

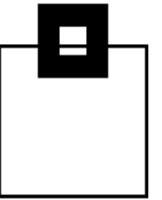
BUFFERPOOL Tuning The Next Generation

Roy Boxwell,
Software Engineering GmbH

P06



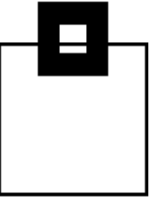
Agenda



- What use are BUFFERPOOLS?
- Is it worth tuning?
- How do you tune them?
- What about GLOBAL BUFFERPOOLS?
- The modern way to visualize BP/GBP problems
- Q&A



Agenda

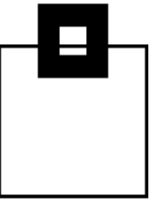
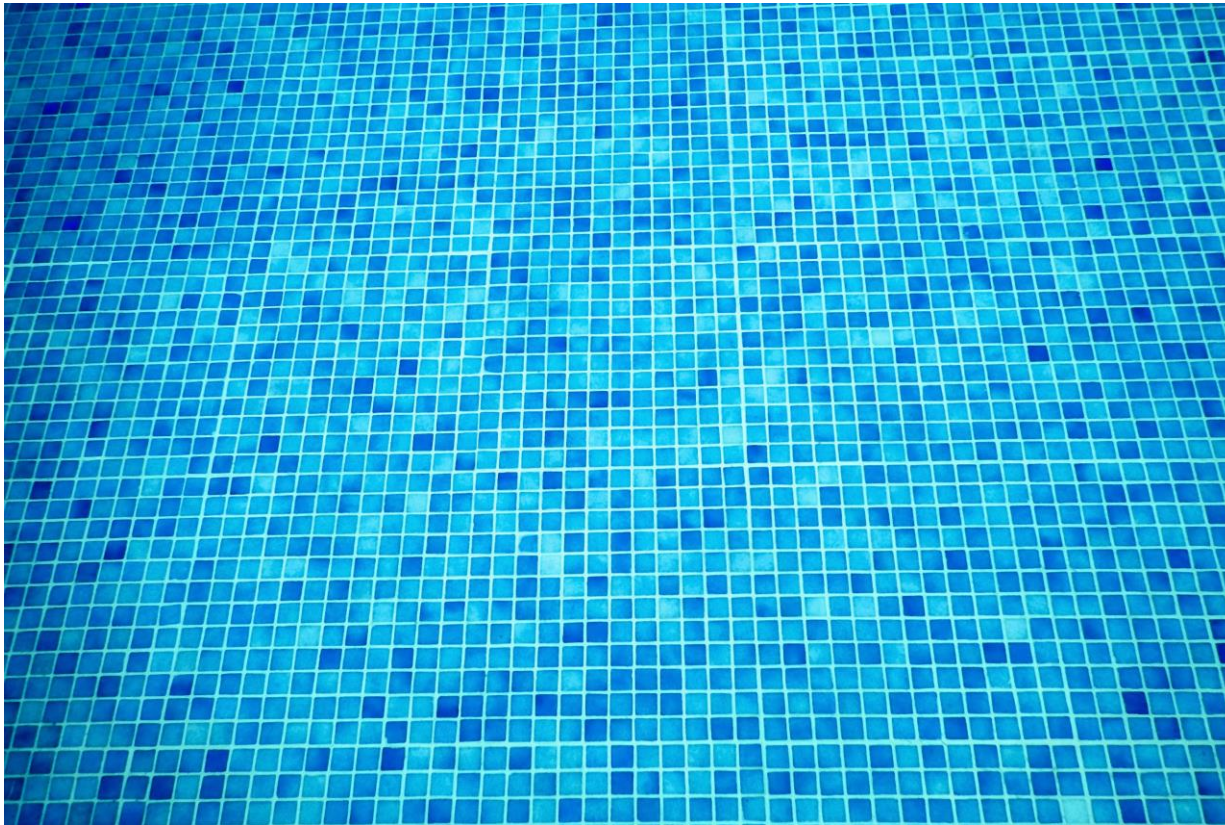


- **What use are BUFFERPOOLS?**
- Is it worth tuning?
- How do you tune them?
- What about GLOBAL BUFFERPOOLS?
- The modern way to visualize BP/GBP problems
- Q&A



What use are BUFFERPOOLS?

Bufferpools:



What use are BUFFERPOOLS?

Real bufferpools (BPs) have existed since the get-go of DB2 (when the B was big!)

The idea, back in the day, was to have two sizes of BP for matching the two sizes of DB2 pages - namely 4KB and 32KB.

DB2 started *very* small - we only had FOUR pools! Three 4KB and a single 32KB:

Buffer Pool Calculation	XA Default	370 Default
Buffers for BP0 ___ * 4K = ___	224 * 4K = 896K	56 * 4K = 224K
Buffers for BP1 + ___ * 4K = ___	+ 0 * 4K = 0K	0 * 4K = 0K
Buffers for BP2 + ___ * 4K = ___	+ 0 * 4K = 0K	0 * 4K = 0K
Buffers for BP32K + ___ * 32K = ___	+12 * 32K = 384K	3 * 32K = 96K
	1280K	320K

Figure 29. Buffer Pool Size Calculation

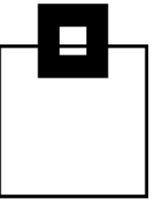
What use are BUFFERPOOLS?

In DB2 V3.1 we got the boost of BP0 to BP49 and BP32K, BP32K1 to BP32K9. BP0 had 2,000 pages as a default and BP32K only 24(!) pages as a default...

In DB2 V6.1 IBM introduced the zero buffers... BP8K0 – BP8K9 and BP16K0 – BP16K9 thus endearing themselves into the heart of all future DBAs!

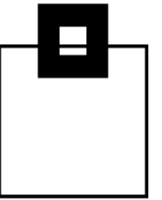
In DB2 V8.1 BP0 default jumped to 20,000, BP8K0 was 1,000, BP16K0 was 500 and BP32K was 250

In Db2 11 the B went lower case...



