing processor capacities are often driven more by application in-efficiencies and other performance related problems than by true volume demands. Throwing multi-million dollar hardware solutions at performance problems has become the norm.

If corporate stockholders became aware of the magnitude of corporate waste and in-efficiencies, many corporate officers might find themselves on the receiving end of legal actions - and many boards of directors would be replaced.

Some reasonable amounts of system and application tuning can provide dramatic paybacks at many companies. The multi-million dollar tuning success stories exist - and should receive more press than the large system failures.

If management could only become truly aware of the opportunities, and reward staff for improving performance - they would be amazed at the results and long term savings.



Numbers, numbers everywhere. What do they all mean? What is important?

Mean and Median

- How many of you remember the difference?

Mean is an average, and averages are easy – right?…

Mean is an average, and averages are easy – misleading...

4, 8, 12, 2, 10, 8, 104, 4, 6, 10 = 168/10 = 16.8

4, 8, 12, ~~2~~, 10, 8, ~~104~~, 4, 6, 10 = 62/8 = 7.75

Mean and Median

Median is a mid-point

2, 27, 11, 19, 33, 22, 4, 2, 19, 24, 22, 40, 38

2, 2, 4, 11, 19, 19, **22**, 22, 24, 27, 33, 38, 40 odd # values

2, 2, 4, 11, 19, 19, **22, 24**, 27, 33, 38, 40 even # values

23 22+24=46/2

What does this mean to us?

- We need multiple data or focus points
  - Drill down to lower levels of detail
    - Exception reporting should be pushed from detail upwards….

- Long averages across periods mask/hide problems
  - 10, 10, 10, 10, 10, **100**, 10, 10, 10, 10, 10,10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10
    - Average is 12.4 *but it's 24% higher than the median, or the mode*

DB2 Performance is relative to work

- Work performed by the application
  - Getpages
  - Fetches
  - I/Os
- Class 1 elapsed time
  - Ratio - compared to class 2 elapsed time
- Class 2 elapsed time (online transactions)
  - .050 seconds
  - .100 seconds
  - .500 seconds

Performance is relative. What are your expectations? Are they reasonable? What constitutes good performance? What is a reasonable amount of work for an online transaction? How do we characterize this?

Many well designed online transaction applications have average elapsed times of less than .020 seconds, and provide true sub-second response time to the end user. However, applications like this are becoming exceedingly rare in today's environment of bloated application code, poorly designed databases, and poorly coded SQL.

We will try to address what is good, what is reasonable, and basic metrics that indicate when performance is poor or marginal.

Keys to better performance

- Thought
- Inquisitiveness
- Reasonable
- Expectations
- Monitoring
  - Test/Dev - Early Warning
- Performance History
  - Yesterday?
  - Last week?

- Data relationships
- Knowing what is good, or bad
  - And why
- Caring about it
- Not waiting for complaints
- It can *Always be Better*

9

Performance is everyone's job, and responsibility!

Simply caring about performance issues, and looking at data will provide opportunities to make things better.

Those who wait for complaints, or wait for someone else to take action are not serving their company well - they are having a negative impact on the profitability of the entire corporation.

When you find something, and fix it - tell the world!!

Tell them how much you saved them, and how many more opportunities exist in your systems and applications.

Delaying or avoiding a processor upgrade is worth millions!!

Top down tuning

- Important/Meaningful periods
- Summary data – *don't get buried in detail*
- What looks *big*, or *strange*, or *not like* most other things
- Don't assume others are doing their job properly
- Use multiple tools to look at performance

It is not usually helpful to look at reports summarized over an entire days, or several days. You should be primarily concerned with the peak workload periods - if you can process well there, the rest of the time will run just fine….

Don't assume that others are doing their jobs, or even that they always know what they are doing - unless you are confident of a persons ability. Even then, they may be overloaded and not have the time do everything you think they are doing. Priorities of work assigned to others changes at the discretion of their management, so unless you work for the same person you can't always be sure what the priorities are.

Different tools often use a slightly different perspective to look at the same set of data - so seek input from all tools available to you.

Top down tuning

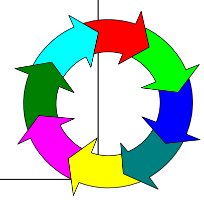
- Know what values are good, what is OK, what is bad
- Know what is ***normal for your system***
- Exception reporting
- Don’t get too bogged down in detail…
- But - be able to drill down to find the lower level data

11

It is really important to understand what performance numbers are good, and then what is *normal* for your system and applications. Tracking performance over time is a great way to find problems before they become critical.

Performance is like a traffic circle

- Application problems may lead to the DB2 system
- DB2 system problems often lead to application design and SQL coding
- Application and system problems may lead to DASD, z/OS, CICS, IMS
- z/OS performance problems may lead to applications or DB2 memory consumption



Memory and paging V8 and V9..

12

As you find a problem, as correct it, other performance problems will surface that may not have been obvious before.

Monitoring and tuning is a constant process, not a one time exercise.

The only exception to this is if/when your environment is static - no application changes, no workload or data growth, no migration to newer releases or maintenance levels….

System Overhead

- (C2 Elapsed - C3 Wait) / C2 CPU

- So what's a *reasonable ratio*?
  - Online IMS/CICS transactions < 1.5
  - Batch 10.0 + or -

- These are affected by:
  - Processor busy rate, possible system paging
  - Address space priorities, WLM setup

13

An area that is often overlooked is the overall system overhead. If the priorities for all the address spaces do not have the correct relationship, varying amounts of overhead will slow your work.

Is your processor more than 95% busy on a regular basis? If so, your performance is hurt by the machine busy rate. Running at 100% is not a case of using the machine to its fullest capacity - it's a case of hurting all performance because the processor is overloaded!

Important application performance data

- Workload
- Class 1 - 2 relationship
- Class 3 Wait time
  - Components
  - Avg I/O times
  - Total % of Class 2
- Deadlocks
- Timeouts

- Buffer pool performance
  - I/O per second **(important)**
    - Type of I/Os
  - Application hit % **(useless)**
  - System hit % **(historical & not useful)**
    (GP-PagesRd)/GP
- Ridpool failures
  - How many
  - Why
- Online transaction - Seq. scan

14

You must always consider the amount of work your application is performing - look at the number of getpages, fetches, I/Os, locks, etc, etc.

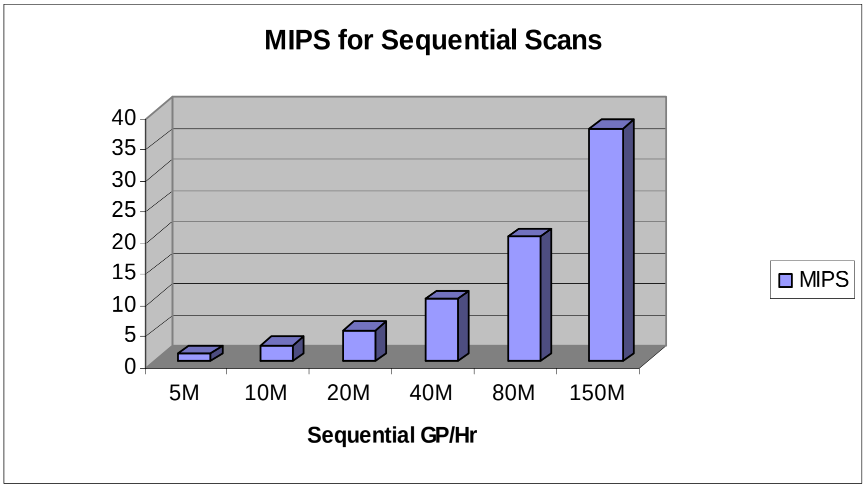
Look for the problem areas. What can be improved?

Calculate the System BP IO Rates, calculate the avg. I/O wait time by dividing the total synch wait time by the number of I/Os.

Look at any failures of resources, and look for things that don't seem normal to you.

Much of this can be programmed into your online monitor exception reports, or other reporting facilities. You really want a reasonable level of exception reporting so you don't have to look at tons of detail reports.

How much do you think Getpages cost?



Many large systems today are > ¼ Billion GP/Hr

15

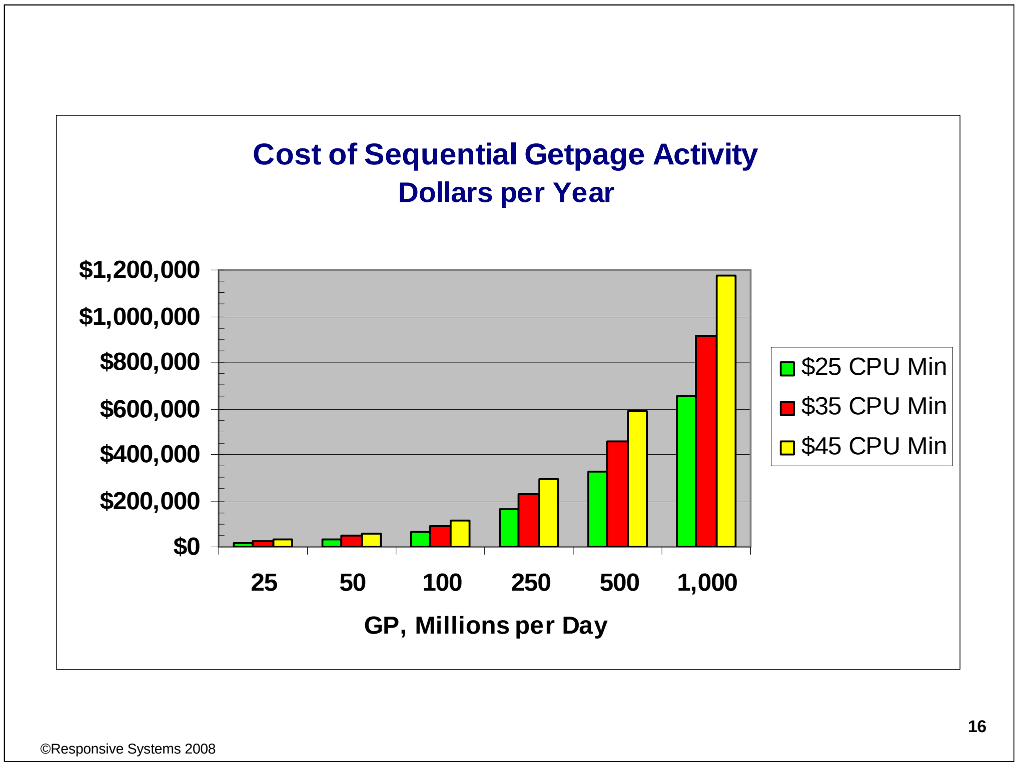
Finding one object dominating your system with sequential I/O?

One company fixed a problem like this years ago, and got back 18% of the machine and cancelled their upgrade !

Think you don't have sequentially accessed data in your system? Don't have some indexes with sequential scan?

Think again…. Most installations do – and nobody is complaining about performance. But management is complaining about CPU consumption and having to upgrade…..

But they won't spend a nickel to tune!!!



This is based on a 2064 processor with 210 MIP engine speeds.

