Planning for IBM DB2 10 for z/OS Upgrade

Executive Summary

In the spring of 2010, DB2 10 for z/OS was released to 24 worldwide customers for beta testing. The evaluation focused on regression testing, "outof- the-box" performance, and additional performance and scalability, as well as other new functions.

Customer experience and feedback about the program have been mainly positive, and most customers who were involved in the program plan to start migration to DB2 10 for z/OS in 2011. An incremental improvement was observed in the effectiveness of the program, in terms of the quality of the issues and problems found, relative to the respective programs for DB2 Version 8 and Version 9. Some customers did very well with regression and new function testing; others provided only limited qualification about what they did and what they achieved.

After the early stages of planning and execution, it is often difficult for customers to sustain the effort required during a six-month period, due to competing business and technical priorities. People, hardware, and time are usually constrained to varying degrees. As of the end of the beta program, no customers were in "true, business production."

The release of DB2 10 for z/OS provides many opportunities for price/ performance and scalability improvements. But there is a tradeoff in terms of some increased real storage consumption.Customers need to carefully plan, provision, and monitor their real storage consumption.

The new, 64-bit SQL runtime can provide generous, 31-bit virtual storage constraint relief in the DB2 DBM1 address space. This support provides enhanced vertical performance scalability of an individual DB2 subsystem or DB2 member. It also opens opportunities for further price/performance improvement, through greater use of persistent threads running with the BIND option RELEASE(DEALLOCATE), DB2 member consolidation, and LPAR consolidation.

Introduction

This paper focuses on the planning stage of migrating to IBM DB2 10 for z/OS. The key points of emphasis are:

- Make sure everyone is educated as to what is needed to ensure project success.
- Production of a detailed project plan, communicated to all involved, is crucial for success.
- Some preparation can occur very early, in terms of understanding, obtaining, and installing the prerequisites.

The release of DB2 10 for z/OS was announced on February 9, 2010, and began shipping on March 12, 2010. It was the largest beta test program in the history of DB2 for z/OS.

The information in this paper is drawn from the lessons learned in cooperation with 24 of IBM's largest customers, representing a variety of industries and countries around the world. An extended beta test program started in Q3 2010 and lasted for six months. The program also included 73 parties in vendor programs.

These customers were looking mainly for 31-bit virtual storage constraint relief in the DBM1 address space and all opportunities for price/performance improvement. Other areas of interest included:r.

- Regression testing (*Be sure to approach regression testing in the order in which you plan to move to production.*)
- "Out-of-the-box" performance
- Additional performance improvements
- Scalability enhancements
- New functions

Stages of migration

The primary stages of migration to a new version are:

1. Planning
   - Early stages:
     - Making the decision to migrate
     - Determining what can be gained
     - Planning for prerequisites
     - Avoiding incompatibilities
     - Planning performance and storage
     - Assessing available resources
2. Migration
3. Implementation of the new improvements

Needed application changes can be made over a longer period to make the migration process easier and less costly. Plans for monitoring virtual and real storage resource consumption, as well as performance, are necessary. An early health check, communication of the required changes, and staging of the work will make the project go much more smoothly.

Highlights of the Beta Test

DB2 10 for z/OS delivers great value by ***reducing CPU resource consumption in most customer cases***. IBM internal testing and early beta customer results revealed that, depending on the specific workload, many customers could achieve ***"out-of-the-box" DB2 CPU savings of up to 10 percent*** for traditional OLTP workloads and up to ***20 percent for specific new workloads (e.g., native SQL procedures)***, compared with running the same workloads on DB2 9 for z/OS.

The objective of providing and proving generous, 31-bit virtual storage constraint relief in the DBM1 address space was achieved by the end of the program. This achievement is significant in terms of the enhanced vertical scalability of an individual DB2 subsystem or DB2 member of a data sharing group. We are confident that customers can scale up, in practical terms, the number of active threads by 5 to 10 times to meet their demands.