What use are BUFFERPOOLS?

In fact, the defaults have not changed at all so they must be good and stable, yes?



What use are BUFFERPOOLS?

In fact, the defaults have not changed at all so they must be good and stable, yes?

```
DSNTIP1          INSTALL DB2 - BUFFER POOL SIZES - PANEL 1
====>

Enter 4 KB buffer pool sizes in number of pages.
  1 BP0  ==> 20000      18 BP17 ==> 0          35 BP34 ==> 0
  2 BP1  ==> 0          19 BP18 ==> 0          36 BP35 ==> 0
  3 BP2  ==> 0          20 BP19 ==> 0          37 BP36 ==> 0
  4 BP3  ==> 0          21 BP20 ==> 0          38 BP37 ==> 0
  5 BP4  ==> 0          22 BP21 ==> 0          39 BP38 ==> 0
  6 BP5  ==> 0          23 BP22 ==> 0          40 BP39 ==> 0
  7 BP6  ==> 0          24 BP23 ==> 0          41 BP40 ==> 0
  8 BP7  ==> 0          25 BP24 ==> 0          42 BP41 ==> 0
  9 BP8  ==> 0          26 BP25 ==> 0          43 BP42 ==> 0
 10 BP9  ==> 0          27 BP26 ==> 0          44 BP43 ==> 0
 11 BP10 ==> 0          28 BP27 ==> 0          45 BP44 ==> 0
 12 BP11 ==> 0          29 BP28 ==> 0          46 BP45 ==> 0
 13 BP12 ==> 0          30 BP29 ==> 0          47 BP46 ==> 0
 14 BP13 ==> 0          31 BP30 ==> 0          48 BP47 ==> 0
 15 BP14 ==> 0          32 BP31 ==> 0          49 BP48 ==> 0
 16 BP15 ==> 0          33 BP32 ==> 0          50 BP49 ==> 0
 17 BP16 ==> 0          34 BP33 ==> 0

PRESS:  ENTER to continue  RETURN to exit  HELP for more information
```


What use are BUFFERPOOLS?

In fact, the defaults have not changed at all so they must be good and stable, yes?

```
DSNTIP1          INSTALL DB2 - BUFFER POOL SIZES - PANEL 1
====>

Enter 4 KB buffer pool sizes in number of pages.
 1 BP0  ==> 20000
 2 BP1  ==> 0
 3 BP2  ==> 0
 4 BP3  ==> 0
 5 BP4  ==> 0
 6 BP5  ==> 0
 7 BP6  ==> 0
 8 BP7  ==> 0
 9 BP8  ==> 0
10 BP9  ==> 0
11 BP10 ==> 0
12 BP11 ==> 0
13 BP12 ==> 0
14 BP13 ==> 0
15 BP14 ==> 0
16 BP15 ==> 0
17 BP16 ==> 0

PRESS: ENTER to continue

DSNTIP2          INSTALL DB2 - BUFFER POOL SIZES - PANEL 2
====>

Enter 8 KB, 16KB, and 32 KB buffer pool sizes in number of pages.
 1 BP8K0 ==> 2000      11 BP16K0 ==> 500      21 BP32K  ==> 250
 2 BP8K1 ==> 0         12 BP16K1 ==> 0         22 BP32K1 ==> 0
 3 BP8K2 ==> 0         13 BP16K2 ==> 0         23 BP32K2 ==> 0
 4 BP8K3 ==> 0         14 BP16K3 ==> 0         24 BP32K3 ==> 0
 5 BP8K4 ==> 0         15 BP16K4 ==> 0         25 BP32K4 ==> 0
 6 BP8K5 ==> 0         16 BP16K5 ==> 0         26 BP32K5 ==> 0
 7 BP8K6 ==> 0         17 BP16K6 ==> 0         27 BP32K6 ==> 0
 8 BP8K7 ==> 0         18 BP16K7 ==> 0         28 BP32K7 ==> 0
 9 BP8K8 ==> 0         19 BP16K8 ==> 0         29 BP32K8 ==> 0
10 BP8K9 ==> 0         20 BP16K9 ==> 0         30 BP32K9 ==> 0

31 DEFAULT 4-KB BUFFER POOL FOR USER DATA ==> BP1      BP0      - BP49
32 DEFAULT 8-KB BUFFER POOL FOR USER DATA ==> BP8K0     BP8K0    - BP8K9
33 DEFAULT 16-KB BUFFER POOL FOR USER DATA ==> BP16K0    BP16K0   - BP16K9
34 DEFAULT 32-KB BUFFER POOL FOR USER DATA ==> BP32K      BP32K    - BP32K9
35 DEFAULT BUFFER POOL FOR USER LOB DATA  ==> BP0        BP0      - BP32K9
36 DEFAULT BUFFER POOL FOR USER XML DATA  ==> BP16K0    BP16K0   - BP16K9
37 DEFAULT BUFFER POOL FOR USER INDEXES    ==> BP0        BP0      - BP32K9

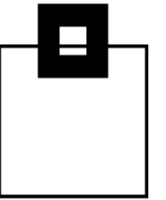
PRESS: ENTER to continue RETURN to exit HELP for more information
```

What use are BUFFERPOOLS?

Ok, so we have a nice set of default sizes but what are they used for?

Well, even today on our “fake” DASD there is a nasty thing called I/O and I/O is slow!

The best I/O does not cause a disk seek at all, in fact that is the entire point of a BP. A piece of required data is found in memory so that the data is instantly available to the application process.



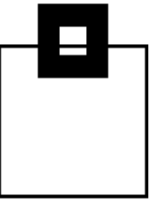
What use are BUFFERPOOLS?

Why do we have so many BPs in Db2 for z/OS? Why not just one huge area of RAM stuffed full of data?

Well, the answer to that is “Horses for Courses”

The performance of any given BP is strongly related to the applications running and using it. Think of a process that is sequentially reading through a table for summation purposes. It reads data but will **never want to read it again.**

Is this “good” for the BP?