Application Delay issues

```
LOCATION: CASISDBDG                OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V3)                  PAGE: 1-4
GROUP: DBDG                                  ACCOUNTING REPORT - LONG                 REQUESTED FROM: 06/14/07 20:05:00.00
      MEMBER: DDG2                                                                                TO: 06/14/07 20:09:00.00
   SUBSYSTEM: DDG2                           ORDER: PLANNAME-REQLOC                  INTERVAL FROM: 06/14/07 20:05:25.65
 DB2 VERSION: V8                               SCOPE: MEMBER                                    TO: 06/14/07 20:08:59.11
 PLANNAME: DISTSERV  REQLOC: 10.2.20.23
 ELAPSED TIME DISTRIBUTION                                            CLASS 2 TIME DISTRIBUTION
 ----------------------------------------------------------------     ----------------------------------------------------------
 APPL    |============================================> 87%           CPU     |==========> 21%
 DB2     |==> 4%                                                      NOTACC  |======> 12%
 SUSP    |====> 9%                                                    SUSP    |==================================> 67%     ***

 AVERAGE       APPL(CL.1)  DB2 (CL.2)  IFI (CL.5)   CLASS 3 SUSPENSIONS   AVERAGE TIME  AV.EVENT    HIGHLIGHTS
 ------------  ----------  ----------  ----------   --------------------  ------------  --------    --------------------------
 ELAPSED TIME    0.139919    0.018450         N/P   LOCK/LATCH(DB2+IRLM)      0.003620      1.28    #OCCURRENCES    :    33700
  NONNESTED      0.139919    0.018450         N/A   SYNCHRON. I/O             0.005955      2.12    #ALLIEDS        :        0
  STORED PROC    0.000000    0.000000         N/A    DATABASE I/O             0.000017      0.02    #ALLIEDS DISTRIB:        0
  UDF            0.000000    0.000000         N/A    LOG WRITE I/O            0.005938      2.11    #DBATS          :     3370
  TRIGGER        0.000000    0.000000         N/A   OTHER READ I/O            0.000006      0.00    #DBATS DISTRIB. :        0
                                                    OTHER WRTE I/O            0.000321      0.18    #NO PROGRAM DATA:        0
 CP CPU TIME     0.004498    0.003869         N/P   SER.TASK SWTCH            0.000016      0.00    #NORMAL TERMINAT:     3370
  AGENT          0.004498    0.003869         N/A    UPDATE COMMIT            0.000002      0.00    #ABNORMAL TERMIN:        0
   NONNESTED     0.004498    0.003869         N/P    OPEN/CLOSE               0.000000      0.00    #CP/X PARALLEL. :     3370
   STORED PRC    0.000000    0.000000         N/A    SYSLGRNG REC             0.000000      0.00    #IO PARALLELISM :        0
   UDF           0.000000    0.000000         N/A    EXT/DEL/DEF              0.000014      0.00    #INCREMENT. BIND:        0
   TRIGGER       0.000000    0.000000         N/A    OTHER SERVICE            0.000000      0.00    #COMMITS        :    33639
  PAR.TASKS      0.000000    0.000000         N/A   ARC.LOG(QUIES)            0.000000      0.00    #ROLLBACKS      :       64
                                                    ARC.LOG READ              0.000000      0.00    #SVPT REQUESTS  :        0
  IIPCP CPU      0.000000         N/A         N/A   DRAIN LOCK                0.000000      0.00    #SVPT RELEASE   :        0
                                                    CLAIM RELEASE             0.000000      0.00    #SVPT ROLLBACK  :        0
 IIP CPU TIME    0.000000    0.000000         N/A   PAGE LATCH                0.000151      0.06    MAX SQL CASC LVL:        0
                                                    NOTIFY MSGS               0.000000      0.00    UPDATE/COMMIT   :     1.92
 SUSPEND TIME    0.000000    0.012340         N/A   GLOBAL CONTENTION         0.000185      0.22
  AGENT               N/A    0.012340         N/A   COMMIT PH1 WRITE I/O      0.000000      0.00
  PAR.TASKS           N/A    0.000000         N/A   ASYNCH CF REQUESTS        0.002085      2.94
  STORED PROC    0.000000         N/A         N/A   TOTAL CLASS 3             0.012340      6.80
  UDF            0.000000         N/A         N/A
```

17

Within this five minute interval, there were 33,639 commits, and 64 ROLLBACKS.

Note that 67% of the application elapsed time is *WAIT*. Log Write IO is 48% of the Class 3 Wait Time !

Application Delay issues

```
ELAPSED TIME DISTRIBUTION                                           CLASS 2 TIME DISTRIBUTION
----------------------------------------------------------------    ----------------------------------------------------------
------
APPL     |======================================> 75%               CPU     |=====> 11%
DB2      |==> 5%                                                    NOTACC  |====> 9%
SUSP     |==========> 20%                                           SUSP    |=========================================> 80%

AVERAGE       APPL(CL.1)  DB2 (CL.2)  IFI (CL.5)    CLASS 3 SUSPENSIONS   AVERAGE TIME  AV.EVENT    HIGHLIGHTS
------------  ----------  ----------  ----------    --------------------  ------------  --------    --------------------------
ELAPSED TIME    0.181775    0.045574         N/P    LOCK/LATCH(DB2+IRLM)      0.024806      2.51    #OCCURRENCES       :    20890
 NONNESTED      0.181775    0.045574         N/A    SYNCHRON. I/O             0.004653      2.21    #ALLIEDS           :        0
 STORED PROC    0.000000    0.000000         N/A     DATABASE I/O             0.000098      0.04    #ALLIEDS DISTRIB:           0
 UDF            0.000000    0.000000         N/A     LOG WRITE I/O            0.004555      2.17    #DBATS             :     2089
 TRIGGER        0.000000    0.000000         N/A    OTHER READ I/O            0.000028      0.01    #DBATS DISTRIB.    :        0
                                                    OTHER WRTE I/O            0.000259      0.09    #NO PROGRAM DATA:           0
CP CPU TIME     0.005798    0.005033         N/P    SER.TASK SWTCH            0.000018      0.00    #NORMAL TERMINAT:        2089
 AGENT          0.005798    0.005033         N/A     UPDATE COMMIT            0.000007      0.00    #ABNORMAL TERMIN:           0
  NONNESTED     0.005798    0.005033         N/P     OPEN/CLOSE               0.000000      0.00    #CP/X PARALLEL.    :     2089
  STORED PRC    0.000000    0.000000         N/A     SYSLGRNG REC             0.000000      0.00    #IO PARALLELISM    :        0
  UDF           0.000000    0.000000         N/A     EXT/DEL/DEF              0.000011      0.00    #INCREMENT. BIND:           0
  TRIGGER       0.000000    0.000000         N/A     OTHER SERVICE            0.000000      0.00    #COMMITS           :    20839
 PAR.TASKS      0.000000    0.000000         N/A    ARC.LOG(QUIES)            0.000000      0.00    #ROLLBACKS         :       64
                                                    ARC.LOG READ              0.000000      0.00    #SVPT REQUESTS     :        0
 IIPCP CPU      0.000000         N/A         N/A    DRAIN LOCK                0.000000      0.00    #SVPT RELEASE      :        0
                                                    CLAIM RELEASE             0.000000      0.00    #SVPT ROLLBACK     :        0
IIP CPU TIME    0.000000    0.000000         N/A    PAGE LATCH                0.000985      0.17    MAX SQL CASC LVL:           0
                                                    NOTIFY MSGS               0.000000      0.00    UPDATE/COMMIT      :     2.04
SUSPEND TIME    0.000000    0.036660         N/A    GLOBAL CONTENTION         0.001066      0.60    SYNCH I/O AVG.     : 0.002103
 AGENT               N/A    0.036660         N/A    COMMIT PH1 WRITE I/O      0.000000      0.00
 PAR.TASKS           N/A    0.000000         N/A    ASYNCH CF REQUESTS        0.004844      4.32
 STORED PROC    0.000000         N/A         N/A    TOTAL CLASS 3             0.036660      9.91
```


Hopefully obvious the big problem is locking….

Little Red Flags…

```
SUBSYSTEM SERVICE COMPONENT
----------------------------
IDENTIFY                            31529
CREATE THREAD                      757527
SIGNON                              28235
TERMINATE                          789499
ABORT                              180102
COMMIT PHASE 1                    3546312
COMMIT PHASE 2                    2910873
READ ONLY COMMITS                  643648
UNITS OF REC GONE INDOUBT               0
UNITS OF REC INDOUBT RESOLVED           0
SYNCHS (SINGLE PHASE COMMIT)      1857513
```


Hopefully, it should be obvious that large numbers of aborts can be a problem.

Little Red Flags…

```
From 03/28/08 09:00:00 To 03/28/08 09:30:00
EXCEPTIONS Crit Warn Info
 Subsystem   11    2   16                         EDM POOL
                        BUFFERS                    Free Pg  20687     % Total   41.4
ADDRESS SPACE CPU        Warnings        2440      DBD Lds      0     % Rqsts    0.0
 DBAS   10:24.95         Act Pools          0      CT  Lds      0     % Rqsts    0.0
 SSAS      55.92         %NStl Pgs        7.8      PT  Lds     26     % Rqsts    0.0
 IRLM       2.70         Getpages  135233941       Dyn Ins      0     % Rqsts    0.0
 DIST   55:00.61         Sync Rds     5296912
                         Read Eff        25.5     LOCKING           LOGGING
POOL FAILURES            Buf Updts    8210578      Suspend   2629    Dlyd Wrts     0
 RID              0      Pg Writes     937334      Escalate     0    Arch Read     0
 EDM              0      Write I/O      76331      Timeout      0    Min/Ckpt *****
                                                   Deadlock     0    Warnings      0
THREADS                 SQL                       DATA SHARING      STORED PROCS
 Created       44345     Dynamic        25710      Group   DSNDBXX   CALLs       352
 Terminated    44653     In+Up+Dl     2171956      Member DB66       Fails         0
 Aborts         5986     Open+Sel     1153851
 Commits       84603
```

20

Hopefully, it should be obvious that large numbers of aborts can be a problem.
At this time, there's a lot of free space in the EDM pool.

This should be tracked across other time frames to see if memory is wasted.

DB2 Read I/Os

Two basic types of object access and resulting I/Os

- Random or Synchronous I/O
- Prefetch or Asynchronous I/O
  - Sequential
  - Dynamic
  - List Prefetch

21

There are many thresholds and exceptions that should be monitored and tracked on a regular basis, and some are more critical than others.

SPTH, DMTH, WITH, and EDMPool failures are critical and require immediate action - IWTH is not a problem unless you have hit SPTH and DMTH first.

Log Buffer unavailable and Unavailable Read/Write engines often points to poor dasd subsystem performance, and may have to be addressed from both sides to eliminate the problem.

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 %
    - Don't need a large pool

WkSet is subjective – every system and application is different

Change the size of a pool, and every object wkset changes *almost….*

Move an object to a different pool, and the WkSet size changes

You must be able to *predict the effect of changes*, or you are guessing with your performance life….

22

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.

