Abstract:

Effective pool tuning offers great opportunities for improving both system and application performance. Aside from the elapsed time and cpu savings from I/O elimination, this process often highlights significant application performance problems. This presentation will take you through the steps of analysis and tuning using the industry standard Buffer Pool Tool for DB2 from Responsive Systems, and show you how to analyze and tune your system, and evaluate the payback. Converting I/O savings to elapsed time, $ savings, and improved productivity.

Aside from the elapsed time and cpu savings from I/O elimination, the tuning process often highlights significant application performance problems. See the steps of analysis and tuning using the industry standard Buffer Pool Tool for DB2, how to analyze and tune your system, and evaluate the payback. Converting I/O savings to elapsed time, $ savings, and improved productivity.

Buffer Pool Tool is the only product that can predict the I/O rate per second. This is a measurable metric. Unlike Hit Ratios, it can be converted into CPU costs, and elapsed time savings for both online and batch functions.
Presentation Objectives

- Software Objectives
- Architecture
- Components and Functions
  - Mainframe and Workstation
    - Utilities
- Using Buffer Pool Tool
- Analyzing sets of data
  - Finding the problems and opportunities
- Illustrate before/after tuning performance
- Summaries and the next version – V8.3

The objective of the presentation is to provide an understanding of the Buffer Pool Tool as a tuning product, its components, and capabilities.

The presentation will step through several sets of system data to illustrate the types of data and analysis that BPT provides, and to show how it predicts the effects of tuning changes.
Software Objectives

- Provide the ability to predict the effect of buffer pool changes
  - Simple changes – size & thresholds
  - Moving objects into other existing, or new, pools
  - Predict the I/O rate/sec – the only measurable metric
    - Hit Ratios are interesting, but useless as performance metrics

- Provide a reliable technique for grouping objects into multiple pools
  - Ramos and Samos (and then working set size)
    - Is the Industry Proven technique!!

- Show the performance projections, and let you make the intelligent decisions based upon your resources

The Buffer Pool Tool® set the benchmark for DB2 buffer pool performance analysis and tuning approaches. The objective for the software, when it was designed, was to provide an easy to use, low overhead, reasonably priced product, that can predict the performance effect of changes to buffer pool configurations. It has been proven over the years that several to many pools are necessary to obtain good performance, and the proper grouping of objects into pools based upon access type and working set size yields the best performance. In most cases, using only two or three large pools does not provide the same level of performance and system throughput as multiple pools. The technique used to group objects into multiple pools is Ramos/Samos (random mostly, sequential mostly), and within these groups there are sub-groups based upon working set size. The wkset size is determined from object pool usage, and has no relationship to object statistics in the catalog.
Software Objectives - 2

• Show you how to tune
• Show you the effect of changes
• Show you the rational behind tuning changes
• Show you both system and application performance issues
• Learn how things work – your system & your applications
• Optimize system and application performance, and memory utilization
  – Memory isn’t unlimited yet, and certainly not at most installations
  • Some large client systems are already past 6 Gigabytes of pool allocations

Before the existence of the Buffer Pool Tool, pool tuning, beyond avoiding pool thresholds, was mostly a guessing game.

We had some basic guidelines, such as separating indexes from data - but there was nothing that could predict the effect of changes. This was not acceptable for important business systems.
Responsive Systems – Performance Software that Works!!

**NOT** - Software Objectives

- Be a black box, and present only recommendations
  - You can see the reasons/rationale behind any tuning changes you make – *tuning should be a learning process*

- Change the system dynamically from snapshot statistics
  - All changes to be determined and implemented by the performance analyst
    - By the time active monitoring detects, and reacts, to a significant performance change, the system has already changed again…
    - **Too much overhead to do it properly…**
    - **Not possible to predict from Statistics…. Averages of averages…** you need a buffer manager and IO trace

- Run/Monitor all the time
  - Continuous CPU consumption/overhead
  - Not necessary when tuning peak periods, or problem periods

One of the essential parts of our approach is showing the analyst how all the objects are used, and the effect of changes at varying pool sizes. There are simple changes, such as increasing the pool size, and complex changes such as moving objects into different pools, and showing the performance impacts of the change at varying pool sizes.