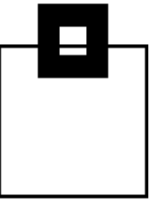
What use are BUFFERPOOLS?

Why do we have so many BPs in Db2 for z/OS? Why not just one huge area of RAM stuffed full of data?

Well, the answer to that is “Horses for Courses”

The performance of any given BP is strongly related to the applications running and using it. Think of a process that is sequentially reading through a table for summation purposes. It reads data but will **never want to read it again.**

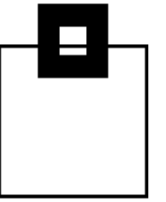
Is this “good” for the BP? Nope.



What use are BUFFERPOOLS?

Now imagine an application process that is randomly reading data through an index. It fetches the leaf and non-leaf pages into the BP as it needs them and then carries on.

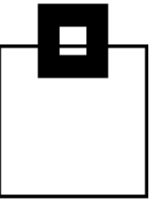
When it needs that “used” leaf page it will find it again in the BP.
Is this a “good” use of the BP?



What use are BUFFERPOOLS?

Now imagine an application process that is randomly reading data through an index. It fetches the leaf and non-leaf pages into the BP as it needs them and then carries on.

When it needs that “used” leaf page it will find it again in the BP. Is this a “good” use of the BP? Yes.

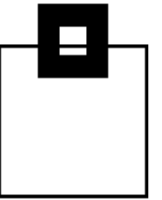


What use are BUFFERPOOLS?

Now imagine an application process that is randomly reading data through an index. It fetches the leaf and non-leaf pages into the BP as it needs them and then carries on.

When it needs that “used” leaf page it will find it again in the BP. Is this a “good” use of the BP? Yes.

If both of these applications share the same BP this is obviously not good, but this is what most, if not all, Db2 shops do!

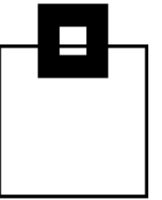


What use are BUFFERPOOLS?

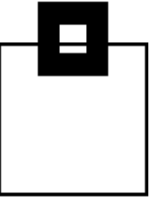
Now imagine a sort – Yes I am talking about DSNDB07 usage here! You might not know it but SORT requires a BP as well. What are the odds of a repeated reread in a sort pool?

You can imagine they are pretty low!

Sort should **always** be in its own little/large pool with VPSEQT set to 99%



What use are BUFFERPOOLS?



What about LRU, FIFO, NONE when stealing pages?

“It depends” raises its ugly head here!

LRU is least recently used - so that the “stalest” pages can be got rid off and replaced with newer ones and is, naturally, the default.

NONE is great for “permanent in-memory data” Xref tables etc.

FIFO is great if you really do not care about the least recently used logic. It is really a niche use case though.

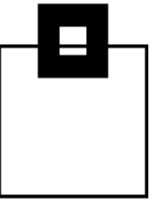


What use are BUFFERPOOLS?

Most shops have no personnel to look at, monitor, tune and change the BPs at their site. The expertise is “graying” and lots of people are “afraid” of changing something so crucial as a BP.

This is not healthy!

BP usage always changes over time! (Death by cut-and-paste is a classic...)

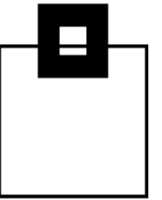


What use are BUFFERPOOLS?

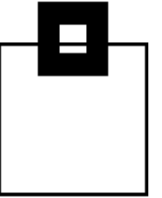
The Db2 Directory and Catalog are the most important part of any Db2 system. They are the meta-data repositories of the entire system and contain **everything** you need to run, but these objects go into only the listed default BPs and I will bet that most shops also use these BPs for application data – This I call BP Pollution.

The first thing you must do is move all non-Db2 objects **out** of BP0, BP8K0, BP16K0 and BP32K.

This can be tough, but it must be done! It is useless starting to tune BPs when their usage is completely broken!



What use are BUFFERPOOLS?

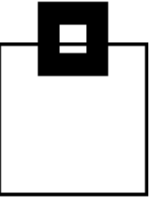


A standard set of “Rules of Thumb” is:

- 1) The Db2 Directory and Catalog on their own
- 2) Application tablespaces and indexes kept apart
- 3) LOB and XML on their own
- 4) Sort on its own
- 5) In-memory tables (PGSTEAL(NONE) on their own)
- 6) Randomly accessed data on their own
- 7) Sequentially accessed data on their own
- 8) GBP correctly sized
- 9) The rest...



Agenda

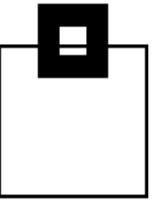


- What use are BUFFERPOOLS?
- Is it worth tuning?
- How do you tune them?
- What about GLOBAL BUFFERPOOLS?
- The modern way to visualize BP/GBP problems
- Q&A



Is it worth tuning?

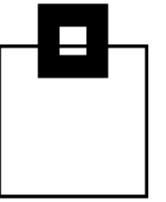
The answer is a massive



Is it worth tuning?

The answer is a massive

YES!



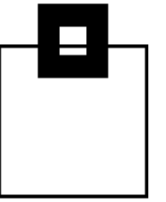
Is it worth tuning?

It is often written that the quickest and best results of any type of tuning are indeed with BP and GBPs.

SQL Tuning is still, obviously, required but the big system wide ROI can be had in the BP and GBP area.

IBM state that “no brainer” options such as PGFIX(YES) give up to 8% cpu savings. That is at the *system* level!

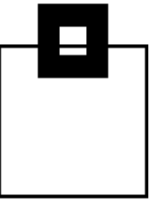
Using large frame sizes (1Mb or 2GB) saves up to 4%



Is it worth tuning?

If you can imagine a system where it is actively paging, actively reading and rereading data and index pages all the time just because it cannot find the data in the BP it is clear that you can

“Tune the SQL until you die, it will not get faster!”