Buffer Pool – Tuning Methodology

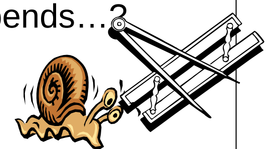
- What is a transient object?
  - Large wkset size
  - Wkset grows proportionally to pool size increases
  - Pages not frequently re-referenced
- When do we care?
  - When it impacts other objects in the pool
    - If it has a low GP and IO rate, we don't care
      - High, we care a lot - and immediately..
- Typically very large, and random or sequential
  - Random may have low or high Dynamic Prefetch…

We don't have any control over dynamic prefetch. This is determined by the buffer manager, at the application cursor level. It may help performance of the object in use, and it may also hurt other objects in the pool.

Pool Performance

- Is rarely linear
  - More memory does not always improve performance
  - Small pool increases often don’t show any improvement
  - Sometimes additional “smallish” increases can provide substantial improvements
  - Many times large increases do not help
    - Sometimes they do….
    - It may depend on what “large” means to you…
      - A number, percentage, or it depends…?

DB2 Performance – it depends…



24

We will see from data later in this presentation that pool performance is not linear. Doubling the pool size does not double the hit ratio, or cut the I/O rate in half. We will always reach a point of diminishing returns, when adding buffers to a pool.

Now, I realize that some of the above items seem contradictory, and will explain them in more detail during the presentation.

The data shown in future slides will also illustrate all the above points.

Performance tuning complaint…

"I modified a bufferpool to set the DWQT from the default 50% value to DWQT=4 and VDWQT = 0 - No other change -

After the change, a program doing the inserts (in ascending key), and delete on it, took a lot of time, 40min instead of 17min (it is a temporary table), STROBE shows that 90% of the time was on the Insert, and 80% of the Wait was on "OTHER WRITE".

I proved to them that the INDEX and TS has never been organized , and an increase in volume can magnify the problem. (Statistics showed 70,000 inserts instead of 10,000 during this day) ***That's 7 times the workload*****, but only 2.3 times the elapsed**

They told me that the problem comes from the change, because even if it was disorganized , other executions were good… "

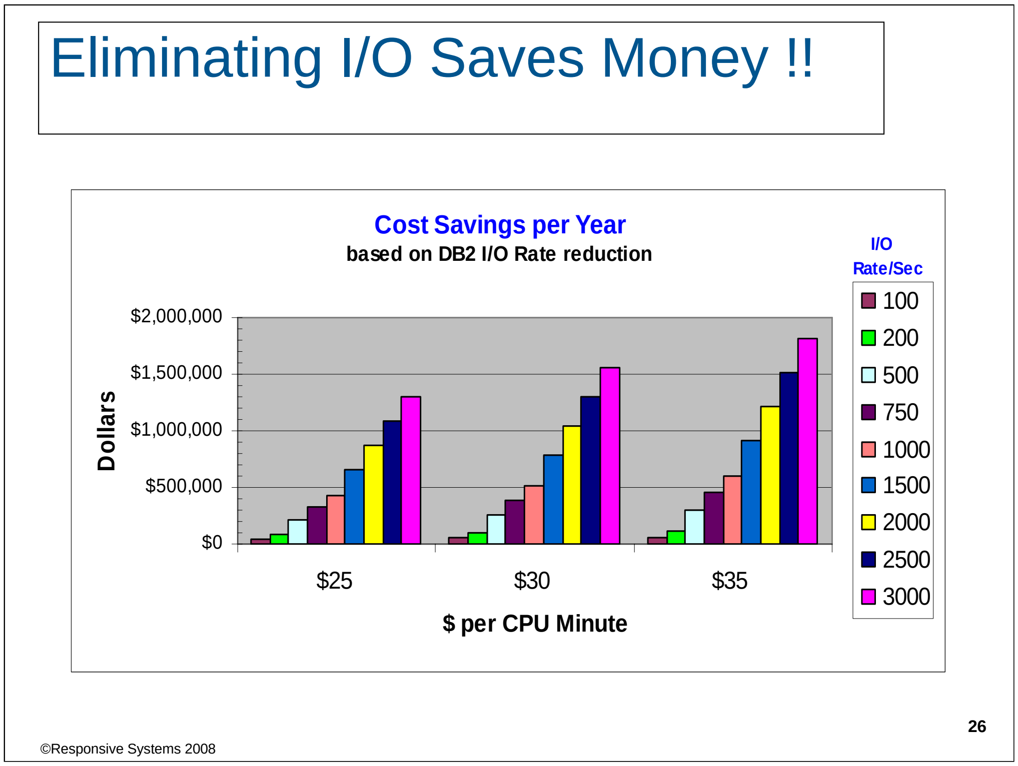
Three blind mice….

History data, history data…

25

Wow, do you think that increasing the workload by a factor of 7 has any impact? Actually, performance was much better, not worse, and not longer.

The elapsed time was only 2.3 times the original job elapsed times.



These I/O rate per sec savings, up through 2,500 per second, have been achieved by clients

Buffer Pool Performance Data - Red Flags

BP5 GENERAL	QUANTITY	/SECOND	/THREAD	/COMMIT
CURRENT ACTIVE BUFFERS	410.10	N/A	N/A	N/A
UNAVAIL.BUFFER-VPOOL FULL	0.00	0.00	0.00	0.00
NUMBER OF DATASET OPENS	8446.00	0.10	0.01	0.00
BUFFERS ALLOCATED - VPOOL	7507.00	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	300.00	N/A	N/A	N/A
PREF.I/O STREAMS REDUCTION	3382.00	0.04	0.00	0.00
PARALLEL QUERY REQUESTS	88003.00	1.02	0.12	0.03
PARALL.QUERY REQ.REDUCTION	215.00	0.00	0.00	0.00
PREF.QUANT.REDUCED TO 1/2	866.2K	10.06	1.21	0.34
PREF.QUANT.REDUCED TO 1/4	73224.00	0.85	0.10	0.03

What critical data items are missing from these sets of data?

27

There are a lot of red flags in this data report, all related to a lack of buffers available for prefetch.

Buffer Pool Performance Data - Red Flags

BP5 READ OPERATIONS	QUANTITY	/SECOND	/THREAD	/COMMIT
BPOOL HIT RATIO (%)	31.59			
GETPAGE REQUEST	179.4M	2083.30	250.37	70.40
GETPAGE REQUEST-SEQUENTIAL	130.1M	1510.58	181.54	51.05
GETPAGE REQUEST-RANDOM	49309.8K	572.71	68.83	19.35
SYNCHRONOUS READS	5285.0K	61.38	7.38	2.07
SYNCHRON. READS-SEQUENTIAL	1797.4K	20.88	2.51	0.71
SYNCHRON. READS-RANDOM	3487.7K	40.51	4.87	1.37
GETPAGE PER SYN.READ-RANDOM	14.14			
SEQUENTIAL PREFETCH REQUEST	3856.3K	44.79	5.38	1.51
SEQUENTIAL PREFETCH READS	3684.0K	42.79	5.14	1.45
PAGES READ VIA SEQ.PREFETCH	109.6M	1273.45	153.05	43.04
S.PRF.PAGES READ/S.PRF.READ	29.76			
LIST PREFETCH REQUESTS	1328.8K	15.43	1.85	0.52
LIST PREFETCH READS	392.2K	4.56	0.55	0.15
PAGES READ VIA LIST PREFTCH	2049.5K	23.80	2.86	0.80
L.PRF.PAGES READ/L.PRF.READ	5.23			
DYNAMIC PREFETCH REQUESTED	197.4K	2.29	0.28	0.08
DYNAMIC PREFETCH READS	184.4K	2.14	0.26	0.07
PAGES READ VIA DYN.PREFETCH	5726.7K	66.51	7.99	2.25
D.PRF.PAGES READ/D.PRF.READ	31.06			
PREF.DISABLED-NO BUFFER	0.00	0.00	0.00	0.00
PREF.DISABLED-NO READ ENG	74.00	0.00	0.00	0.00
PAGE-INS REQUIRED FOR READ	0.00	0.00	0.00	0.00

28

Running out of Read Engines is often a sign of poor dasd performance.

Buffer Pool Performance Data - Red Flags

BP5 WRITE OPERATIONS	QUANTITY	/SECOND	/THREAD	/COMMIT
BUFFER UPDATES	18460.7K	214.41	25.77	9.39
PAGES WRITTEN	2220.6K	25.79	3.10	1.13
BUFF.UPDATES/PAGES WRITTEN	8.31			
SYNCHRONOUS WRITES	25517.00	0.30	0.04	0.01
ASYNCHRONOUS WRITES	1063.6K	12.35	1.48	0.54
PAGES WRITTEN PER WRITE I/O	2.04			
HORIZ.DEF.WRITE THRESHOLD	1.00	0.00	0.00	0.00
VERTI.DEF.WRITE THRESHOLD	451.00	0.01	0.00	0.00
DM THRESHOLD	0.00	0.00	0.00	0.00
WRITE ENGINE NOT AVAILABLE	85.00	0.00	0.00	0.00
PAGE-INS REQUIRED FOR WRITE	0.00	0.00	0.00	0.00

Running out of read engines & write engines is often a sign of DASD performance problems. 600 read engines, 300 write engines (apar).

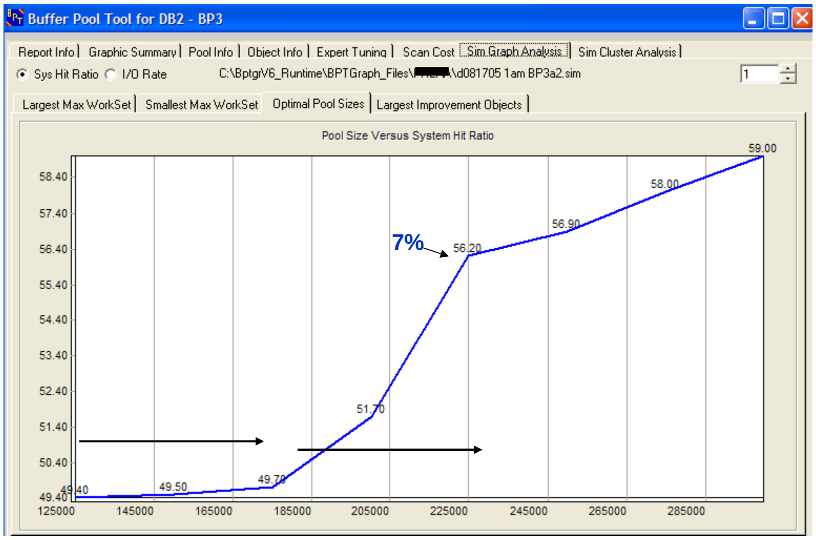
What critical data items are missing from these sets of data?

29

The number of buffers in the pool, and pool thresholds.. And the ELAPSED TIME for the data !!

V8 has 600 read and 300 write engines; however, this number is for the entire DB2 system, not one pool.

What does a Hit Ratio really tell you?



