MYTH-BUSTING

DB2 Version 8 CPU Regression

BY WILLIE FAVERO

The cry heard around the DB2 world of late has been, "How much CPU is DB2 Version 8 going to use?" Whether you call it CPU consumption, regression, or simply usage, this seems to be a big question on everyone's mind. For some, it's merely curiosity; they're moving forward regardless. However, they'd like to be prepared should something increase so they can adequately plan for it. Others, however, have questions that must be answered before they'll start their V8 upgrade. Still others aren't even considering an upgrade to V8 until they're comfortable they won't be affected by any CPU increase.

This article addresses concerns that DB2's CPU utilization will be horrible. Please note that the opinions expressed here are mine only and not necessarily those of IBM, its affiliates, or employees.

When you completed previous version upgrades, you always faced the possibility of some kind of CPU increase. With each version, DB2 is adding functionality, improving features, and increasing the amount of code that comprises DB2, all of which have the potential of increasing the amount of CPU it uses. This is the price of progress and V8 is no exception.

Consider what you're getting in V8:

- A ton of new SQL
- Lots of new access paths
- A bunch of stuff that makes DB2 easier to work with.

The catalog has been converted to Unicode and now uses long names. V8 has considerably more code than V7, and it's now taking advantage of 64-bit processing, which, by itself, can make things more exciting. Sometimes, 64-bit

instructions may take a little more time to run than their 31-bit counterparts; words are longer, registers are longer, control blocks are larger (some as much as twice as large), and some old 24- and 31-bit stuff needs to be expanded to 64-bit before it can be used.

So, if you're thinking CPU could increase, you're probably right. The problem lies in the amount of increase you think you might see. Those estimates are all over the board and based on a variety of information, some of which is misunderstood and some just plain ignored.

It's fascinating that CPU is such a focal point in V8 upgrade discussions because CPU was rarely a significant discussion point in past upgrades. We (IBM, DB2 consultants, users) created most of this controversy for ourselves. Sometimes, we can be just a little too helpful. In the past, we'd discuss CPU on

a feature-by-feature basis. However, in V8, we knew there could be some increases because of the reasons previously listed, so we tried to forewarn people with a little too much information.

So, this article carefully looks at what you might expect to see in V8. You might see some CPU regression. That's a given. So what actions will you take if you do see those CPU numbers increase more than expected? There are two answers here: one for Compatibility Mode (CM) and one for New Function Mode (NFM). Let's look at CM first.

Compatibility Mode

The first action you can take to reduce the amount of CPU consumed in V8 CM is to start rebinding your plans and packages. Some access paths you may get in V8 could use less CPU. If you don't rebind anything, you'll pay the price for V8 without reaping any of the benefits. If you have a tool such as Path Checker that tells you if you'll improve your access path by doing a rebind, you can reduce the number of total rebinds you may have to do and limit those to just the rebinds that will benefit you. Rebind is your first defense to increasing CPU cost, so take advantage of it.

If new and better access paths aren't reason enough to do a rebind, here's another: Back in V3, DB2 delivered a function called fast column processing or SPROCs (SELECT procedures). A SPROC is a performance enhancement that internally assists DB2's movement of columns. If a plan or package used a SPROC in V7, the SPROC was using 31-bit code. Because DB2 V8 is running in 64-bit, the SPROC is disabled. The only way you can re-enable the procedure is by rebinding it. Until you do that rebind, and if the plan or package was using a SPROC in V7, your application's performance will be degraded. Following the rebind, you should see a performance improvement and a reduction in CPU. The rebind can occur in CM or in NFM. If you do rebind the plan or package in CM, there's no reason to bind it again in NFM because of the SPROC.

DB2 does have other procedures, IPROCs (INSERT procedures), and UPROCs (UPDATE procedures) that aren't affected by their move to 64-bit and require no action.

There's a fairly old APAR available that fixes a problem with the metrics that track SPROCs. Make sure you have the APAR applied to ensure that the information gathered on a SPROC's use isn't misleading. The APAR is PK31412: *"Invalid SPROC Counter QISTCOLS Not Updated."*

OK, so that eased some of the CPU increase we might have experienced. What else can we do in CM that's more or less non-disruptive? One of the easiest changes you can make is to long-term page fix DB2's buffer pools. You don't want to fix all of them right away; rather, slowly take advantage of this fix. A page has to be fixed in memory during the I/O process. In V7, DB2 did this for you because the 31-bit instructions were very inexpensive. However, in V8, the 64-bit flavors of these instructions are more expensive. The pages must be fixed in memory; you have no control over that. The solution is to long-term page fix an entire buffer pool. V8 added the PGFIX keyword to the ALTER BUFFERPOOL command to accomplish this. You should long-term page fix your critical buffer pools that are experiencing a lot of I/O first, and slowly move your way through the rest of the pools while monitoring the effect. Leave the catalog and sort work pools for last, if at all.