We did not want to be a black box – and just present a change scenario. Tuning is a continual learning process – and it’s not possible to factor all these scenarios into software, and keep it current. There is rarely a final solution, as every tuning scenario includes some trade-offs.
Responsive Systems – Performance Software that Works!!

Software Installation

• Receive emailed software as Tersed files
• Upload to the mainframe
• Un-Terse the files
• Edit the JCL
• Bind the BPT Plan to the DB2 Sub-System
• Ready to run….

• About 20 Minutes of effort

KISS…..
Responsive Systems – Performance Software that Works!!

Overall System

The overall Buffer Pool Tool software system.
The individual components are discussed on the following slides.
The Buffer Pool Tool has four components:

1. Collector that attaches to DB2, starts the buffer manager and I/O traces, and writes the data to our own proprietary dataset.

2. Statistical analysis that provides the un-paralleled level of pool and object information.

3. Simulation that predicts the performance of changes. Pool sizes, thresholds, and moving objects into different pools. Predicts hit ratios, wkset sizes, and I/O rate/sec.

4. Workstation component that provides graphic analysis, performance drill down, expert tuning, scan cost impacts, and clustering of wkset sizes for Ramos and Samos objects.

There are several utilities that provide object analysis, connection analysis, and page reference frequencies.
Statistics and Simulation components provide detailed output reports, and small summarized files that are downloaded to the workstation for graphic analysis.
PC files can be Statistics, Simulation, or Consrep that is a combined Statistics and Simulation file. The Consrep job produces the statistics and also performs a base simulation for all pools in use. This reduces the necessity to run initial simulations on all pools and then determine which additional simulations are necessary. You may not need to run simulations for every pool, only those that need tuning, based on high I/O rates.

Downloaded PC files are only a few hundred K in size, versus hundreds of Meg for a collection file.
Collecting Data

• What are your busiest or problem times?
  – This is when you want to collect data
    • At least twice, to determine workload consistency

• Depending upon system size/volume, collector should run up to one hour – really big systems, 5-15 Mins
  – Full Buffer Manager and IO trace
    • Small snapshots, at intervals, lack statistical validity
      – National Bureau of Standards sampling techniques
    – Can collect up to 16 Gigabytes of data

• Low overhead process, 3-5% - only during the collection

Buffer Manager and I/O traces are voluminous. You can not run this type of trace to SMF – you will lose massive amounts of DB2 and other vital system performance data. Using GTF is very high overhead, and GTF will wrap the dataset, invalidating all data. Using efficient assembler language, and discarding unnecessary data from each record keeps overhead at a minimum, and reduces the output dataset size. It’s still very large. Several hundred Meg to 16 Gig, depending on system size and length of collection.
Collected Data… Now what?

- Run Statistics PCSTAT
  - Statistical analysis
  - Reports
  - PC file

- Run Simulation(s) PCSIM Base Simulation
  - Shows performance at varying pool sizes
  - Reports
  - PC file – Wkset sizes, as well as I/O rates
    - Base simulation – one size the same as the current size

- Consolidated Reporting CONSREP
  - Does both of the above automatically
  - Saves having to download multiple files to the workstation

A Base Simulation starts at a slightly smaller pool pool size, and runs up substantially larger.

This shows performance at a smaller size, if the pool is oversized, and shows improvements if memory will reduce I/Os.

The prediction of Working Set Sizes (WKSET) is critical, since this is used to determine object groupings into pools.

The WKSET does not have any relationship to the number of object pages in the DB2 catalog.

Consolidated reporting automatically runs a simulation for each pool, and creates a combined file for PC download. This ranges from a few hundred K to a Meg in size. Compared to hundreds of meg or Gig for a collection file.
Collected Data… *Now what?*

- In most cases we go to the workstation now

- We need the detail print reports in rare cases, since almost all the data is available on the workstation.

- We’ll look at some data and analysis from several systems
  - *There is no absolute analysis path, it all depends on the performance data*

**EYEBALL technique** - Some problem areas, or areas for potential savings will be obvious. Sometimes the solutions are not obvious at all. I see unusual and interesting things in client data several times a year, and they often require a bit of thought, digging around through data, to determine the cause and possible approaches to better performance.
Large System – Buffer Pools     160M GP/hr

BP2 has 2/3 of the entire system I/O rate

This DB2 sub-system issues about 160 Million Getpages per hour. Not the largest I work with, but bigger than most in the industry…
Responsive Systems – Performance Software that Works!!