Further opportunities for price/performance improvement are made possible through the use of persistent threads with the BIND option RELEASE(DEALLOCATE). Examples of using persistent threads include protected ENTRY threads with Customer Information Control system (CICS®), Wait For Input (WFI) regions with Information Management System/Transaction Manager (IMS/TM), and high-performance database access threads (DBATs) for incoming Distributed Data Facility (DDF) workloads.

Another goal was to improve INSERT performance, particularly in the area of universal table spaces (UTS). We wanted to ensure that insert performance for UTS was equal to, or better than, the classic table space types, such as segmented and partitioned. This goal was achieved in most cases.

Hash access was good, provided we hit the smaller-than-expected "sweet spot." Results for complex queries were also good.

Provided users chose the correct value, the performance of inline large objects (LOBs) was also impressive. Support for inline LOB column values has the potential to save even more on performance by avoiding indexed access to the auxiliary table space. However, it is important to note that the value you choose for the inline LOB value must ensure that most of the LOB column values are 100 percent inline in the base table space.

In the area of latch contention reduction, we focused on the hot latches in DB2 10 for z/OS in such a way that, once we solved the 31-bit virtual storage constraint in the DBM1 address space, enabling you to scale five to ten times, we wanted to be sure there were no secondary issues related to latch contention that would inhibit the vertical scalability of a single DB2 subsystem or DB2 member.

As the beta program progressed, the reliability of, and customer confidence in, DB2 10 for z/OS greatly improved.

Generally speaking, online transaction processing (OLTP) performance improvements were as predicted. We were aiming for a target of 5 percent to 10 percent reduction in CPU resource consumption for most traditional OLTP workloads. During testing, several customers ran benchmarks showing that such reductions could be achieved. However, in cases where the transactions consisted of a few very simple SQL statements, the 5 percent to 10 percent target was not achieved.

This is where the increase in package allocation cost outweighed the improvement in SQL runtime optimization. However, we did identify some steps that can be taken to improve this. We have delivered an Authorized Program Analysis Report (APAR) to reduce package allocation cost. It is also possible to mitigate this situation by making more use of persistent threads with the BIND option RELEASE(DEALLOCATE).

Another issue was single-thread BIND/REBIND performance. Even in Conversion Mode (CM), the performance, in terms of CPU resource consumption and elapsed time, was degraded. One reason for this result was that in DB2 10 for z/OS the default for access plan stability is EXTENDED. Also, DB2 10 for z/ OS uses indexed access, even in CM, to access the respective DB2 Catalog and Directory tables.

Another area where we had mixed results was SQL Data Definition Language (DDL) concurrency. We had hoped that by restructuring the DB2 Catalog and Directory to introduce row-level locking, remove hash link access, and more, we could improve concurrency when running parallel SQL DDL and parallel BIND/REBIND operations. The concurrency improvement was eventually achieved for parallel BIND/ REBIND activity. Although it also helped in some cases with SQL DDL, most customers will still have to run SQL DDL activity single-threaded.

The final issue was access path lockdown. Two new options in DB2 10 for z/OS, APREUSE and APCOMPARE, enable you to generate a new SQL runtime while in most cases keeping the old access paths. Unfortunately, there were some issues with the underlying OPTHINTS infrastructure inherited by DB2 10 for z/OS, which is used by APREUSE and APCOMPARE. The introduction of APREUSE and APCOMPARE was delayed until these issues were addressed. These features are now available in the service stream via APARs, and their use is strongly recommended.

In general terms, the results of the beta program were mainly positive customer experiences, and we received good feedback about the program. A majority of customers in the beta program plan to start migrating to DB2 10 for z/OS in 2011. We observed incremental improvement in the program over what we experienced with the DB2 8 and DB2 9 for z/OS programs.

There was really no “single voice” or message across the customer set. We saw significant variation in terms of customer commitment and achievement. A small subset of customers did a very good job on regression and new function testing and provided good feedback. Others, due to limited resources, provided only limited qualification about what they were going to do and what they were able to achieve.