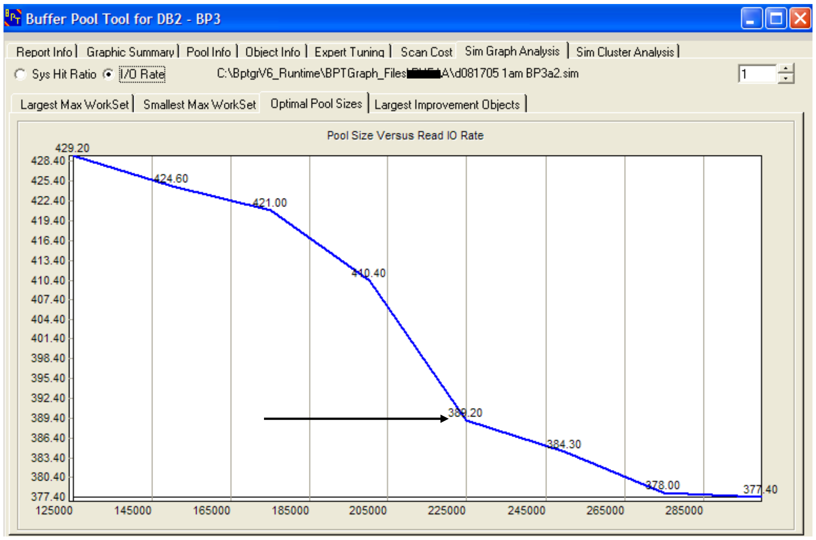
The first 50% does not get much payback

Ok, it shows you that performance is better. But how much better is it? How much CPU and elapsed times have been saved from I/O avoidance?

Increasing the pool by 50% does not give much payback, the next 50,000 shows a large improvement, and then the improvement curve flattens.

Again, it looks nice, but you can't take any of the numbers to the bank.

The I/O rate is a *measurable* Metric



Why does the next 50% help so much? **A critical WKSET was reached** 31

The I/O rate is convertible into CPU costs, and elapsed time savings.

This is not just a suggestion to make the pool larger, it shows you the real benefit, and where to stop.

It shows you that the first 50,000 additional buffers don't provide much payback, but the next 50,000 give a huge payback.

The large payback from the second increment of 50,000 buffers is because we passed a critical working set threshold for a heavily accessed object. As stated earlier, the wkset size of an object has nothing to do with the number of pages shown in the catalog. It is the number of pages in the pool at a specific point in time.

The I/O rate is a *meaningful* Metric

INTV DATE	INTV TIME	GET PAGE	SYNC IO	SEQ PREFETCH	LIST PREFETCH	DYN PREFETCH	IORATE /SEC	HIT RATIO	
2007-02-21	00.00.00	1,187,068	43,331	625,280	156,611	202	459	30.47%	++++
2007-02-21	00.30.00	**10,913,350**	**342,348**	**3,226,519**	**289,002**	**1,069,556**	**2,737**	54.85%	
2007-02-21	01.00.00	671,955	51,854	373,032	49,730	893	264	29.24%	
2007-02-21	01.30.00	743,700	67,202	435,841	2,948	95	281	31.95%	
2007-02-21	09.30.00	4,105,758	124,423	2,961,328	12,364	69,313	1,760	22.85%	
2007-02-21	10.00.00	4,232,240	100,715	3,088,771	9,277	64,952	1,813	22.88%	
2007-02-21	10.30.00	3,127,959	70,646	2,452,744	18,310	58,303	**1,444**	16.88%	++++
2007-02-21	11.00.00	3,890,741	112,328	2,520,929	9,911	51,934	**1,497**	30.73%	
2007-02-21	11.30.00	4,011,848	117,173	2,394,283	11,760	74,255	1,443	35.26%	
2007-02-21	12.00.00	4,580,930	109,663	2,339,369	639,288	68,607	1,754	31.09%	
2007-02-21	12.30.00	6,150,020	188,803	3,202,333	11,852	64,002	1,926	43.63%	

If the hit ratio was *meaningful*, it would not show a big increase when there is a large increase to the IO rate ***% GP increase vs. IO***
yes we did find a lot more of the pages in the pool….

32

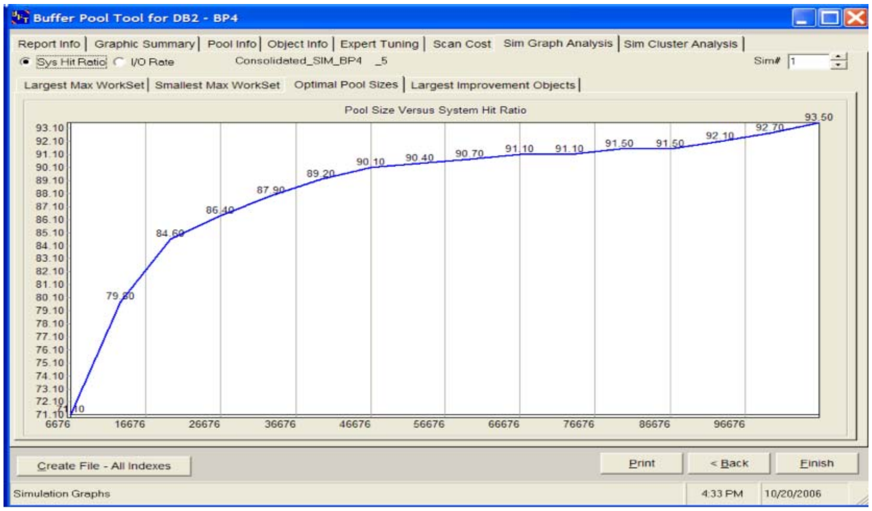
The IO rate can increase, and the hit ratio can increase.

The IO rate can decrease, and the hit ratio can decrease.

This is the opposite the "expectation"

Changes in the workload, the type of accesses taking place, and the objects in use, cause the counter-intuitive swings of the hit ratio. But it's counter-intuitive only if you are looking uniquely at getpages and IOs.

Bigger is *not* always better - 1

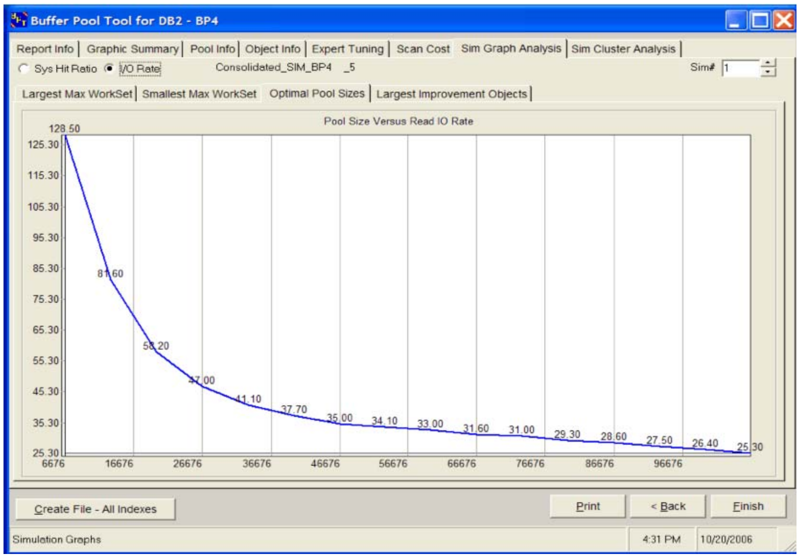


33

The increase for the buffer pool hit ratio flattens, and drops to .4% for every 10,000 buffers, 40 meg of memory.

As stated many times, these gains cannot be equated to elapsed times or CPU reductions.

Bigger is *not* always better - 2



34

The IO reduction/gain from increasing the pool size flattens, and eventually drops to only 1 IO/Sec per 10,000 buffers. This is not considered a useful payback for 40 Meg of memory.

18,000 to 36,000 saves about 15 IO/Sec., and we can see how the rate of saving drops off to almost nothing as we continue to add more memory.

Overall, the 36-38,000 buffer range is the best range for performance/memory trade-off.

The previous slide with graph showed..

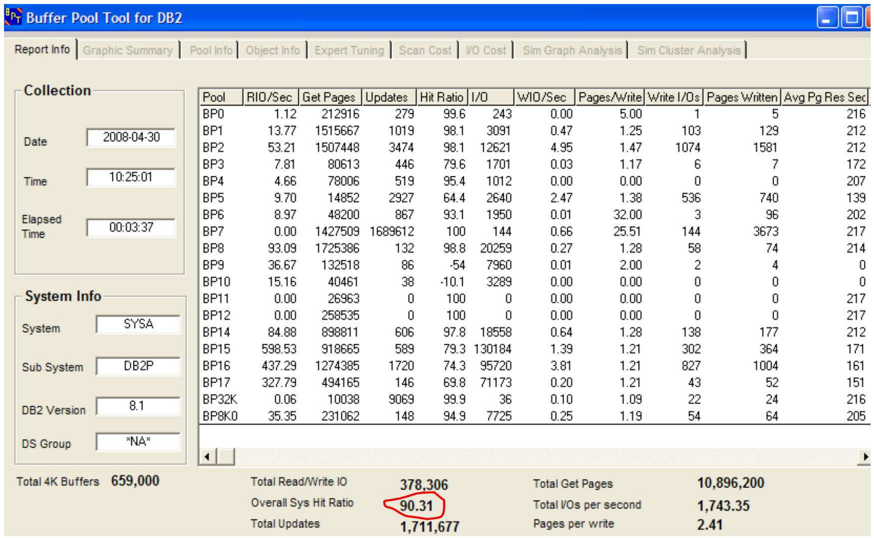
- When a pool is much too small, more memory will provide substantial improvements
- There is a point of limited, and possibly no return
  - Further increases provide very little gain, not enough to justify the added memory
  - The first tripling of size cut the IO rate 50%, ***70 IOs***
  - Then a doubling cut the IO rate by ***15 IOs***
  - The third doubling cut the IO rate by ***5 IOs***

 35

Roughly, increase 6,000 to 18,000 has a large payback – 70 IO/Sec, and is a 54% saving of the IO rate/sec.

Look at Hit Ratio again – Looks good…

Misleading – implies very good performance



Pool	RIO/Sec	Get Pages	Updates	Hit Ratio	I/O	WIO/Sec	Pages/Write	Write I/Os	Pages Written	Avg Pg Res Sec
BP0	1.12	212916	279	99.6	243	0.00	5.00	1	5	216
BP1	13.77	1515667	1019	98.1	3091	0.47	1.25	103	129	212
BP2	53.21	1507448	3474	98.1	12621	4.95	1.47	1074	1581	212
BP3	7.81	80613	446	79.6	1701	0.03	1.17	6	7	172
BP4	4.66	78006	519	95.4	1012	0.00	0.00	0	0	207
BP5	9.70	14852	2927	64.4	2640	2.47	1.38	536	740	139
BP6	8.97	48200	867	93.1	1950	0.01	32.00	3	96	202
BP7	0.00	1427509	1689612	100	144	0.66	25.51	144	3673	217
BP8	93.09	1725386	132	98.8	20259	0.27	1.28	58	74	214
BP9	36.67	132518	86	-54	7960	0.01	2.00	2	4	0
BP10	15.16	40461	38	-10.1	3289	0.00	0.00	0	0	0
BP11	0.00	26963	0	100	0	0.00	0.00	0	0	217
BP12	0.00	258535	0	100	0	0.00	0.00	0	0	217
BP14	84.88	898811	606	97.8	18558	0.64	1.28	138	177	212
BP15	598.53	918665	589	79.3	130184	1.39	1.21	302	364	171
BP16	437.29	1274385	1720	74.3	95720	3.81	1.21	827	1004	161
BP17	327.79	494165	146	69.8	71173	0.20	1.21	43	52	151
BP32K	0.06	10038	9069	99.9	36	0.10	1.09	22	24	216
BP8K0	35.35	231062	148	94.9	7725	0.25	1.19	54	64	205

36

If we show the overall System Hit Ratio, it looks and sounds great. But the IO rate is quite high…. The is a lot of room for tuning to reduce the IO rate, save CPU, reduce transaction elapsed times, and improve productivity. The pools with the highest IO rates are quite a bit lower than that overall System Hit Ratio.