BP2 – with the high I/O rate

Default graph at pool level

Large pool, mostly sequential access.
How much do you think SP Getpages cost?

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Pool</th>
<th>Sequential Getpages</th>
<th>CPU-Seconds Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 VIEWS</td>
<td>BP2</td>
<td>123456</td>
<td>23.123</td>
</tr>
<tr>
<td>MARIA,MARIA</td>
<td>BP2</td>
<td>234567</td>
<td>34.567</td>
</tr>
<tr>
<td>TAR1.15ESTAB</td>
<td>BP3</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>Dent.12 4&amp;6.3</td>
<td>BP3</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>BTAB1.23 BTAB22</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>KRA1 NUL1</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB123 STAB13</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>BTAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>STAB15 STAB19</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
<tr>
<td>L16P_L1AP</td>
<td>BP2</td>
<td>123456</td>
<td>34.567</td>
</tr>
</tbody>
</table>
This is based on a 2064 processor with 210 MIP engine speeds.
Responsive Systems – Performance Software that Works!!

Same System – 7 Mos. Later   198M GP/hr

Getpage rate is UP 20%         added 2 more pools
IO rate is 1100 per second Lower

This DB2 sub-system now issues about 198 Million Getpages per hour. The I/O rate has dropped 1100 per second.
This is 6 minutes of data.
Over the course of an hour, this system will see 120M+ Getpages.
Four pools simply don’t cut it.…
They used to have good separation of objects – until an external “consultant” told them they didn’t need to do this, and they could get good performance with only two pools by throwing a lot of memory at them.

So their performance went to a very hot place, hand carried in a straw basket.
 Aren’t euphemisms great??
This pool has all the Tablespaces, and 41.5% of the getpages are sequential.
So leaving the vpseqt at the default of 80% is a really good idea…. *NOT!!*
Two heavy sequential objects are killing the pool performance.
Responsive Systems – Performance Software that Works!!

BP20 First heavy sequential

Memory Resident, 20% of pool GP, 1/3 of SP GP

Living in the pool.
More than 1/3 of the pool sequential access, and about 20% of the pool getpages.
Mostly in the pool.
Almost 1/3 of the pool sequential access.
Between these two objects we have 2/3 of the sequential access
BP21 has all the Indexes....
Responsive Systems – Performance Software that Works!!

BP21 Heavy sequential Index

12% of system getpages

BP21 has all the Indexes....
Indexes normally shouldn’t have much sequential access.
Of course there are some exceptions to this – when it’s known, and planned that way.
Responsive Systems – Performance Software that Works!!

BP21 Heavy sequential Index - Resident

This Index is memory resident, but the SP is eating CPU.
Responsive Systems – Performance Software that Works!!

Sequential access costs…. $1M per year

Within six ½ minutes of processing time, the sequential scanning of all objects cost 133 seconds of CPU.

Based on a recent installation’s internal chargeback costs of $.46 a CPU second, this is $61.

Multiplying it out, that’s $600 per hour, and > $5000 per day, more than one million dollars per year.

And this is a company that’s crying about their processor busy rate, and processing costs.

Costs are calculated by using the number of instructions for a getpage request, converting this to MIPS, and then CPU seconds based upon processor speed.
What does a Hit Ratio really tell you?

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

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.
What is a Working Set?

A WKSET is the number of pages in the pool at any point in time...

No relationship to DB2 catalog information

Track and calculate over time, not a momentary snapshot

Snapshots are useless and mis-leading

Has no relationship to the number of pages reflected in the DB2 catalog. As an example, take an object with 1,000,000 pages. The wkset will never be larger than 100,000 and will usually be quite less if it is in a pool with other objects. Wkset is also dependant upon overall pool activity, interactions with other objects, and reference of pages. Unfortunately, it's not a simple number we can easily determine without playing back all pool activity, and tracking the resident pages for all objects across time.
As the pool size increases, the WKSET increases by 125% and 112%.