It is worth keeping in mind, for those who have never been involved in a Quality Partnership Program (QPP)/beta program, that it can be a challenge for customers to sustain the effort over a six-month period, due to competing business and technical priorities as well as constraints on people, hardware resources, and time.

By the end of the program, no customers were in true, business production. But we also need to appreciate that a QPP/beta program is not the same as an Early Support Program. We continue to develop and test the DB2 for z/OS product as the program progresses.

One of the benefits of DB2 10 for z/OS is that it provides many opportunities for price/performance (cost reduction) improvements. It is a major theme of this release. In discussions with customers, these opportunities for price/performance improvement are most welcome.

Also keep in mind that customers can be intimidated by some of the marketing "noise" about improved price/performance, often because of the raised expectation level of their respective CIOs. But in some cases, it is because when they run their own workloads, they do not see the anticipated improvements in CPU resource consumption and elapsed time performance. Many customers saw big improvements for certain workloads, while for other workloads, they saw little, if any, improvement.

Also note that if you have small test workloads that are untypical of the total mixed workload running in production, this can skew expectations on savings—either positively or negatively. Once DB2 10 for z/OS is in production, the results with the full, mixed workload may differ. We found that some measurements and quotes were overly positive and should be ignored.

A remaining question is: "How do you extrapolate from a small workload and project what the savings would be for the total, mixed workload in production?" Estimating with accuracy and high confidence is not practical, or possible, without proper benchmarking using a workload that truly represents production. Most customers reported incremental improvement over the DB2 8 and DB2 9 for z/OS programs.

Overall, most tests identified opportunities for price/performance (cost savings) improvements, which is the major theme of this release. Some customers reported big improvements in CPU and elapsed time reduction for certain workloads, while others did not. Keep in mind that smaller workloads may skew expectations on savings.

Summary of results

The DB2 10 for z/OS beta program confirmed improvements in the following areas:

- 31-bit virtual storage constraint relief in the DBM1 address space
- Insert performance
- Hash access good when hitting the smaller-than-expected sweet spot
- Complex queries
- Inline large objects (LOBs) and structured large objects (SLOBs)
- Latch contention reduction
- Quality of problems and issues found
- Reliability and confidence as program progressed

Performance and Scalability

One of the key lessons learned in the beta program was the need to plan on additional real storage. A 10 percent to 30 percent increase of real memory is a very rough estimate. For small systems with tiny buffer pools, the increase will be toward the high end of the range; for big systems with large buffer pools, it will be toward the low end of the range. It is important for customers to properly provision and monitor real storage consumption.

Many traditional OLTP workloads saw a 5 percent to 10 percent reduction in CPU utilization in CM mode after REBIND under DB2 10 for z/OS (some more, some less). On the initial migration to DB2 10 for z/OS, most customers will not perform a mass REBIND of all plans and packages. So, before REBINDing plans and packages, you may see little or no reduction in CPU resource consumption.

To maximize the price/performance benefits after migrating to CM, take these two steps:

1. REBIND your packages and plans to generate the new 64-bit SQL runtime. This way, you avoid the overhead of making the runtime for migrated packages from earlier releases look like the DB2 10 for z/OS runtime and re-enable fast column (SPROC) processing, which would otherwise be disabled.
2. Take advantage of 1 MB size real storage page frames to reduce translation lookaside buffer (TLB) misses. The 1 MB size real storage page frames are available on the z10™ and z196 processors. The prerequisite for using them is to specify the long-term page fix option for your local buffer pools. Long-term page fix buffer pools, which were introduced in DB2 8, provide an opportunity to reduce CPU resource consumption by avoiding the repetitive cost of page fix and page free operations for each page involved in an I/O operation.

The lesson is, be sure to use PGFIX=YES on your local buffer pools, provided there is sufficient real storage provisioned to fully back the requirement of the total DB2 working set below and above the 2 GB bar.