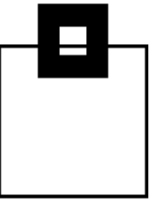
Is it worth tuning?

Another very popular problem is the:

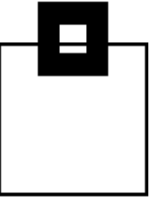
“Stuff it in BP8K0, I know that BP exists!”

This is especially popular for COMPRESS YES indexes – Thus solving one problem and introducing another, even worse, problem at the same time!

Naturally, the BP problem is not seen and everyone wonders why performance tanks every now and again...



Agenda

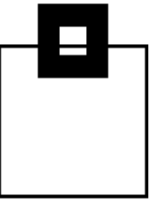


- What use are BUFFERPOOLS?
- Is it worth tuning?
- **How do you tune them?**
- What about GLOBAL BUFFERPOOLS?
- The modern way to visualize BP/GBP problems
- Q&A



How do you tune them?

There is an ALTER command...

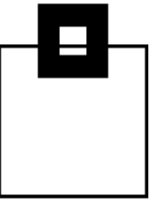


How do you tune them?

There is an ALTER command...

Now in Db2 for LUW you would just throw a few more CPUs or GBs of RAM at the problem and walk away a happy bunny.

On z/OS it is a little bit more tricky but I know of shops that have basically done the same!



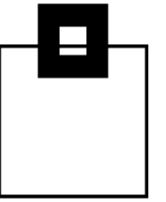
How do you tune them?

There is an ALTER command...

Now in Db2 for LUW you would just throw a few more CPUs or GBs of RAM at the problem and walk away a happy bunny.

On z/OS it is a little bit more tricky but I know of shops that have basically done the same!

You must find the current state of your BPs.



How do you tune them?

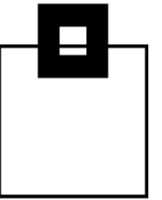
There is an ALTER command...

Now in Db2 for LUW you would just throw a few more CPUs or GBs of RAM at the problem and walk away a happy bunny.

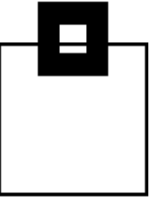
On z/OS it is a little bit more tricky but I know of shops that have basically done the same!

You must find the current state of your BPs.

You must then ALTER them to do what they should be doing!



Agenda



- What use are BUFFERPOOLS?
- Is it worth tuning?
- How do you tune them?
- **What about GLOBAL BUFFERPOOLS?**
- The modern way to visualize BP/GBP problems
- Q&A

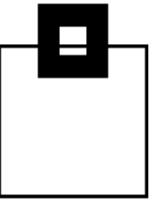


What about GLOBAL BUFFERPOOLS?

These are the forgotten “zombies” of the Db2 for z/OS BP world!

They are incredibly important for the well-running of any data-sharing system and basically require the same style of tuning as normal – local – BPs.

They have sizes and thresholds just like local BPs but they have normally been completely forgotten about in most shops!



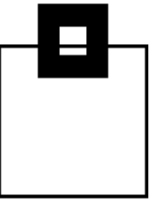
What about GLOBAL BUFFERPOOLS?

These are the forgotten “zombies” of the Db2 for z/OS BP world!

They are incredibly important for the well-running of any data-sharing system and basically require the same style of tuning as normal – local – BPs.

They have sizes and thresholds just like local BPs but they have normally been completely forgotten about in most shops!

You must find the current state of your Global BPs.



What about GLOBAL BUFFERPOOLS?

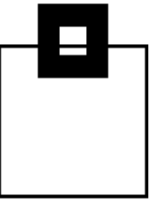
These are the forgotten “zombies” of the Db2 for z/OS BP world!

They are incredibly important for the well-running of any data-sharing system and basically require the same style of tuning as normal – local – BPs.

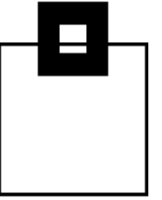
They have sizes and thresholds just like local BPs but they have normally been completely forgotten about in most shops!

You must find the current state of your Global BPs.

You must then ALTER them to do what they should be doing!



Agenda



- What use are BUFFERPOOLS?
- Is it worth tuning?
- How do you tune them?
- What about GLOBAL BUFFERPOOLS?
- **The modern way to visualize BP/GBP problems**
- Q&A

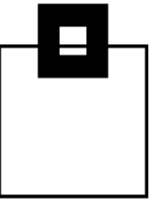


The modern way to visualize BP/GBP problems

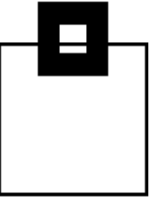
We now know that we have a problem!

How can we actually visualize this and do stuff?

The archaic way that 3270 green screen outputs data and/or the extremely detailed obscure formulae that must be used and/or the different sources of data that must be trawled, all make it “non trivial”...



The modern way to visualize BP/GBP problems



We now know that we have a problem!

How can we actually visualize this and do stuff?

The archaic way that 3270 green screen outputs data and/or the extremely detailed obscure formulae that must be used and/or the different sources of data that must be trawled, all make it “non trivial”...

How “non trivial”???