While you're still in CM, you also can "supersize" your buffer pools and increase your use of DB2's hardware compression. Both can reduce the amount of I/O performed by DB2, reducing the amount of CPU DB2 uses. Compression requires a DDL change and, for both, you need to keep an eye on your IFCID 225 records and RMF reports. Even though people talk like you now have unlimited space above the bar, you don't. You still need to ensure that the amount of virtual storage DB2 uses *never* exceeds the amount of real storage you have available. Even the slightest amount of paging could be catastrophic to DB2's performance.

To save CPU while in CM, consider whether you can stop using the DSNZPARM keywords MINSTOR and CONTSTOR. Both of these may have been invaluable to you while you were in V7 and particularly if you were having storage problems. However, they both come with a CPU cost to reduce storage use. When you have to choose between using a little extra CPU and keeping DB2 up and running by not running out of storage, the choice was easy. Now that you're in V8, you should re-examine your need to use these keywords. If you have verified you no longer have a storage issue and aren't even close to having a storage issue, you can consider turning them off and reducing the CPU they use. However, if there's any chance your storage problem may resurface, you probably want to leave them on. There's an article that discusses CONTSTOR on IBM's DB2 support Website. To access, search on the DSNZPARM keyword CONTSTOR.

If you're using your accounting data to compare your V7 CPU usage to CPU use in V8 and you observe an increase, it's possible your CPU cost didn't really go up as much as you think. There are some cases in V8 where the CPU that was charged to the DB2 address spaces in V7 is now being charged to the application's TCB. What will initially appear to be an increase is no more than movement of the CPU cost from one place to another. To address this, examine your entire DB2 to see if an increase in one place isn't being offset by a decrease someplace else. Of course, you can't compare anything if you're not saving the necessary records to reproduce your accounting and statistics reports from V7.

New Function Mode

If you're using data sharing, you'll want to quiesce your entire data-sharing group at some point *after* going to NFM. This is necessary to enable DB2's new locking protocol II. This changes the locking strategy used with the coupling facility and could buy back a few CPU cycles. There's a great deal of detail in the DB2 manuals about locking protocol II.

If you haven't rebound your plans and packages in CM, you're definitely going to want to do it once you get to NFM. This will kick in the SPROCs again and reduce the "puffing" DB2 has to perform when it executes a plan or package. Until rebind, everything in a plan or package is 31-bit. At run-time, DB2 has to "puff" everything up to its 64-bit equivalent so it will run. Once DB2 completes the process, it throws it away. Then, the next time that package is needed, it has to go through the entire "puffing" process again. Rebinding for the sole purpose of eliminating the "puffing" is only going to buy back a minuscule amount of CPU, almost immeasurable. However, it's CPU, and rebind is something you should do anyway.

This same puffing process happens to a DBD. In CM, DB2 keeps DBDs in V7 format for fallback purposes and puffs them each time they're used; there's nothing you can do to change that. With NFM, though, you can no longer fall back to V7; it's time to get the

DBDs into V8 64-bit format and stop doing all the "puffing." Again, the CPU cost is minor, but it's a cost. Any change made to a DBD will force DB2 to store it in a 64-bit format. So you can do something minor, such as change its secondary extent to -1 and flip the DSNZPARM MGEXTSZ to YES to take advantage of sliding secondary extents, which is another nice NFM-enabled feature.

Let's go back to the idea of reducing I/O. A new DSNZPARM keyword DSVCI lets DB2 use 4KB, 8KB, and 16KB CIs that match the DB2 page sizes. If you have an 8KB page, you get an 8KB control interval. When you were still running V7, DB2 had to chain all its I/Os when the page size was greater than 4KB. For example, if a page was defined as 8KB, DB2 did two 4KB chained reads to get the page into the buffers. If a page was defined as 32KB, DB2 did an eight 4KB-chained read.

With DSVCI set to YES, the VSAM CI size will be the same as the page size with the exception of a 32KB page. If you have a page size of 8KB, you'll have a CI size of 8KB and a physical block size of 8KB, a page size of 16KB will have a CI size of 16KB and a physical block size of 16KB. However, if you have a page size of 32KB with a 32KB CI, the CI will span two physical 16KB blocks. The 32KB exception is there only for space utilization. With a 48KB track size, only one 32KB CI could fit per track if a 32KB block was used. Using 16KB blocks allows VSAM to span tracks for a 32KB CI and it lets VSAM handle all the I/O.