Sort Pool Performance

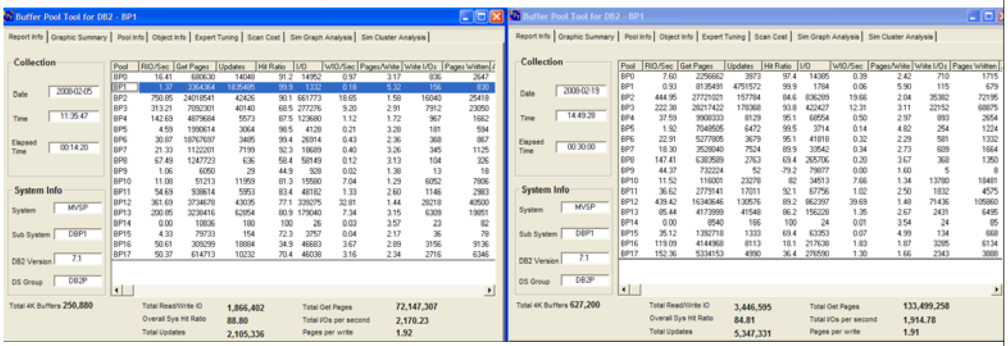
```
LOCATION: DB2PROD          GROUP : DSNGDSN           MAINVIEW FOR DB2                INTERVAL START: 2020-01-01-00.00.31.0000
SSID    : DSN              MEMBER: DSN1             PERFORMANCE REPORTER             INTERVAL END  : 1991-01-01-00.00.01.0000
VERSION : 81               SCOPE : MEMBER          STATISTICS LONG  REPORT           INTERVAL      : 00001     PAGE 0006
----BP7    GENERAL------------ --------QTY  ----BP7    READ OPERATIONS---- --------QTY  ----BP7    GLOBAL BP---------- --------QTY
CURRENT ACTIVE BUFFERS               21775  GETPAGES                         17054.28K  SYNC RD INV BUFF -WITH DATA               0
VP BUFFER POOL FULL                      0  GETPAGE SEQ REQ                  11306.15K                 - NO DATA                  0
SUCCESSFUL OPEN                          0  GETPAGE RANDOM REQ                5748.13K  SYNC RD NOT FOUND-WITH DATA               0
VP BUFFERS ALLOCATED                 25000  SYNC READ I/O                         2255                 - NO DATA                  0
HP BUFFERS ALLOCATED                     0  SYNC READ IO SEQ REQ                  2154  ASYNC READ - DATA RETURNED                0
EXPANDED STORAGE HP BUFFERS              0  SYNC READ IO RANDOM REQ                101                 - NO DATA                  0
MIGRATED DS ENCOUNTERED                  0  GETPAGE/SYNC READ RANDOM             56912  SYNC PAGES WRITTEN -CHANGED               0
RECALL TIMEOUTS                          0  SEQ PREFETCH REQ                     77737                       -CLEAN               0
HP EXP/CONTRACTION                       0  SEQ PREFETCH READ IO                 57118  ASYNC PAGES WRITTEN-CHANGED               0
VP EXP/CONTRACTION                       0  SEQ PREFETCH PAGES READ            449.87K                       -CLEAN               0
EXPAND SOS FAIL                          0  SEQ PREFETCH PAGES/READ                  7  CASTOUT -PAGES WRITTEN                    0
HWM PREFETCH IO STREAMS                  0  LIST PREFETCH REQUESTS                   0          -NO ENGINE                        0
PRFETCH IO STREAMS REDUCED               0  LIST PREFETCH READ IO                    0          -CLASS THRESHOLD                  0
REQUESTS FOR PARALLELISM                 0  LIST PREFETCH PAGES READ                 0          -GBP THRESHOLD                    0
PARALLEL REDUCTION-NO BUFF               0  LIST PREFETCH PAGES/READ               N/C  NO WRITE ENGINE                           0
PREFETCH QTY CUT TO 1/2                  0  DYNAMIC PREFETCH REQUESTS                0  READ FAIL  -STORAGE                       0
PREFETCH QTY CUT TO 1/4                  0  DYNAMIC PREFETCH READ IO                 0  WRITE FAIL -STORAGE                       0
                                            DYNAMIC PREFETCH PAGES READ              0  RD STG STATS(5.1)/OTHER(4.1)              0
----BP7    WRITE OPERATIONS--- --------QTY  DYNAMIC PREFETCH PAGES/READ            N/C  GBP CHECKPOINT                            0
PAGE UPDATES                     13349.07K  PF DISABLED - NO BUFFER       ++       171  GBP REBUILD                               0
PAGES WRITTEN                      626.00K  PREFETCH DISABLED-NO ENGINE              0  UNLOCK CASTOUT                            0
BUFF UPDATES/PAGES WRITTEN              21  MVPG PAGES SYNC HP->VP                   0  READ CASTOUT CLASS                        0
SYNC WRITES                              0  MVPG PAGES ASYNC HP->VP                  0  READ CASTOUT STATISTICS                   0
ASYNC WRITE IO                       21696  HP->VP MVPG FAIL                         0  DELETE DIR/DATA ENTRIES                   0
ASYNC WRITES + SYNC WRITES           21696  DATA MOVER ASYNC HP->VP                  0  READ DIRECTORY INFORMATION                0
HORIZONTAL DEF. WR REACHED    ++      1478  DATA MOV ASYNC FAIL HP->VP               0  REGISTER PAGE                             0
VERTICAL DEFER WR REACHED                0  PAGEINS FOR READ IO                      0  UNREGISTER PAGE                           0
DM CRITICAL REACHED                      0  BPOOL HIT RATIO-ALL(%)                  97  REGISTER PAGE LIST                        0
NO WRITE ENGINE                          0  BPOOL HIT RATIO-RANDOM(%)               99  REGISTER PAGE LIST-RD CHNGE               0
PAGES SYNC VP->HP                        0                                              REGISTER PAGE LIST-RD CLEAN               0
PAGES ASYNC VP->HP                       0  ----BP7    SORT/MERGE--------- --------QTY  EXPLICIT CROSS INVALIDATES                0
PAGES WRITE FAIL VP->HP                  0  MAX WORKFILE IN MERGE                    0  DUPLEX-WRITE REQUEST                      0
DATA MOVER ASYNC VP->HP                  0  NUMBER MERGE PASSES                  31803        -WRITE FAIL                         0
DATA MV ASYN FAIL VP->HP                 0  MERGE PASSES/INSUFF BUFFER               0        -DELETE NAME LIST                   0
PAGEINS FOR WRITE IO                     0  WORKFILES REJECTED LOW BUFF              0        -DELETE NAME                        0
                                            TOTAL WORKFILES IN MERGE             64253        -READ CASTOUT STATS                 0
                                            WKFILES NOT CREATED-NO BUF               0
                                            PREFETCH DISABLED-WK FILES               0
                                            PAGES FOR DESTRUCTIVE READ        4148.62K
                                            DEQUE FROM VDWQ DISTRUCT RD       3707.49K
```


Effect of setting the sort pool thresholds *<u>vdwqt and dwqt too high</u>*. The pool hit spth, and prefetch was disabled.

Current Active Buffers – is a snapshot at the moment the Statistics record is produced, not a high water mark, and it's at 87% of the total pool buffers.

While performance looks good based on the Hit Ratios, the pool is having performance problems highlighted in Red. Other important metrics are highlighted in Blue.

Sort Pool – BP1



The sort pool was increased from 51,200 buffers to 307,200 buffers

Bigger is *NOT* always better… saved .4 IO/sec - *insignificant*

Wasting a Gigabyte of memory…..

vdwqt=90, dwqt=90 pool hitting spth and turning off prefetch

38

Since the vdqwt and dwqt thresholds were incorrectly set at 90%, the pool was hitting spth, and turning off prefetch. An exception reporting system flagged the threshold problem, and suggested making the pool larger. It continued to hit spth for large sorts. The number of "buffers unavailable, or in-use" was generally 80% of the buffers, no matter how large the pool was made.

Sort Pool – *Bigger is not always better*

# Buffers	Read IO/Sec	Write IO/Sec
51,200	1.37	.18
256,000	1.25	.10
307,200	0.93	.06

The sort pool was oversized from the beginning, and memory would have been used more effectively on other pools. The reason for the expansive growth was….

Looking at one performance variable or exception may miss the real problem, and bad recommendations from an exception reporting product
You *need to understand performance, and evaluate recommendations…*

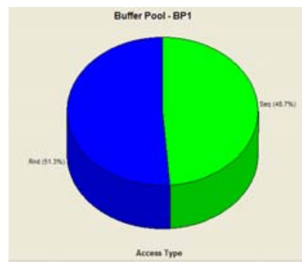
39

An analysis tool that only looked at the exception hitting spth, and merely recommended making the pool larger. No consideration was given to the vdwqt and dwqt settings, or the information that the pool consistently showed active (unavailable) pages at 80% of the pool.

Sort pool facts

- New pages are created in the pool, they are not read into the pool first – no initial read IO delay
- Pages that are written out, are not always read back
- Pools often have a high % of random getpages
- The prefetch read qty is 8 pages

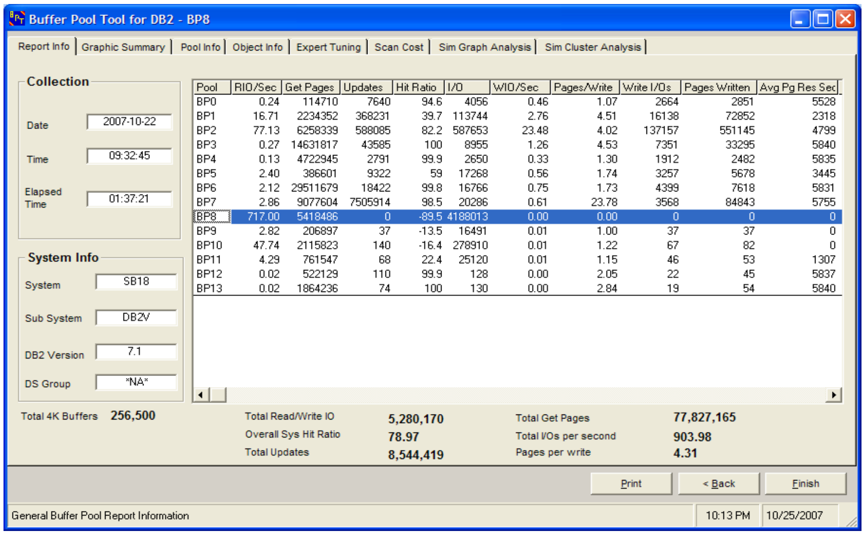
This is not an unusual access & usage illustration



40

There is a lack of published information regarding sort functions and processing methods.

