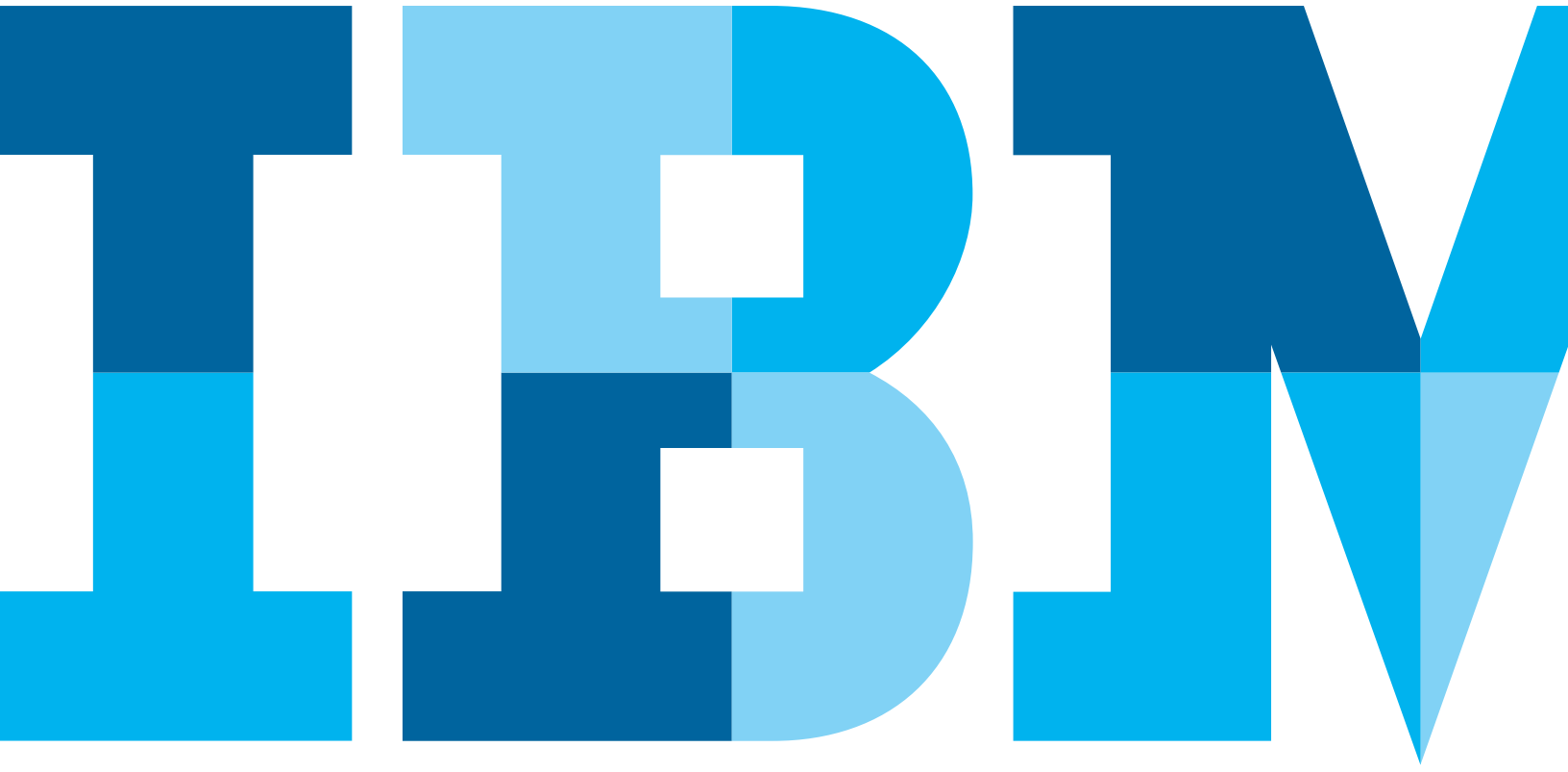
Small Steps to Big Gain DB2 10 helps save CPU resources



Contents

PART 1 - DB2 10 for z/OS Conversion Mode

This first part is limited to a discussion of DB2 10 Conversion Mode, in which it is still possible to fallback to the prior release (either DB2 VERSION 8 or DB2 9). Conversion mode is the state that DB2 is in when you first migrate from DB2 Version 8 or DB2 Version 9. During that time, you will not be allowed to use new functions that would cause incom-patibilities if you fell back to a prior release. In the next section of this whitepaper, we will talk about further CPU savings that one can achieve in DB2 10 New Function Mode.