Other objects may have higher, or smaller, growth percentages. These use more pool resources, but don’t get much I/O relief from the increased pool space.
Objects that got the greatest benefit…

Sometimes not as we expect…

Looking at the objects that show the largest improvements can be surprising. Very often it isn't the objects you thought would get the most benefit.
An original Pool Size, Sequential Objects

We should always look at the wkset size of the SP objects in a pool, to see their impact on the random objects.
Larger Pool Size

- 15,000

Also one fewer object in cluster

How much do they grow if add memory to the pool?
Max Wkset does not grow – 1st Object

<table>
<thead>
<tr>
<th>Pool Size</th>
<th>125000</th>
<th>150000</th>
<th>175000</th>
<th>200000</th>
<th>225000</th>
<th>250000</th>
<th>275000</th>
<th>300000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Pages</td>
<td>1605091</td>
<td>1575664</td>
<td>1553950</td>
<td>1529814</td>
<td>1518617</td>
<td>1514142</td>
<td>1507511</td>
<td>1497335</td>
</tr>
<tr>
<td>Avg Work Set</td>
<td>4638</td>
<td>5462</td>
<td>6281</td>
<td>7105</td>
<td>7544</td>
<td>8008</td>
<td>8455</td>
<td>8773</td>
</tr>
<tr>
<td>Max Work Set</td>
<td>40230</td>
<td>40290</td>
<td>40290</td>
<td>40290</td>
<td>40290</td>
<td>40290</td>
<td>40290</td>
<td>40290</td>
</tr>
<tr>
<td>Sys Mem Hit Ratio</td>
<td>1.3</td>
<td>1.2</td>
<td>1.3</td>
<td>1</td>
<td>57.2</td>
<td>59.1</td>
<td>63.7</td>
<td>67.8</td>
</tr>
<tr>
<td>I/O per second</td>
<td>30.5</td>
<td>30.2</td>
<td>29.8</td>
<td>29.4</td>
<td>13.1</td>
<td>12.5</td>
<td>11</td>
<td>9.8</td>
</tr>
</tbody>
</table>

Payoff

Note the payoff point, where the I/O rate takes a large drop....
Responsive Systems – Performance Software that Works!!

Wkset *does* grow, initially.. – 2nd Object

<table>
<thead>
<tr>
<th>Pool Size</th>
<th>125000</th>
<th>150000</th>
<th>175000</th>
<th>200000</th>
<th>225000</th>
<th>250000</th>
<th>275000</th>
<th>300000</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPage</td>
<td>1631778</td>
<td>1631778</td>
<td>1631778</td>
<td>1631778</td>
<td>1631778</td>
<td>1631778</td>
<td>1631778</td>
<td>1631778</td>
</tr>
<tr>
<td>Avg Work Set</td>
<td>4256</td>
<td>5205</td>
<td>6120</td>
<td>7028</td>
<td>7577</td>
<td>8268</td>
<td>8826</td>
<td>9406</td>
</tr>
<tr>
<td>Max Work Set</td>
<td>25076</td>
<td>30060</td>
<td>35056</td>
<td>39522</td>
<td>43522</td>
<td>48522</td>
<td>53522</td>
<td>58522</td>
</tr>
<tr>
<td>Sys Load Hit Rate</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
<td>23.9</td>
<td>35</td>
<td>41.6</td>
<td>48.6</td>
<td>57.2</td>
</tr>
<tr>
<td>I/O per second</td>
<td>33.3</td>
<td>33.4</td>
<td>33.5</td>
<td>28.2</td>
<td>23.4</td>
<td>21.1</td>
<td>18.8</td>
<td>15.9</td>
</tr>
</tbody>
</table>

Note the payoff point, where the I/O rate takes a large drop....
Responsive Systems – Performance Software that Works!!

Different DB2 System

Overall summary, few pools, two heavy I/O pools. BP3 is 57% of the system I/O
Different System

BP3 with the heaviest I/O. 10% of the access is SP, so lets see which objects are accessed that way.
The objects in BP3, sorted by SP access

It’s easy to see all the heavy hitters that are impacting the random objects.
Correcting some of the application access, will save lot’s of CPU.
Largest contributors to the pool I/O rate

One really heavy hitter…
Metrics for the heaviest I/O object

233 is a very high I/O rate for one object. High Synch I/Os, but lot’s of LP pages read too.
Effect of giving BP3 more memory – 30,000 buffers will save about 100 I/O Sec.
Cluster analysis of the Wksets