The modern way to visualize BP/GBP problems

You have to get the correct info from 100's of metrics to then calculate and check all the performance-relevant thresholds:

Frame boundary

Frame sizing

Frame size with LFAREA

System residency

Random residency

Sequential residency

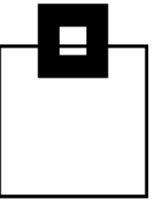
PGFIX(NO) used

Page-ins for read required

Page-ins for write required

DMTH Threshold hit

Prefetch disabled



The modern way to visualize BP/GBP problems

You have to get the correct info from 100's of metrics to then calculate and check all the performance-relevant thresholds:

Random Sync I/O

Bufferpool paging

System Hit Ratio

Application Hit Ratio

No. of page updates for each page written

No. of pages written for each write I/O

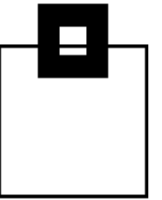
Page arrival rate

Prefetch size

VPSEQT should be changed

DWQT hit rate / Second

VDWQT hit rate / Second



The modern way to visualize BP/GBP problems

You have to get the correct info from 100's of metrics to then calculate and check all the performance-relevant thresholds:

GBP Hit ratio

GBP Writes failed

GBP Reclaims for directory entries

GBP Cross Invalidations (XI) due to directory reclaims

GBP Snapshot

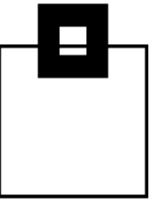
GBP Castout

GBP Sizing and Duplexing

GBP Sync read XI miss ratio – high

GBP Sync read XI data not returned per day – high

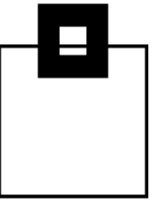
GBP Sync read XI data not returned per second – high



The modern way to visualize BP/GBP problems

Not only that, but you have to know which threshold and/or which value to ALTER to actually do the corrective action!

This is all pretty nasty work that someone probably did way back in the 1990's but since then it has not been updated...

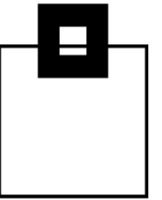


The modern way to visualize BP/GBP problems

Not only that, but you have to know which threshold and/or which value to ALTER to actually do the corrective action!

This is all pretty nasty work that someone probably did way back in the 1990's but since then it has not been updated...

Let WorkloadExpert for Db2 z/OS (WLX) do the heavy lifting!



The modern way to visualize BP/GBP problems

SQL WorkloadExpert for Db2 z/OS

Selected profile: SC10-IQA0610 - IQA0610 - Z100SC10 - 192.168.9.98 - 5125

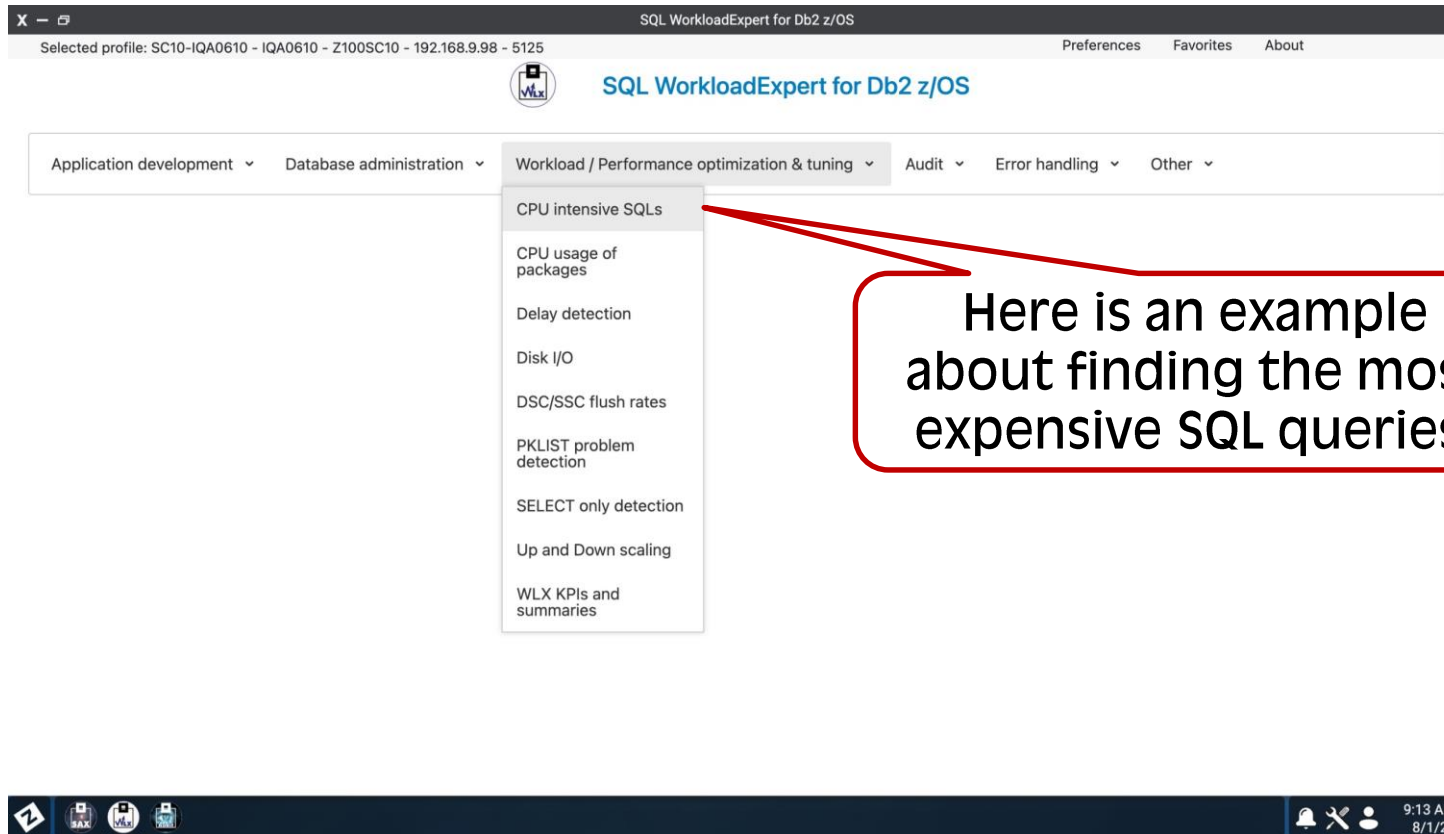
SQL WorkloadExpert for Db2 z/OS

Application development ▾ Database administration ▾ Workload / Performance optimization & tuning ▾ Audit ▾ Error handling ▾ Other ▾

WLEX comes with more than 70 prepared Use Cases, categorized into six areas of interest for quick and easy workload analysis scenarios.