New function is usually associated with new externals and application changes. Once those changes are made, the customer can often reduce DB2's CPU consumption. However, new functions occasionally require infrastructure changes. While circumstances vary depending on specific configurations, such new infrastructure typically causes an average "CPU regression" of as much as 5 percent when the new function isn't yet exploited. This is illustrated in Figure 1, where a negative CPU improvement indicates CPU regression. There was greater average CPU regression (typically 5 to 10 percent) when DB2 Version 8 introduced 64-bit support and long-name support in the DB2 catalog. Catalog infrastructure change is one of the causes for CPU regression in DB2 Version 8.

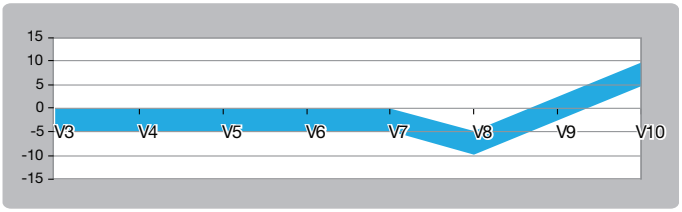


Figure 1. Average percent CPU improvement version to version

Saving CPU Resources Using IBM DB2 10 for z/OS -15 -10 -505 10 15 V3 V4 V5 V6 V7 VERSION 8 9 10 Figure 1. Average Percent CPU Improvement Version to Version.

The 64-bit structures in DB2 Version8 enabled much larger buffer pools and enabled other structures to grow, yielding better I/O performance and better CPU performance in some cases. DB2 9 made modest improvements in 64-bit exploitation, but DB2 10 is considered to be almost full exploitation. By that, we mean that in DB2 10 there are very few data structures below the bar. You will have to rebind packages in order to create the new 64-bit thread structures.

Large z/OS page frames

Among the infrastructure changes is the way that PGFIX(YES) buffer pools are managed on a z10™ or z196 processor. The z10 processor with z/OS 1.10 introduced 1MB page frames. Large page frames help z/OS improve the CPU performance as the amount of memory increases. IBM laboratory testing has measured CPU improvements of a few percent for specific transaction workloads. The only customer actions needed to exploit large frames are to define the buffer pools with PGFIX(YES) and to specify the LFAREA z/OS system parameter. LFAREA is the amount of real storage that z/OS will use for large frames.

RELEASE(DEALLOCATE)

RELEASE(DEALLOCATE) has been part of DB2 for a long time, but DB2 10 makes the function more useful. The dramatic DBM1 virtual storage constraint relief in DB2 10 that you can achieve with rebind makes it possible to use RELEASE(DEALLOCATE) more. This change has been shown to save up to 10 percent CPU time for high volume transactions with short running SQL without changing applications or DDL.

Distributed applications

For Distributed Data Facility (DDF) work, after rebinding packages with RELEASE(DEALLOCATE), the customer must issue the MODIFY DDF PKGREL(BINDOPT) command to allow DB2 to use RELEASE(DEALLOCATE) processing for packages bound with RELEASE(DEALLOCATE). DDF inactive thread processing (CMTSTAT=INACTIVE) takes advantage of new "high performance Database Access Threads (DBATs)" to increase distributed DBAT reuse. MODIFY DDF PKGREL(COMMIT) can be used to use commit behavior when, for example, you need to allow utilities to break in. With DB2 Logical Unit of Work (LUW) 9.7 Fixpack 3, the Call-Level Interface (CLI) and Java Database Connectivity (JDBC) packages are bound with RELEASE(DEALLOCATE) by default. RELEASE(DEALLOCATE) depends upon having very well debugged, well behaved applications that are careful with locking and committing frequently.

DB2 further improves overall Distributed Relational Database Architecture (DRDA) application performance for result sets from SELECT statements that contain the FETCH FIRST 1 ROW ONLY clause by combining the OPEN, FETCH and CLOSE requests into a single network request. DB2 10 also offers improved DDF performance through restructuring distributed processing on the server, particularly the inter-action between the DDF address space and the DBM1 address space.

All together, simply migrating to DB2 10 Conversion Mode, rebinding packages, exploiting large page frames and exploiting RELEASE(DEALLOCATE) can typically save from 5 to 10 percent of the CPU for transaction workloads, and up to 20 percent for native SQL procedure applications. Much greater CPU savings is possible for queries that contain large numbers of index stage 1 predicates, as well as IN-list predicates. Also, more stage 2 predicates can be pushed down to stage 1.

Capacity improvements

Since most of the thread storage is moved above the bar, DB2 10 can support more threads than DB2 9, thereby making it possible to reduce the number of DB2 data sharing members or, at least, hold the number of members constant while increasing transaction throughput.

If you were previously unable to increase the MAXKEEPD zparm value due to a DBM1 virtual storage constraint in DB2 9, you may be able to increase MAXKEEPD since the local statement cache is moved above the bar. This may provide CPU savings by avoiding prepares for more dynamic SQL statements.