Two very large objects, may not belong in this pool

Two very large objects, the vast majority of the objects in the pool are much smaller.
Perhaps they should be moved out of this pool...
Exclude – effect of taking them out of the pool

Taking out two objects saves 260 I/Os per second.
Include into new pool

Component: >>> DB2 Buffer Pool Simulation <<<

Results of Simulation for Buffer Pool...........BP5
Bpool GetP total.....................1,639,783

<table>
<thead>
<tr>
<th>Bpool Size</th>
<th>GetP used</th>
<th>Num. of Hits</th>
<th>ApHit Ratio</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>12,000</td>
<td>1,611,264</td>
<td>1,465,361</td>
<td>91.0 %</td>
<td>00:11:47</td>
</tr>
<tr>
<td>16,000</td>
<td>1,606,878</td>
<td>1,471,613</td>
<td>91.6 %</td>
<td>00:11:44</td>
</tr>
<tr>
<td>20,000</td>
<td>1,601,930</td>
<td>1,474,626</td>
<td>92.1 %</td>
<td>00:11:41</td>
</tr>
<tr>
<td>24,000</td>
<td>1,593,240</td>
<td>1,473,702</td>
<td>92.5 %</td>
<td>00:11:38</td>
</tr>
<tr>
<td>28,000</td>
<td>1,561,755</td>
<td>1,448,138</td>
<td>92.8 %</td>
<td>00:11:32</td>
</tr>
<tr>
<td>32,000</td>
<td>1,549,369</td>
<td>1,441,324</td>
<td>93.1 %</td>
<td>00:11:25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bpool Size</th>
<th>Pages Read</th>
<th>Read I/O</th>
<th>SyHit Ratio</th>
<th>Norm. I/O Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>12,000</td>
<td>370.9 /S</td>
<td>212.4 /S</td>
<td>83.7 %</td>
<td>9.3 %</td>
</tr>
<tr>
<td>16,000</td>
<td>341.7 /S</td>
<td>197.8 /S</td>
<td>85.0 %</td>
<td>8.7 %</td>
</tr>
<tr>
<td>20,000</td>
<td>322.2 /S</td>
<td>187.1 /S</td>
<td>85.9 %</td>
<td>8.2 %</td>
</tr>
<tr>
<td>24,000</td>
<td>305.1 /S</td>
<td>176.6 /S</td>
<td>86.6 %</td>
<td>7.7 %</td>
</tr>
<tr>
<td>28,000</td>
<td>294.0 /S</td>
<td>169.6 /S</td>
<td>87.0 %</td>
<td>7.5 %</td>
</tr>
<tr>
<td>32,000</td>
<td>284.1 /S</td>
<td>163.0 /S</td>
<td>87.4 %</td>
<td>7.2 %</td>
</tr>
</tbody>
</table>

We can put them into a new pool, and have better performance than just making the pool larger.
Almost all random access for BP4
Two dozen analysis graphs….

There are many graph options to help the analysis process.
Responsive Systems – Performance Software that Works!!

The heaviest I/O objects in BP4
Overall summary, few pools, two heavy I/O pools
One large object – perhaps this should be moved out…

One very large random object. Perhaps we should simulate moving it out of this pool.
How many I/Os can we save by making the pool larger?
Put it into a new pool

Results of Simulation for Buffer Pool...........BP6
Bpool GetP total.........................162,392