Looking at interesting data



Pool	RIO/Sec	Get Pages	Updates	Hit Ratio	I/O	WIO/Sec	Pages/Write	Write I/Os	Pages Written	Avg Pg Res Sec
BP0	0.24	114710	7640	94.6	4056	0.46	1.07	2664	2851	5528
BP1	16.71	2234352	368231	39.7	113744	2.76	4.51	16138	72852	2318
BP2	77.13	6258339	588085	82.2	587653	23.48	4.02	137157	551145	4799
BP3	0.27	14631817	43585	100	8955	1.26	4.53	7351	33295	5840
BP4	0.13	4722945	2791	99.9	2650	0.33	1.30	1912	2482	5835
BP5	2.40	386601	9322	59	17268	0.56	1.74	3257	5678	3445
BP6	2.12	29511679	18422	99.8	16766	0.75	1.73	4399	7618	5831
BP7	2.86	9077604	7505914	98.5	20286	0.61	23.78	3568	84843	5755
BP8	717.00	5418486	0	-89.5	4188013	0.00	0.00	0	0	0
BP9	2.82	206897	37	-13.5	16491	0.01	1.00	37	37	0
BP10	47.74	2115823	140	-16.4	278910	0.01	1.22	67	82	0
BP11	4.29	761547	68	22.4	25120	0.01	1.15	46	53	1307
BP12	0.02	522129	110	99.9	128	0.00	2.05	22	45	5837
BP13	0.02	1864236	74	100	130	0.00	2.84	19	54	5840

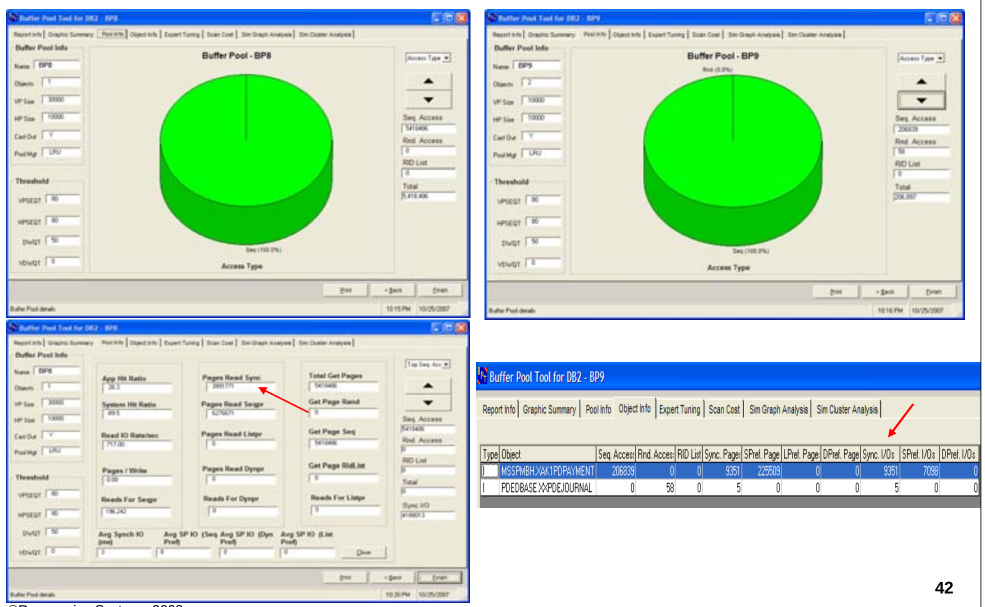
BP8 has the killer IO rate.

It has one object, a TS

BP9 has two objects, both indexes on the TS in BP8.

This is a heavy batch system, even though it is mid-morning.

Looking at interesting data



BP8 has the killer IO rate.

It has one object, a TS

BP9 has two objects, both indexes on the TS in BP8.

Email to client after looking at data

You have a classic pool thrashing scenario, complicated by an application design/access issue.

Are you running multiple concurrent batch jobs against this, attempting to get better throughput by hitting different partitions?

BP8
As you said, one object, 10 partitions. The access to this object is 100% sequential.
The high Synch IO tells me that the prefetched pages are thrown out ***before a low priority batch job can be dispatched to read them***, and they are **re-read using synch IO.**
A major performance, and cpu/cost killer.
** you need to have the batch job(s) run at a higher system priority
** if you are running several, against different partitions, try running a few less concurrent jobs

BP9
Two indexes on the TS in BP8
PDPAYMENT is 100% SP access - for an index???   This is a design or sql coding problem, or lack of full stats for the index and TS..
It appears that there were at least 9,351 complete scans of this index.
The other index has almost no usage, and all random.

Response from the client

Joel:

Dang you are good!! We had 3 jobs running going against 3 different partitions on the same table. I can't remember what SRVCLASS they were running in. Everybody has the ability to change SRVCLASS, which stinks, but that is my problem.

You say "9,351 complete scans of this index". I assume you are getting that number from this screen. How do you interpret this to be "complete scans" and not "pages read" since SYNC I/O's are one page at time?

Let's go back two slides and look at the data

This isn't genius, just experience.

44

All the getpages issued were sequential.

So what causes a synch IO?

When a prefetch stream starts, it issues 1 synch IO for the first page, a SP for pages 2-32, and a SP for pages 33-64. When you hit page 32, it issues a SP for page 65-96, etc, etc

I am making an assumption here, since I am not looking at the actual trace data from the collection file, but since this is how sequential prefetch is initiated.... I'm assuming that each synch IO indicates the start of a scan...

Back to the client

All the getpages issued were sequential
So - what causes a synch IO?

When a prefetch stream starts, it issues 1 synch IO for the first page, a SP for pages 2-32, and a SP for pages 33-64.

One synch IO, and two concurrent prefetch IOs.

When you hit page 32, it issues a SP for page 65-96, etc, etc

I am making an assumption here, since I am not looking at the actual trace data from the collection file, but since this is how sequential prefetch
is initiated.... I'm assuming that each synch IO indicates the start of a scan...

I already covered this in the discussion of 3 slides back

45

All the getpages issued were sequential.

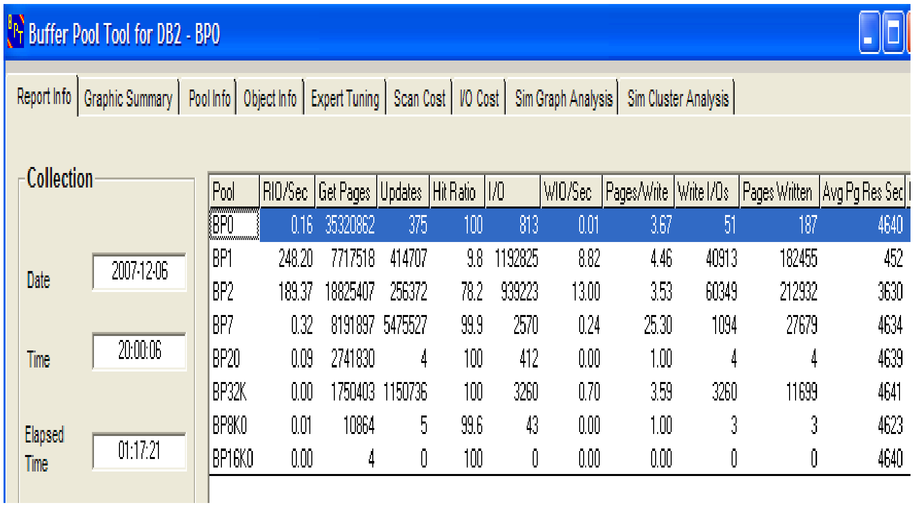
So what causes a synch IO?

When a prefetch stream starts, it issues 1 synch IO for the first page, a SP for pages 2-32, and a SP for pages 33-64. When you hit page 32, it issues a SP for page 65-96, etc, etc

I am making an assumption here, since I am not looking at the actual trace data from the collection file, but since this is how sequential prefetch

is initiated.... I'm assuming that each synch IO indicates the start of a scan...

What's happening in BP0?



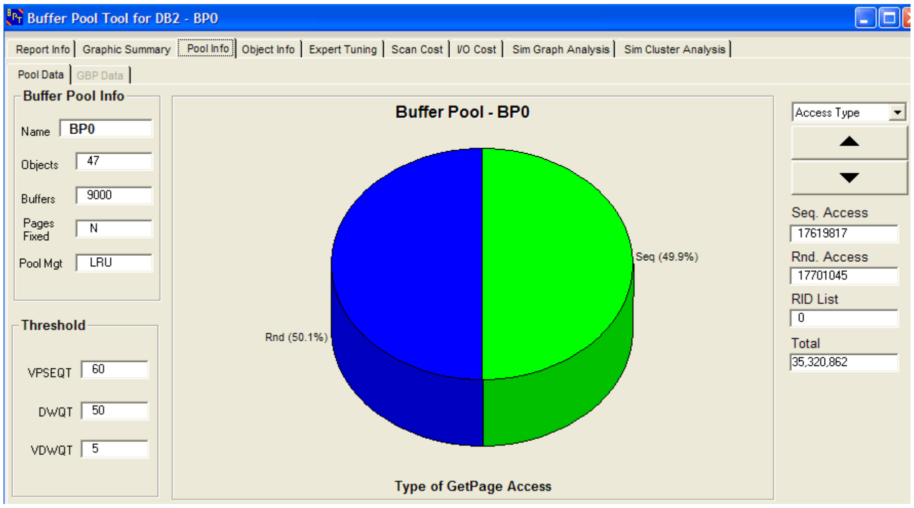
Pool	RIO/Sec	Get Pages	Updates	Hit Ratio	I/O	WIO/Sec	Pages/Write	Write I/Os	Pages Written	Avg Pg Res Sec
BP0	0.16	35320862	375	100	813	0.01	3.67	51	187	4640
BP1	248.20	7717518	414707	9.8	1192825	8.82	4.46	40913	182455	452
BP2	189.37	18825407	256372	78.2	939223	13.00	3.53	60349	212932	3630
BP7	0.32	8191897	5475527	99.9	2570	0.24	25.30	1094	27679	4634
BP20	0.09	2741830	4	100	412	0.00	1.00	4	4	4639
BP32K	0.00	1750403	1150736	100	3260	0.70	3.59	3260	11699	4641
BP8K0	0.01	10864	5	99.6	43	0.00	1.00	3	3	4623
BP16K0	0.00	4	0	100	0	0.00	0.00	0	0	4640

Why does it have twice as many Getpages as any other pool?

46

Very high number of Getpages, almost twice any other pool in the system. Application objects in the pool?