I/O Improvements

Other CPU improvements apply in more specific scenarios. Some of these are related to I/O improvements since I/Os are one of the significant contributors to CPU time. These improvements include an improved dynamic prefetch sequential detection algorithm. List prefetch support for indexes helps minimize index I/Os when scanning a disorganized index. The number of log I/Os also is reduced, and long term page fixing of the log buffers saves CPU time, as well.

zIIP and zAAP exploitation

Buffer Pool prefetch and deferred write SRBs ordinarily are not big CPU consumers, but DB2 10 makes this specific processing eligible for System z® Integrated Information Processors (zIIPs). This CPU savings is more significant when using index compression.

As in DB2 9, DB2 can direct up to 80 percent of CPU intensive parallel query processing to run on an available zIIP. DB2 10 makes more queries eligible for query parallelism, which can result in more zIIP exploitation.

In DB2 10, portions of the RUNSTATS utility are eligible to run on a zIIP. XML schema validation and non-validation parsing of XML documents is eligible for zIIP or System z Application Assist.

Processor (zAAP). If XML parsing is done under DDF enclave threads, it is eligible for zIIP processing. If the XML parsing is done under a batch utility, it is eligible for zAAP processing.

Query performance

Range-list index scan is a new type of access path used by DB2 10 to significantly improve the performance of certain scrolling-type applications in which the returned result set is only part of the complete result set. The alternative in DB2 9 was to use multi-index access (index ORing) which is not as efficient as single-index access. Prior to DB2 10, list prefetch could not be used for matching IN-list access. In DB2 10, list prefetch can be used for IN-list table access ACCESSTYPE='IN'.

The process of putting rows from a view or nested table expression into a work file for additional processing by a query is called "physical materialization." Physical materialization itself is an overhead; in addition, it limits the number of join sequences that can be considered and can limit the ability to apply predicates early in the processing sequence. The join predicates on materialization work files also are not indexable. In general, avoiding materialization is desirable. In DB2 10, there are additional areas where materialization can be avoided, particularly for view and table expressions involved in outer joins.

The processing of stage 1 and non-index matching predicates has been enhanced. DB2 now processes non-Boolean predicates more efficiently when accessing an index and stage 1 data access predicates. You do not need to rebind your static applications to take advantage of some of these optimization improvements. However, a rebind is required to take full advantage.

More complex queries with many predicates show higher improvement. Queries that scan large amounts of data also show a higher saving in CPU.

DB2 10 also contains some SQL sorting enhancements. DB2 10 introduces hash support for large sorts, which potentially reduces the number of merge passes needed to complete them. Hashing can be used to identify duplicates on input to sort if functions such as DISTINCT or GROUP BY are specified. There also are some additional cases where, when FETCH FIRST N ROWS is used, DB2 10 can avoid the sort process altogether.

Insert Performance

The processing of stage 1 and non-index matching predicates has been enhanced. DB2 now processes non-Boolean predicates more efficiently when accessing an index and stage 1 data access predicates. You do not need to rebind your static applications to take advantage of some of these optimization improvements. However, a rebind is required to take full advantage.

When a series of Inserts are sequential with respect to the cluster key, but where the key is less than the highest key in the index, a new page selection algorithm helps minimize getpages, which can help reduce CPU cost.

DB2 10 contains some referential integrity checking improvements on Inserts that may help reduce CPU utilization and I/O processing.

Utilities

Generally speaking, DB2 utility CPU performance in DB2 10 is equivalent to that of DB2 9, but DB2 9 already introduced performance improvements of 5 to 50 percent CPU savings compared to DB2 VERSION 8 for various utilities processing.

Large Objects (LOBS)

DB2 10 contains numerous Large Objects (LOB) enhancements, and one of them applies to Conversion Mode: namely LOB materialization avoidance. For large LOBs, we have observed up to a 16-percent reduction in CPU consumption for DDF Inserts.

Conclusion

All of CPU savings that we have examined here apply to Conversion Mode. Some of the performance improvements are available upon installation of DB2 10, without additional changes, while others require minimal changes such as a change to installation parameters. Some require a z10 processor and perhaps changing the buffer pool parameters. Most require appropriate rebinds to fully realize the benefits. In section 2 of this report, we will start to look at the CPU benefits of migrating to New Function Mode.

PART 2 - DB2 10 for z/OS New Function Mode (NFM)

This is the second part of the report and focuses on reducing CPU resources or MIPS using IBM® DB2® 10 for z/OS®. The previous section summarized Conversion Mode.

Above we outlined how you can save CPU resources using Conversion Mode when migrating to DB2 10 for z/OS. This article will examine the CPU savings you may achieve when you migrate to New Function Mode (NFM) and begin to exploit the new functions.

There are two new functions that require universal table spaces and will help with CPU performance. These are inline large objects (LOBs) and hash access.