Another method to reduce your CPU takes advantage of some of the new SQL being delivered in V8, but is probably the most difficult method to implement. This method isn't available until you're in NFM, although you can start planning what you want to use and how you want to use it long before moving to NFM.

The most significant coding change you should consider is the use of multi-row FETCH and INSERT. The CPU savings when using these new SQL statements are significant. This improvement, along with its CPU savings, is also available with multi-row cursor UPDATE and DELETE. Multi-row processing also is included in the new DSNTEP4 (replaces DSNTEP2 when you get to NFM) and DSNTIAUL.

Star joins used in memory work files and Materialized Query Tables (MQTs) also can be used to help reduce your CPU time. AFTER Trigger processing and CPU utilization have been improved by eliminating the need for work files in certain situations. There's even a slight CPU improvement when using Rexx. Even changes such as allowing multiple DISTINCTs in a single SQL statement, support for multi-column predicates in a sort merge join, and backward index scan can help reduce your CPU usage.

Something else available only in NFM is the ability to combine a SELECT INTO, ORDER BY and FETCH FIRST to reduce the CPU cost of finding the first occurrence of a row.

The Nice Thing About Maintenance

Maintenance is critical to your success running DB2 V8 and, in some cases, keeping those CPU numbers low. You should have a solid maintenance strategy, preferably using Recommended Service Upgrade (RSU) and Consolidated Test (CST), attempting to stay as current on maintenance as possible. If you've been following this strategy, the following PTF may already be applied to your DB2 subsystem. The APAR is *PK28561: "Accounting Package Detail"* and it was closed back in September 2006; its PTF, UK18090, became available in October 2006.

This APAR has potential benefits in letting you reduce the amount of CPU DB2 will use collecting package accounting information. Many customers run Accounting Class 1, 2, 3, 7, and 8 traces as a standard practice. Prior to PK28561, V8 added a bunch of new stuff to the Class 7 trace record, more specifically to IFCID 239. Detailed information about lock manager, buffer manager, and SQL statistics were accumulated at the package level. The idea, excellent in theory, was to make IFCID 239 more useful. Unfortunately, this idea ended up adding an unacceptable increase in CPU to the process. IBM then decided to still give you the ability to gather this invaluable information. However, this APAR gives you the choice of incurring the additional CPU by enabling IFCID 239 to a different trace class. After this APAR, a Class 10 trace needs to be active to record the new information to IFCID 239. Monitor Class 10 also accomplishes the same thing.

You still have to turn on Accounting Class 7 or Class 8 before anything will be recorded by Class 10. Turning on only Class 10 (no 7 or 8) will not cause IFCID 239 to be written. You should turn on Class 7 and 8 so you have package-level accounting information. If you need more detail for problem determination, then also turn on Class 10 (in addition to 7 and 8) while you're trying to solve the problem. Not having Class 10 active will reduce the amount of CPU DB2 will use collecting package accounting information. Also, if you think you're using too much CPU for the accounting traces, make sure this APAR is on. Be aware, however, that if you have any homegrown tasks that require an old version of IFCID 239 from the Class 7 or 8 records before this APAR was applied, they may no longer give you the results you expect after applying this APAR. Also, if you're an OMEGAMON fan, there are a couple of APARs affected by this APAR that you also may want to check out. Take a look at APARs PK31295 and PK33459.

Conclusion

So, it's possible, more likely probable, that you will experience some CPU regression with V8 CM, but there are ways to control it and even reduce your CPU usage. However, once you move to NFM, you should start to see a decrease in CPU used. You also should experience a CPU improvement again when you upgrade to DB2 9 CM, and even more significant CPU improvements in DB2 9 NFM. In both DB2 V8 and DB2 9 NFM, the more new functions used, the greater the CPU savings. You're going to be quite pleased by the improvements in DB2 9. **z**

About the Author
In the past 29 years, Willie Favero has been a customer, worked for IBM, worked for a vendor, and is now an IBM employee again. He has always worked with databases and has more than 20 years of DB2 experience. A well-known, frequent speaker at conferences and author of numerous articles on DB2, he's currently with North American Lab Services for DB2 for z/OS, part of IBM's Software Group.
Email: wfavero@attlglobal.net
Website: www.ibm.com/software/data/db2/support/db2zos/

For more on the features and functions of V8 and its performance, see these resources:

- The Redbooks *DB2 UDB for z/OS Version 8 Performance Topics* (SG24-6465) and *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know ... and More* (SG24-6079), available at ibm.redbooks.com
- My blog at blogs.ittoolbox.com/database/db2zos
- DB2's Support Website at www.ibm.com/software/data/db2/support/db2zos/. Here, search on "John Campbell" or "Akira Shibamiya" and you'll find several valuable presentations and white papers.