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!!**

The foundations of performance

- **Processor**  
  - CPU speed  
  - Number of engines  
    - Depending upon your workload, more MIPS with fewer engines may degrade performance

- **Memory** – more, more, more… Terabyte is coming...

- **I/O**  
  - Elapsed time – faster devices, more cache, what’s good?  
    - Even 1 Ms can be painful with a high rate/second  
  - CPU cost of I/O  
    - Performance is dependant upon application design & coding

© Responsive Systems 2014
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?
Eyeball method – what looks big or out of proportion to other data?

**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?
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.
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!!*

**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
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.
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”.

**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
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.
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.
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.
Pool Pagefix CPU savings

- Expect about 8% CPU saving of IO cost
  - It's the cost of fixing and releasing the page in memory that causes the CPU cost
- It will vary depending upon your workload
  - A synch read is 1 page
  - A prefetch may actually read anywhere from 1 – 64 pages
  - A write *has to write* every page, but may have multiple IOs
- We don’t have a lot of control over write activity
- Pagefix the pools with high IO rates/second
  
Check memory availability first..

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...
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?
We normally estimate about 8% at the application level.
LRU vs. FIFO Pool Management

- LRU – Least Recently Used
  - Queues have to be maintained for every getpage

- FIFO – First In First Out
  - Eliminating queue management will save CPU
    - if it doesn’t cause the IO rate to increase...

- Can help for:
  - Objects that are pool resident – fit into the pool
  - Objects very large & very random, almost always an IO..
  - Does not occur too often, based on the system data I have seen

One company switched back to LRU from FIFO, and saved 6% CPU

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
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.
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.

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

**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?
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.
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.
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 resolution?
    - 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..
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.
Dataset opens, Prefetch Disabled-No Read Engine.
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)
Responsive Systems – Performance Software that Works!!

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.

When this is less than 10.0, set vdwait=(0.40) or (0.80) or (0.128) depending upon number of objects in the pool... unavailable pool pages....
Note Pages for Prefetch read...

Time -- elapsed time for the statistics data !!!
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.
Drilling down to performance problems…

Responsive Systems – Performance Software that Works!!

© Responsive Systems 2014
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.
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....
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 !!!