Inline Large Objects (LOBS) and HASH ACCESS — CPU Performance Enhancements

Small LOBs can be inlined with the other columns in the row. If a LOB is small enough to be completely inlined, the aux index and LOB table space aren't used, eliminating a lot of getpages and I/Os, as well as simplifying space management. The smaller the LOBs are, the greater the CPU savings, especially when sequential processes are involved. Tests with 200-byte inline LOBs yielded the following CPU savings:

- 93 percent for UNLOAD
- 86 percent for LOAD and a table scan
- 70 percent for sequential inserts
- 40 percent for random selects and deletes
- 21 percent for random updates

These CPU savings are in addition to significant elapsed time and DASD space savings.

Hash access is a new table-space organization that can save CPU costs when a clustering index isn't needed. Hash access is applicable when the data doesn't need to be clustered according to a clustering key, and where the queries involve equal predicates or an IN list predicate. IBM designers applied a hash access to the IBM Relational Warehouse Workload (IRWW), an IBM laboratory transaction workload that uses IBM IMS™ as a front end. IBM researchers determined that 55 percent of the tables merited conversion to hash access, and this resulted in a 10-percent CPU savings.

Non-key index columns are a new method of enabling index-only access, while at the same time minimizing the number of indexes and enforcing key uniqueness. By reducing the number of indexes, IBM researchers have observed a 10-to 30-percent CPU reduction for Inserts.

Dynamic Statement Literal Replacement is a new method of helping to improve the hit ratio in the global dynamic-statement cache, which reduces the CPU time when query statements differ only with respect to the particular literal value.

For LOBs, Copy and Recover can use IBM FlashCopy® to create and restore Virtual Storage Access Method (VSAM) image copies, thereby saving all of the CPU time associated with copying a DB2 object the old-fashioned way. Measurements have shown that FlashCopy saves CPU time for objects bigger than 7 MB: the bigger the object, the greater the savings.

Previous to DB2 10, a stored procedure could only return result sets to the intermediate caller. If the stored procedure is in a chain of nested calls, the result sets must be materialized at each intermediate nesting level, typically through a Dynamic Global Temp Table (DGTT) or Create Global Temp Table (CGTT). With the new Return to Client Result Set support, a result set can be returned from a procedure at any nesting level directly to the calling client application. No materialization through a DGTT or CGTT is required. By recoding the application to use the new support, IBM researchers have observed 96-percent CPU savings for nested stored procedures.

Bi-temporal and DB2 10

Bi-temporal is a feature of DB2 10 for z/OS that allows time-period attributes to be defined for a table: both BUSINESS_TIME (application-managed time period) and SYSTEM_TIME (DB2-managed time period). This not only gives users the flexibility in querying data based upon time periods, but also greatly reduces the development complexity for doing this type of functionality in user applications. Previously, applications have included temporal logic outside of DB2 by using triggers or stored procedures. Exploiting the bi-temporal support in DB2 10, these triggers and stored procedures may be removed, allowing some CPU savings.

Binary XML (formally called Extensible Dynamic Binary XML–DB2 Client/Server Binary XML Format) is an inter-change format that DB2 understands. It can help reduce DB2 CPU time spent parsing XML. Java applications that parse XML can use binary XML objects and send data to the DB2 application in a pre-parsed format so the application doesn't have to parse the XML again. IBM researchers have seen binary XML save up to 51 percent of the CPU time for an XML insert SQL statement.

XMLMODIFY (also called sub-document update) is a new function that allows updates to a portion of an XML document instead of the whole thing. This can save both CPU time and elapsed time. IBM researchers have seen 34-to 44-percent CPU savings for small updates (10KB), 72-to 84-percent savings for medium-size updates (100KB), and more than 99-percent savings for large updates (25MB), compared to full document updates.

Summary–DB2 10 and Savings CPU Resources

DB2 10 delivers significant benefits that save CPU resources. Many customers will be able to enjoy them with little or no effort. The benefits associated with the log latch require no administrative actions. However, other new functions do require some effort. The effort begins with some workload analysis to ensure the new function is applicable and will actually improve performance. For inline LOBs and hash access, existing data needs to be reorganized. For other enhancements, the application itself needs some modification or development. Collectively, these features help to deliver real and quantifiable business benefit.

About the author

Jeff Berger has worked for IBM for 33 years and is an expert on hardware and software synergy between DB2 and IBM's storage products.

IBM Corporation
Software Group
Route 100
Somers, NY 10589

Produced in the United States of America
August 2012

This document is current as of the initial date of publication and may be changed by IBM at any time.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary. It is the user's responsibility to evaluate and verify the operation of any other products or programs with IBM products and programs.

THE INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT. IBM products are warranted according to the terms and conditions of the agreements under which they are provided.

Actual available storage capacity may be reported for both uncompressed and compressed data and will vary and may be less than stated.

Please Recycle

SWW14013-USEN-00