9:12 AM 8/1/23

The modern way to visualize BP/GBP problems



The modern way to visualize BP/GBP problems

SQL WorkloadExpert for Db2 z/OS
Selected profile: DD10SEC-IQA061QB - IQA061QB - Z100DD10 - 192.168.9.98 - 15151



SQL WorkloadExpert for Db2 z/OS

Application Workload

Projection Selection Sorting Preferences Favorites

Label	Description	Add
PROGRAM	Package	>
PACKAGE_COLLID	Collection ID	>
STMT_ORIGIN	Statement Origin	>
NUMBER_OF_STATEMENTS	Number of Statements	>
TOTAL_CPU_TIME	Sum of CPU Time	>
HIGHEST_CPU_TIME	Highest CPU Time	>
TOTAL_ELAPSED_TIME	Sum of Elapsed Time	>
HIGHEST_ELAPSED_TIME	Highest Elapsed Time	>
TOTAL_GETPAGES	Sum of GETPAGES	>
HIGHEST_GETPAGES	Highest GETPAGES	>
TOTAL_SYNC_BUFFER_READS	Sum of Synchronous Buffer Reads	>
TOTAL_SYNC_BUFFER_WRITES	Sum of Synchronous Buffer Writes	>
TOTAL_ROWS_EXAMINED	Sum of Rows examined	>
TOTAL_ROWS_PROCESSED	Sum of Rows processed	>
TOTAL_INDEX_SCANS	Sum of Index scans	>

Label	Description	Remove
PRIM_AUTHOR	Primary Authorization ID	x
TOTAL_EXECUTIONS	Sum of Executions	x
AVERAGE_CPU_TIME	Average CPU Time	x

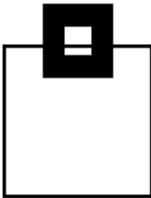
Each Use Case guides you through customization to fit exactly your needs

First we select the data we want to report (Projection)...



9:42 AM
8/1/23

The modern way to visualize BP/GBP problems



Application Workload

- Projection
- Selection**
- Sorting
- Preferences
- Favorites

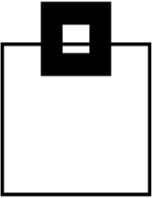
Available Columns		
Label	Description	Add
AVERAGE_CPU_TIME	Average CPU Time	>
AVERAGE_ELAPSED_TIME	Average Elapsed Time	>
AVERAGE_GETPAGES	Average GETPAGES	>
PACKAGE_COLLID	Collection ID	>
CPU_TIME	CPU Time	>
CURRENT_DEGREE_SW	Current Degree special register	>
CUR_PRECISION_SW	Current Precision special register	>
CURRENT_RULES_SW	Current Rules special register	>
CUR_SQLID	Current SQL ID	>
CURRENT_DATA_SW	CURRENTDATA option	>
CURSOR_HOLD_SW	Cursor prepared with hold indicator	>
DYNAMIC_RULE_SW	DYNAMICRULES option	>
ELAPSE_TIME	Elapsed Time	>
END_USERID	End User ID	>
EXECUTIONS	Executions	>

Selected Columns				
Label	Operation	Value	Description	Remove
WLX_TIMESTAMP	=	newest	WLX Key	x
STMT_ORIGIN	=	D	Statement Origin	x

... then you can apply some filtering for the result set of interest (Selection) ...



The modern way to visualize BP/GBP problems



SQL WorkloadExpert for Db2 z/OS
Selected profile: DD10SEC-IQA061QB - IQA061QB - Z100DD10 - 192.168.9.98 - 15151 Preferences Favorites About



SQL WorkloadExpert for Db2 z/OS

Application Workload

Projection Selection **Sorting** Preferences Favorites

Available Columns		
Label	Description	Add
AVERAGE_ELAPSED_TIME	Average Elapsed Time	>
AVERAGE_GETPAGES	Average GETPAGES	>
PACKAGE_COLLID	Collection ID	>
HIGHEST_CPU_TIME	Highest CPU Time	>
HIGHEST_ELAPSED_TIME	Highest Elapsed Time	>
HIGHEST_GETPAGES	Highest GETPAGES	>
NUMBER_OF_STATEMENTS	Number of Statements	>
PROGRAM	Package	>
PRIM_AUTHOR	Primary Authorization ID	>
STMT_ORIGIN	Statement Origin	>
TOTAL_CPU_COST	Sum of CPU costs	>
TOTAL_CPU_TIME	Sum of CPU Time	>
TOTAL_ELAPSED_TIME	Sum of Elapsed Time	>
TOTAL_EXECUTIONS	Sum of Executions	>
TOTAL_GETPAGES	Sum of GETPAGES	>

Selected Columns			
Label	Description	Direction	Remove
AVERAGE_CPU_TIME	Average CPU Time	DE... ▾	✕

... and you can set the order of the result set (Sorting).

Any of this can be kept and/or shared for later execution, or even for automatic reporting.