<table>
<thead>
<tr>
<th>Bpool Size</th>
<th>GetP used</th>
<th>Num. of Hits</th>
<th>ApHit Ratio</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>12,000</td>
<td>139,769</td>
<td>123,354</td>
<td>88.3 %</td>
<td>00:10:19</td>
</tr>
<tr>
<td>14,000</td>
<td>137,578</td>
<td>122,590</td>
<td>89.2 %</td>
<td>00:10:16</td>
</tr>
<tr>
<td>16,000</td>
<td>134,934</td>
<td>121,677</td>
<td>90.2 %</td>
<td>00:10:11</td>
</tr>
<tr>
<td>18,000</td>
<td>134,082</td>
<td>121,914</td>
<td>91.0 %</td>
<td>00:10:09</td>
</tr>
<tr>
<td>20,000</td>
<td>132,577</td>
<td>121,396</td>
<td>91.6 %</td>
<td>00:10:06</td>
</tr>
<tr>
<td>22,000</td>
<td>130,489</td>
<td>120,368</td>
<td>92.3 %</td>
<td>00:10:02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bpool Size</th>
<th>Pages Read</th>
<th>Read I/O</th>
<th>SyHit Ratio</th>
<th>Norm. I/O Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>12,000</td>
<td>94.8 /S</td>
<td>30.6 /S</td>
<td>58.0 %</td>
<td>13.6 %</td>
</tr>
<tr>
<td>14,000</td>
<td>82.0 /S</td>
<td>28.0 /S</td>
<td>63.3 %</td>
<td>12.5 %</td>
</tr>
<tr>
<td>16,000</td>
<td>70.9 /S</td>
<td>25.0 /S</td>
<td>67.9 %</td>
<td>11.3 %</td>
</tr>
<tr>
<td>18,000</td>
<td>62.0 /S</td>
<td>22.8 /S</td>
<td>71.9 %</td>
<td>10.4 %</td>
</tr>
<tr>
<td>20,000</td>
<td>54.5 /S</td>
<td>21.0 /S</td>
<td>75.1 %</td>
<td>9.6 %</td>
</tr>
<tr>
<td>22,000</td>
<td>44.3 /S</td>
<td>19.2 /S</td>
<td>79.6 %</td>
<td>8.9 %</td>
</tr>
</tbody>
</table>

Moving to a new pool provides better overall performance.

Moving the object into a new pool, gives better performance than just making the original pool larger.
Responsive Systems – Performance Software that Works!!

Eliminating I/O Saves Money!!

These I/O rate per sec savings, up through 1,000 per second, have been achieved by clients using Buffer Pool Tool.
Responsive Systems – Performance Software that Works!!

Last large System – Tuning Now….200M GP/Hr

This system is over 200 Million Getpages/Hour, with a very high I/O rate.
Responsive Systems – Performance Software that Works!!

Last large System – after Tuning .. 223M GP/Hr

GP rate is up 10%, IO rate decreased 3,000/Sec

This system is over 223 Million Getpages/Hour, with a greatly reduced, but still high I/O rate.
Buffer Pool Tool V8.3

- New Features and Capabilities
  - Prediction of Group Buffer Pool performance
    - GBP performance itself, at varying sizes, and how the varying sizes impact the local pools on all the members
    - Reports – actual object access and usage across all the members of the group
    - Processes the concurrent activity for Groups up to 32 members
    - Cross system replication factor – simultaneous cache frequency of pages in local member pools
  - Up to a 50% reduction of both elapsed times and CPU for Statistics and Simulations
Buffer Pool Tool V8.3

- New Features and Capabilities - Continued

  - Partition Performance Analysis
    - Statistical analysis at both Object Summary and Partitions
    - Simulation/Prediction of I/O performance at the Object *individual Partition level*

  - Perform up to 16 different pool size simulations in one pass
BPTV8.3  Group Buffer Pool Data

Adding the GBP hit ratio and other data.

Additional data about GBP performance has been added.
BPTV8.3  Group Buffer Pool Object Performance

Shows the Top 10 Objects that had pages retrieved from the GBP

The bar for any object can be double-clicked to drill down to the object performance data.
Several sets of information are available in the new version, and some existing information sets have been expanded.

Note the IO Delay Secs information, both for the collection period, and a 24 hour projection.
Responsive Systems – Performance Software that Works!!

BPTV8.3 expands existing presentation data with 24 hr projections

Projecting a 24 hour CPU cost for Sequential Scan in your system.
BPTV8.3  Partition Analysis – highlights objects

The Object Info tab highlights objects that are partitioned. Selecting an object, then clicking the Partition Data tab takes you the panel that shows you the activity for each individual partition.

All columns on this panel can be widened to see the full object name, and all column headings can be “clicked” to sort data into ascending or descending order.
Access and performance data can easily be obtained at the Partition level.
Tuning your System with the Buffer Pool Tool for DB2

Joel Goldstein

Responsive Systems Company

joel@responsivesystems.com

www.responsivesystems.com

732-972-1261

Based in Morganville, NJ we can be reached through the web, or by calling 732-972-1261