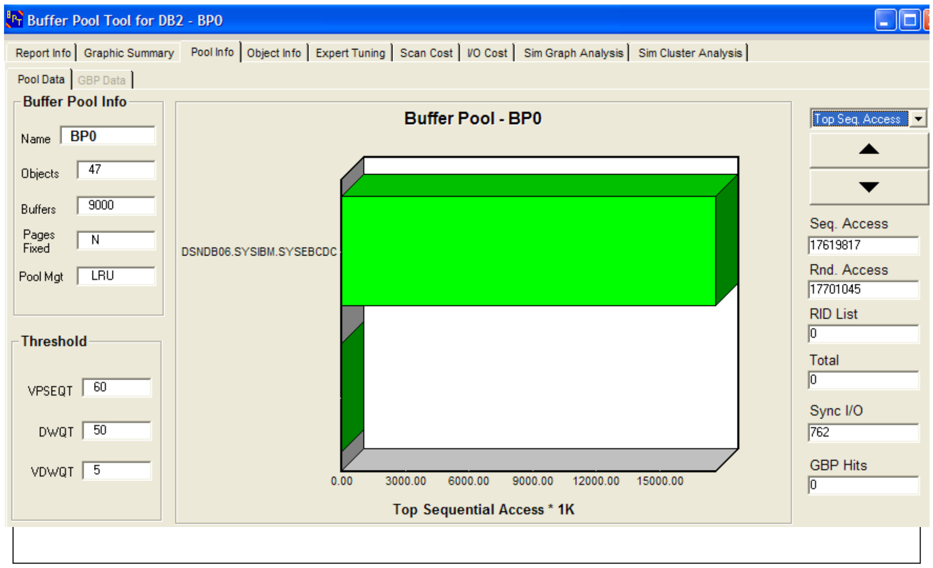
50% Sequential Prefetch



47

This is very unusual access for BP0.

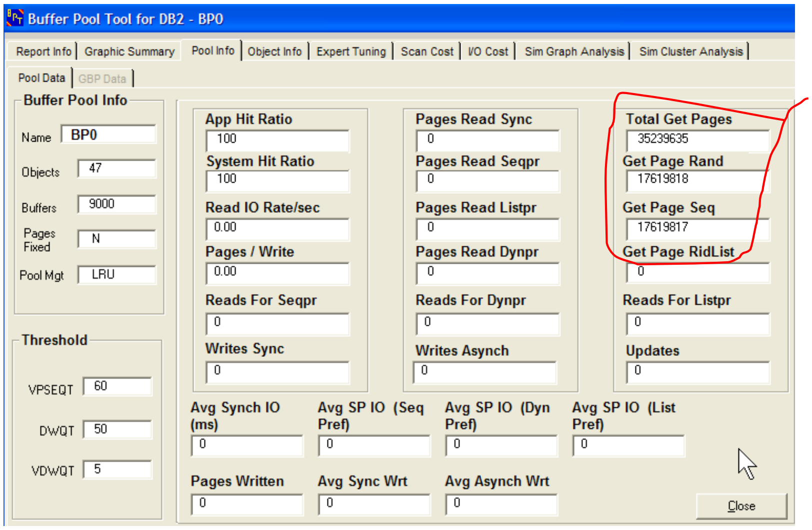
What's this? - using Sysdummy



48

Heavy application usage of SYSDUMMY ….

It’s Pool Resident….



49

This object is “supposedly” one page in size. Updating catalog statistics for it did not get rid of the sequential activity.

Pagefixing Buffer Pools

- Will save 8% of the IO CPU cost
- About a 15-20% reduction of DBM1 CPU cost
- Application savings are harder to measure, but not impossible
- You need *REAL Memory availability* before fixing memory. If the system starts to page, you die…

50

Application saving would have to be measured from the 101 application accounting records, and accumulated over a period. Looking at a few records would not show anything.

Which Buffer Pools would you fix?

Pool	RIO/Sec	Get Pages	Updates	Hit Ratio	I/O	WIO/Sec	Pages/Write	Write I/Os	I
BP0	0.50	1036687	52623	100	328	0.04	19.46	24	
BP1	763.88	8146716	1003697	66.3	479399	19.45	14.66	11906	
BP2	5,035.32	36040067	5143255	91.2	4473817	2,274.84	2.30	1392202	
BP10	0.00	191694	75070	100	0	0.00	0.00	0	
BP8K0	0.61	11974	0	94.4	374	0.00	0.00	0	
BP16K0	0.00	833	0	99.6	3	0.00	0.00	0	

Total Read/Write IO	4,953,921	Total Get Pages	45,427,971
Overall Sys Hit Ratio	86.96	Total I/Os per second	8,094.64
Total Updates	6,274,645	Pages per write	2.41

10 Mins

Pool Data | GBP Data

Buffer Pool Info

Name BP1
Objects 107
Buffers 75000
Pages Fixed N
Pool Mgt LRU

Threshold

VPSEQT 80
DWQT 5
VDWQT 0

Pool Data | GBP Data

Buffer Pool Info

Name BP2
Objects 318
Buffers 280000
Pages Fixed N
Pool Mgt LRU

Threshold

VPSEQT 80
DWQT 5
VDWQT 0

High IO Rate/Second?

IO Intensity? Pages r/w / # Buffers
BP1= 6.4 BP2=15.9

From the performance & capacity planning side, Intensity has to factor in time… and this is missing.

Memory, memory, memory….

51

If you have the memory… the HIGH IO pool gives you the greatest saving, and CPU reduction.

Which Buffer Pools would you fix?

Pool	RIO/Sec	Get Pages	Updates	Hit Ratio	I/O	WIO/Sec	Pages/Write	Write I/Os	Pages Written
BP0	8.26	1285743	6080	96.2	16273	0.78	2.00	1404	2801
BP1	1.31	10936756	6592399	99.9	2981	0.35	24.72	630	15574
BP2	482.76	31988137	210702	81.1	906223	20.70	2.75	37253	102468
BP3	430.81	27041972	268687	81.1	838524	35.04	2.21	63073	139123
BP4	65.61	17374622	7569	95.6	119604	0.84	2.14	1515	3242
BP5	2.41	8797390	9034	99.7	4836	0.27	3.38	490	1657
BP6	21.96	7142422	3616	96.9	40368	0.47	2.47	844	2081
BP7	12.00	3268813	9331	95.2	22660	0.59	2.64	1065	2813
BP8	23.33	4492757	2816	87.4	42206	0.12	5.14	220	1131
BP9	0.24	1130	0	-12.4	426	0.00	0.00	0	0
BP10	2.95	149194	253	93.7	5400	0.05	1.28	97	124
BP11	22.27	1727479	11727	97	43680	2.00	1.84	3601	6626
BP12	198.78	7908916	91289	85.2	394924	20.62	1.73	37117	64127

Total Read/Write IO	3,122,553	Total Get Pages	130,184,720
Overall Sys Hit Ratio	86.32	Total I/Os per second	1,734.75
Total Updates	7,324,265	Pages per write	2.07

30 Mins

Buffer Pool Info

Name	BP2
Objects	535
VP DS Size	64000
HP DS Size	0
Cast Out	Y
Pool Mgt	LRU

Buffer Pool Info

Name	BP12
Objects	109
VP DS Size	12800
HP DS Size	0
Cast Out	Y
Pool Mgt	LRU

High IO Rate/Second?

IO Intensity? Pages r/w / # Buffers

BP2= 14.2 BP12=30.8

not the best indicator

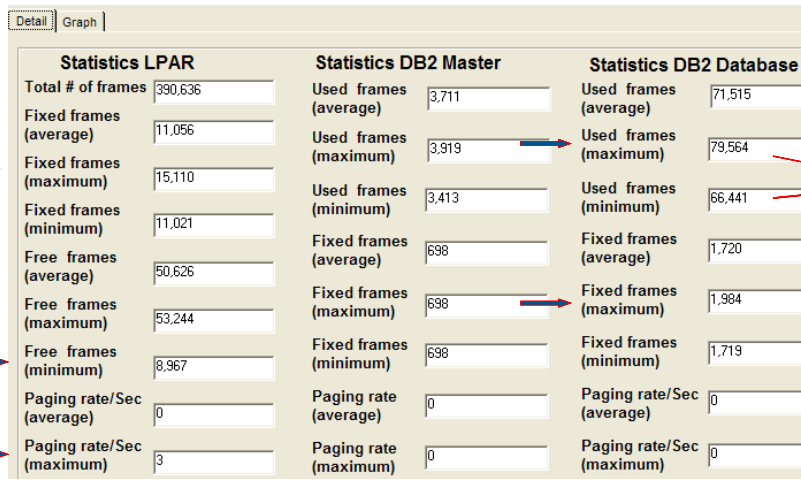
If you have Memory, memory, memory….

52

If you have the memory… the HIGH IO pool gives you the greatest saving, and CPU reduction. If you have the memory, BP2 will give you much better payback than BP12.

So, IO Intensity is NOT the best indicator.

Memory usage – different system



54M

DB2 is not the only user of memory

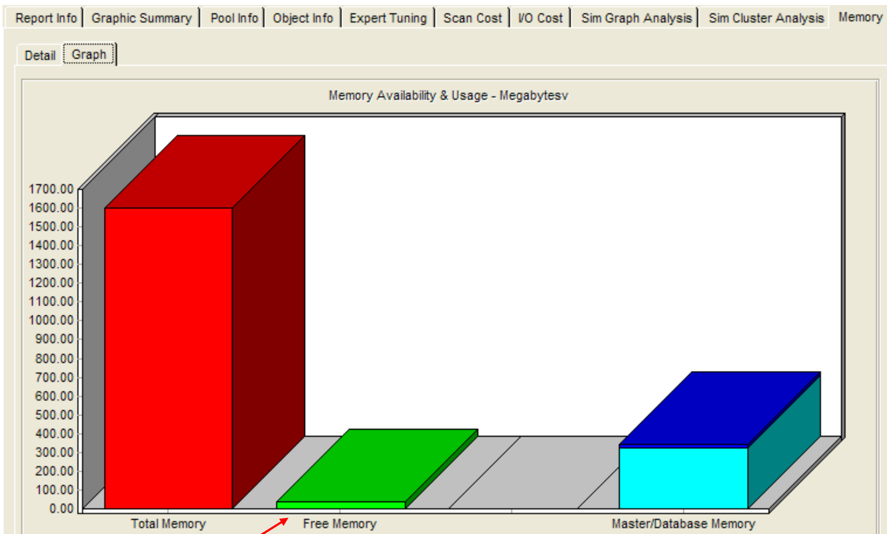
A frame of memory is 4096 bytes

Sometimes we care about the Minimum, sometimes the Maximum

53

If you have the memory… the HIGH IO pool gives you the greatest saving, and CPU reduction. But you don't dare pagefix anything when available memory is this low, and yu already see a count for system paging.

Memory usage – pictures make it obvious



Be careful…

54

If you have the memory… the HIGH IO pool gives you the greatest saving, and CPU reduction. But you don't dare pagefix anything when available memory is this low.

Tracking & comparing performance….