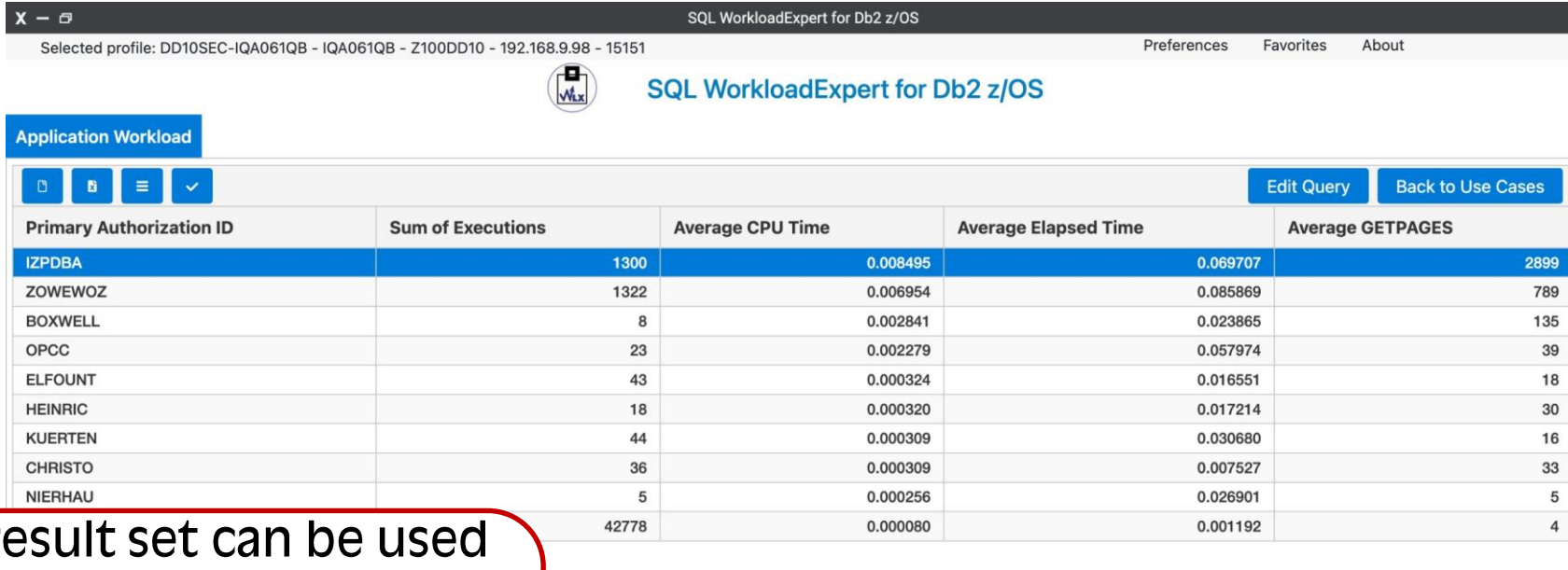
Cancel

OK



9:42 AM
8/1/23

The modern way to visualize BP/GBP problems



The screenshot displays the SQL WorkloadExpert for Db2 z/OS interface. The top bar shows the application name and a selected profile: DD10SEC-IQA061QB - IQA061QB - Z100DD10 - 192.168.9.98 - 15151. Below the title bar, the 'Application Workload' section is active, showing a table with columns for Primary Authorization ID, Sum of Executions, Average CPU Time, Average Elapsed Time, and Average GETPAGES. The table lists various authorization IDs and their corresponding performance metrics.

Primary Authorization ID	Sum of Executions	Average CPU Time	Average Elapsed Time	Average GETPAGES
IZPDBA	1300	0.008495	0.069707	2899
ZOWEWOZ	1322	0.006954	0.085869	789
BOXWELL	8	0.002841	0.023865	135
OPCC	23	0.002279	0.057974	39
ELFOUNT	43	0.000324	0.016551	18
HEINRIC	18	0.000320	0.017214	30
KUERTEN	44	0.000309	0.030680	16
CHRISTO	36	0.000309	0.007527	33
NIERHAU	5	0.000256	0.026901	5
	42778	0.000080	0.001192	4

The result set can be used for further in-depth drill down analysis, cross-reference reporting, exporting into a pdf/excel, or ...



The modern way to visualize BP/GBP problems

The screenshot shows the SQL WorkloadExpert for Db2 z/OS application. A 'Create Report' dialog box is open, allowing the user to configure a report. The dialog includes fields for Profile (set to 'No Profile Saved'), Category Column (set to 'Primary Authorization ID'), and Value Columns (set to 'Sum of CPU Time, Sum of Ela...'). The 'Chart Type' section has radio buttons for Horizontal bar, Vertical bar, Pie, Line, and Radar, with 'Radar' selected. A 'Create Report' button is at the bottom right of the dialog.

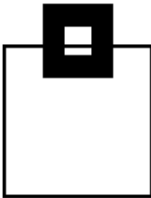
In the background, a table displays data for 'Primary Authorization ID' and 'Sum of Rows processed'. The table has two columns: 'Primary Authorization ID' and 'Sum of Rows processed'. The data rows are as follows:

Primary Authorization ID	Sum of Rows processed
IZPDBA	2636900
	118815
	287297
	943
	990
	83
	159
	340
	149
	0

... to generate a graphical report of many kinds (bars, pies, lines, or radar charts ...)



The modern way to visualize BP/GBP problems



Space AssuranceExpert for Db2 z/OS

SC10-MVNXTEST - MVNXTEST - Z100SC10 - s0w1.fritz.box -5125

Preferences Support About

Preferences

Profile: SC10-MVNXTEST

Profile Details

Location	Schema	Host	Port
Z100SC10	MVNXTEST	s0w1.fritz	5125

Lines Limit: 300

Timeout: 60

Locale: EN

+ Import Export

< Back to Use Cases

8:51 AM 8/1/23

This time I'd like to run my analysis against a different system. All Db2 systems defined can easily be chosen from the App's preferences.



The modern way to visualize BP/GBP problems

The screenshot displays the SQL WorkloadExpert for Db2 z/OS interface. The top navigation bar includes 'Bufferpool overview', 'BP details', 'New BP', and 'BP history'. The 'Current status' section (last extract: 2023/07/11 10:00am) shows:

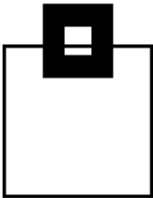
- 34 OK – no violations
- 7 warnings
- 2 critical** (highlighted in red)

The critical items are:

- BP8K0** (highlighted in red): Prefetch 96%, Datamanagement 96%
- BP32K0** (highlighted in red)

The 'Summary' section (last extract: 2023/07/11 10:00am) features an 'Overall allocation' sunburst chart. The chart is divided into segments representing different buffer pools (BP) and tablespaces (TS). The segments are color-coded: yellow (BP32K0, 32K), orange (BP8K0, 8K), blue (BP16K0, 16K; BP8K1, 8K; BP0, 4K; BP1, 4K; BP2, 4K; BP3, 4K), and grey (TS). The chart shows the hierarchical allocation of buffer pools across various tablespaces.

All warnings and critical problems are highlighted.



The modern way to visualize BP/GBP problems

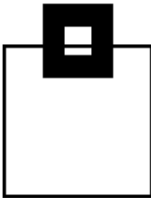
The screenshot displays the SQL WorkloadExpert for Db2 z/OS interface. The top navigation bar includes 'Bufferpool overview', 'BP details', 'New BP', and 'BP history'. The 'Current status' section (last extract: 2023/07/11 10:00am) shows:

- 34 OK – no violations
- 7 warnings
- 2 critical: BP8K0 and BP32K0. BP8K0 has Prefetch 96% and Datamanagement 96%.

The 'Summary' section (last extract: 2023/07/11 10:00am) features a sunburst chart titled 'Overall allocation' showing the distribution of buffer pools across tablespaces (TS). The chart is divided into segments for BP32K, BP8K, BP16K, BP8K1, BP1, BP2, BP0, and BP8K0, with their respective sizes and associated tablespaces.

A top level overview is available.

The modern way to visualize BP/GBP problems



SQL WorkloadExpert for Db2 z/OS

Selected profile: DD10SEC-IQA061QB - IQA061QB - Z100DD10 - 192.168.9.98 - 15151

Preferences Favorites About

BP overview **Bufferpool details** New BP BP history

BP0 ✓ BP1 ✓ BP2 ✓ **BP8K0 !** BP8K1 ⚠ BP16K0 ✓ BP32K0 !

Bufferpool BP8K0 details:

(Last Extract: 2023/07/11 10:00am)

size:	1234567	?
hitratio:	88	?
application:	77	?
system:	99	?
page residency:	333	?
number page updates/page:	987	?
number page writes/IO:	654	?
total page IO rate:	3210	?
synchronous IO rate:	7879	?

Bufferpool Advisor:

BP too small, add 11% more storage

Hitratio should be > 90%

Application hitratio should be > 90%

Page residency should be > 600

I/O rate should be < 1000

→ Use SIMULATE to apply and simulate the changes then verify, using the BP history.

RESIZE DETAILS CHANGE OBJECT LIST SIMULATE VPSEQT

12:53 PM 7/11/23



The modern way to visualize BP/GBP problems

SQL WorkloadExpert for Db2 z/OS

Selected profile: DD10SEC-IQA061QB - IQA061QB - Z100DD10 - 192.168.9.98 - 15151

Preferences Favorites About

BP overview **Bufferpool details** New BP BP history

BP0 ✓ BP1 ✓ BP2 ✓ **BP8K0 !** BP8K1 ⚠ BP16K0 ✓ BP32K0 !

Bufferpool BP8K0 details: (Last Extract: 2023/07/11 10:00am)

size:	1234567	?
hitratio:	88	?
application:	77	?
system:	99	?
page residency:	333	?
number page updates/page:	987	?
number page writes/IO:	654	?
total page IO rate:	3210	?
synchronous IO rate:	7879	?

Details color coded

Bufferpool Advisor:

- BP too small, add 11% more storage
- Hitratio should be > 90%
- Application hitratio should be > 90%
- Page residency should be > 600
- I/O rate should be < 1000

→ Use SIMULATE to apply and simulate the changes then verify, using the BP history.

RESIZE DETAILS CHANGE OBJECT LIST SIMULATE VPSEQT

12:53 PM 7/11/23

The modern way to visualize BP/GBP problems

The screenshot displays the SQL WorkloadExpert for Db2 z/OS interface. At the top, it shows the selected profile: DD10SEC-IQA061QB - IQA061QB - Z100DD10 - 192.168.9.98 - 15151. The main navigation bar includes 'BP overview', 'Bufferpool details' (highlighted), 'New BP', and 'BP history'. Below this, a row of bufferpools is shown: BP0 (green check), BP1 (green check), BP2 (green check), BP8K0 (blue highlight with exclamation mark), BP8K1 (yellow triangle), BP16K0 (green check), and BP32K0 (red exclamation mark).

The 'Bufferpool BP8K0 details:' section (Last Extract: 2023/07/11 10:00am) lists the following metrics:

size:	1234567	?
hitratio:	88	?
application:	77	?
system:	99	?
page residency:	333	?
number page updates/page:	987	?
number page writes/IO:	654	?
total page IO rate:	3210	?
synchronous IO rate:	7879	?

A red callout box labeled 'Recommendations per problem' points to the hitratio (88) and application (77) values.

The 'Bufferpool Advisor:' section provides the following recommendations:

- BP too small, add 11% more storage
- Hitratio should be > 90%
- Application hitratio should be > 90%
- Page residency should be > 600
- I/O rate should be < 1000

At the bottom of the details panel, there are buttons for RESIZE, DETAILS, CHANGE, OBJECT LIST, SIMULATE, and VPSEQT. A note at the bottom of the advisor section states: '→ Use SIMULATE to apply and simulate the changes then verify, using the BP history.'

The modern way to visualize BP/GBP problems

With one click you generate all of the ALTERs you require and, naturally, we do not execute them!

You can “batch up” the changes and issue them at a quiet time.

Please remember:

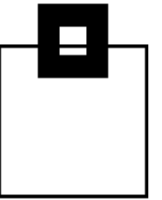
Measure your system first.

Do the ALTERs – Perhaps not **all** of them at once!

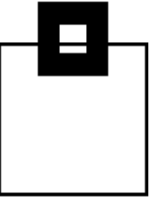
Measure your system.

You *should* see a system-wide improvement and/or application-level improvement.

This is all done **without** even knowing the applications that are running!



Agenda



- What use are BUFFERPOOLS?
- Is it worth tuning?
- How do you tune them?
- What about GLOBAL BUFFERPOOLS?
- The modern way to visualize BP/GBP problems
- **Q&A**



Questions & Answers