- One of the largest issues we face, is trying to determine if performance and workloads are similar

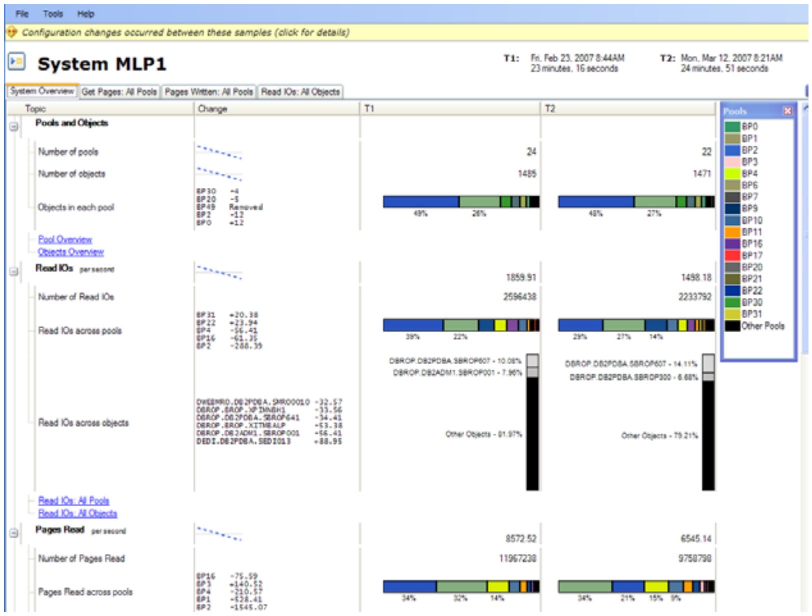
  or

  We know there has been a change in performance, so what changed?

  - *What and where is the difference*?
    - Application workload?
    - System changes?
    - Both?

Some performance differences are obvious, but the reasons may be difficult to find. In most cases, we are looking at performance degradations, and need to find out why the users are complaining about poor response times, or batch jobs are running too long. An automated way of looking at two sets of data, and comparing performance, and highlighting the differences – but at the system level, and at the workload/object usage level would make life much easier.

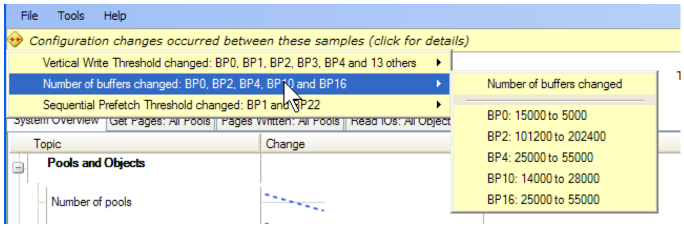
Performance is better – what changed?

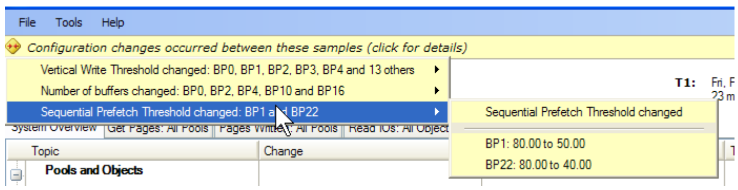


56

We have both configuration and workload changes. Objects not accessed in the second run, a pool not utilized, pool thresholds have been changed.

Performance is better – what changed?





57

This highlights the system configuration changes from the earlier run.

DB2 V8 and V9

- Exploitation of 64bit memory, and moving more control blocks out of the DBM1 address space
- Access to many gigabytes of memory for pools
  - *Does not change basic pool tuning methodologies*
    - Grouping by Random and Sequential, and then by working set size is the industry proven technique (RAMOS, SAMOS)
      - Working set size has no relationship to number of pages for an object (catalog statistics)
        - An object has 1,000,000 pages, but the maximum number in the pool during a specific time period is 2,967
        - The wkset is 2,967
        - This may change, as the pool size is increased or decreased

58

The working set size of an object is the number of pages in the pool at a given point in time. There is no relationship between the working set size, and the information you will find in the DB2 catalog.

DB2 Version 9

- More areas of memory moved above the 2Gig Bar
  - Large parts of the EDM Pool
    - SKCT and SKPT
    - DBDs
    - Parts of CT and PT
    - Parts of Dynamic SQL

You need REAL memory behind everything…

59

A very large memory relief for big systems. Keep in mind that you still need the REAL memory available on the machine!!

DB2 Version 9

- Increasing prefetch quantities and using larger page sizes to reduce IO (vpseqt * #Buffers) > 40,000
  - Double the prefetch and deferred write number of buffers
  - IO remains a huge throughput concern for system scalability
- Supports buffer pool sizes > 5 Gigabytes
  - Up to 1 Terabyte of memory for DB2
- Increased usage of 32K sort file to reduce IO
  - Larger and multiple 32K sort objects
  - Improved sort logic – row < 101 bytes uses 4K *longer uses 32K*
  - It's possible to limit the amount of sort space for a thread (zparm)

60

IO remains a primary concern for system scalability, and these concerns are being addressed in every new version of DB2.

DB2 Version 9

- Automatic Pool Size Management – option autosize=yes
  - Function integrated with WLM
    - Can increase/decrease pool size by 25% of initial size
      - Increments? *Not much real information available yet*
    - Based on long term trends
      - This is not defined anyplace
        - *Long term performance data is a major problem…*
    - Tries to take the memory from other low activity pools first
    - Based on a random hit ratio – *hit ratios are not valid as a performance metric*
  - Seeming fallacy of this approach – *lack of IO prediction capability*
    - We already proved ***bigger is not always better***
  - Will WLM reduce it if the increase does not improve performance?

There are good long term possibilities for this type of approach, and it's obvious that this is just a first cut implementation. It remains to be seen if this provides any real benefit. I suspect it may for small to medium systems at installations where there isn't a lot of DB2 performance knowledge. Large and high performance systems still need real tuning expertise.

Also, remember that the effective way to get good performance is through the proper grouping of objects (Ramos/Samos), and not by throwing memory at few large pools.

DB2 Version 9

- What changes in regard to pool tuning?
  - *Absolutely nothing*
- Methodologies remain the same - Ramos/Samos & Wkset sizes
- There are opportunities for ever larger pools, if you have the memory…. and are sure you get a real benefit
  - Remember, paging is to *DASD*, not expanded memory, so that's 1,000 times slower
    - Just as previous environments, if you start to page, your pool performance may look better statistically, but the users performance is worse… ***be really sure memory is available….***
    - WLM is supposed to manage memory better, and it can, if it's set up properly

62

The basics of performance tuning have not changed over the last four decades, and certainly won't change over the next decade. CPU, Memory, and IO are the important tuning metrics.

DB2 Version 9 – some Important Zparms

- SJMXPOOL - Memory pool for Starjoins, must be enabled by the STARJOIN zparm. Up to 1 Gig of memory, above the 2 Gig bar
- MXTDCACH – in memory data caching, other than Starjoin, allocated per thread
- DSVCI – allows DB2 to create datasets with a CI size that matches the page size, such as 8K, 16K, 32K
- MAXTEMPS – the max workfile storage an agent can use

63

While there are a great many zparm changes in V9, and they are ALL important to your system, these are a few you may want to start with.

DB2 Version 9 – overhead comparison

Table 4-2 Tabulated data for relative CPU % increase for 100 package query application

Accounting Class	CPU%			
	DB2 V8 CL1	DB2 V8 CL2	DB2 V9 CL1	DB2 V9 CL2
1	-	-	-	-
1,2	2.87	-	3.52	-
1,2,3	3.18	0.15	3.96	0.28
1,2,3,7,8	6.96	4.23	5.91	2.43
1,2,3,7,8,10	23.29	21.98	19.82	17.30

The tabulated data shown in Table 4-2 is also shown graphically in Figure 4-13.

Trace filtering capability provides an overhead reduction

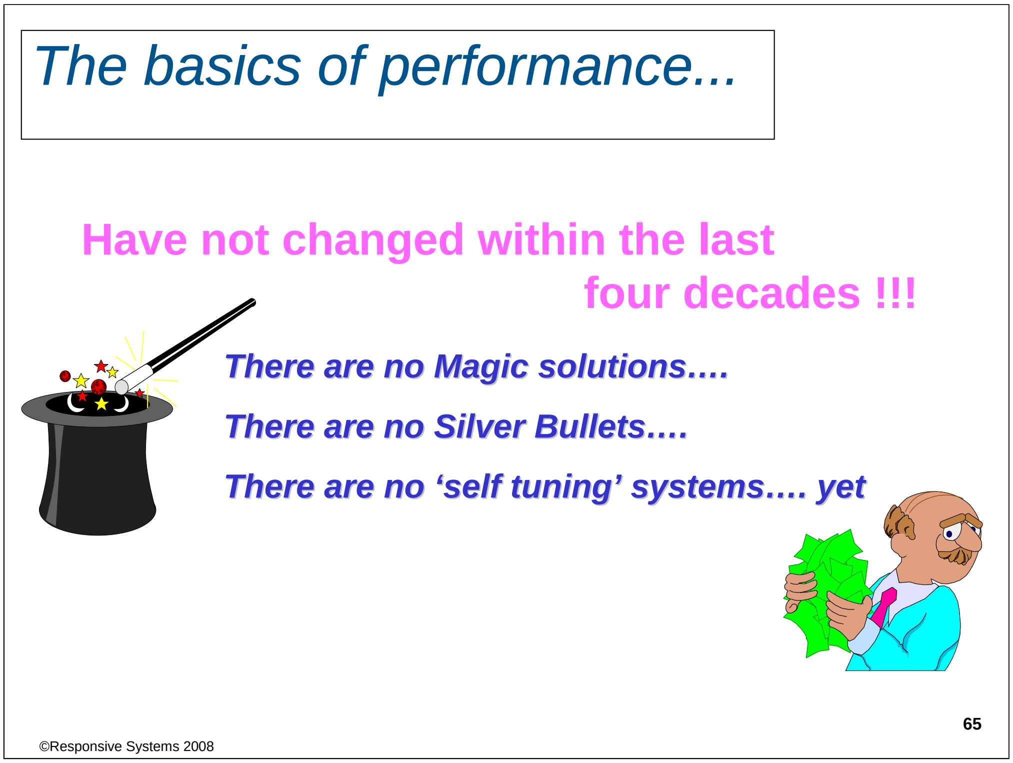
From DB2 9 for z/OS Performance Topics Redbook

64

So – Red is bad, Green is good. The cost of the base C1, C2, C3 accounting traces has gone up. The cost of adding package level information has decreased a lot.

Classes 7, 8, 10 are the high overhead classes, just as V8, but substantially reduced. The traces can have *Include* and *Exclude* lists by many criteria, such as

USERID, WRKSTN, APPNAME, PKGLOC, PKGCOL, PKGPROG, CONNID, CORRID, and ROLE.



Systems and tools are becoming better all the time. While there are some attempts at self tuning systems, or some parts that have a potential for self tuning, we're a long way from realizing these goals. The largest obstacle is the overhead cost of effective approaches. Doing it properly is a lot of work, and very expensive from a CPU perspective.

So – it still requires some work to achieve better performance, and the rewards of lower operational costs that better performance provides.

Session A03

DB2 System Performance, the Basics...
and a Bit, to a Lot More

Joel Goldstein

joel@responsivesystems.com

IDUG 2008 North America