In a few cases, customers saw less than 5 percent saving in CPU resource consumption for traditional OLTP with very light transactions—"skinny" packages with a few simple SQL statements. This result is due partly to the increasing cost of package allocation, which overrides the benefit of the SQL runtime optimizations. APAR PM31614 may solve this issue by improving package allocation performance. Another way to address this is to use persistent threads with the BIND option RELEASE(DEALLOCATE) to amortize away the repetitive cost of package allocation/deallocation per transaction.

Regarding customers' measurements, keep in mind that—unlike the DB2 Lab environment, where a dedicated environment is used—customer measurements are typically performed in a shared environment, and the measurement results are not always consistent and repeatable. There can be wide variation on measurement "noise" in customer measurements, especially regarding elapsed time performance.

In most cases, customers were not running in a dedicated environment or at the scale/size of true business production. Many customers ran a subset (maybe a high-volume subset) of the total production workload. Sometimes, they used a synthetic test workload to study specific enhancements.

In cases where customers had very large numbers that they were not able to reproduce, the numbers on CPU and elapsed time reductions were not trusted.

Recommendation

Customers should not spend anticipated price/performance (cost reduction) savings until they actually see the improvements in their own true business production environment.

Early results

Table 1.1 summarizes some of the beta program results reported by customers. Some of the additional savings were due to features such as using 1 MB size real storage page frames for selective buffer pools, enabling high-performance DBATs, and respecting the package BIND option RELEASE(DEALLOCATE). Another reason was the improvement in COMMIT processing for applications that commit frequently. We now perform parallel writes to the active log dataset pair even when rewriting a log control interval (CI) that was partially filed and written out previously.

Workload	Customer results
CICS online transactions	Approximately 7% CPU reduction in DB2 10 CM after REBIND; additional reduction when 1 MB size real storage page frames were used for selective buffer pools
CICS online transactions	Approximately 10% CPU reduction from DB2 9
CICS online transactions	Approximately 5% CPU reduction from DB2 8
CICS online transactions	10+% CPU increase
Distributed concurrent insert	50% DB2 elapsed time reduction; 15% chargeable CPU reduction after enabling high-performance DBAT
Data sharing heavy concurrent insert	38% CPU reduction
Queries	Average CPU reduction 28% from V8 to DB2 10 NFM
Batch	Overall 20–25% CPU reduction after rebind packages

Table 1.1: Workload results reported by DB2 10 for z/OS beta program customers

Now, let us discuss the use of the 1 MB size real storage page frames on the z10 and z196 processors. The potential exists for reduced CPU resource consumption through fewer TLB misses; however, the local buffer pools must be defined as long-term, page-fixed (PGFIX=YES). This feature was introduced in DB2 8 to mitigate CPU regression and reduce CPU resource consumption for I/O-intensive buffer pools.

Many customers are still reluctant to use the PGFIX=YES option because they are running too close to the edge on the usage of the amount of real storage provisioned and are in danger of paging to auxiliary (DASD) storage. They understand the value of PGFIX=YES, but it applies only for an hour or two each day. Another factor is that this decision is a long-term one; in most cases, implementing this buffer pool attribute requires a recycle of the DB2 subsystem. A change to the attribute goes pending and is materialized when the buffer pool goes through reallocation. It is also worth noting that a 75 percent cost reduction on real storage is incurred on the z196 processor relative to the z10 processor.

Here are a few more things to remember about the use of 1 MB size real storage page frames on the z10 and z196 processors: The actual amount of memory that is allocated as 1 MB size real storage page frames is specified by the LFAREA=nn% parameter in the IEASYSnn parmlib member and is changeable only by IPL. You are partitioning out the total real storage provisioned between 4K size frames and 1 MB size frames. 1 MB size real storage page frames are nonpageable. If these page frames are overcommitted, DB2 10 for z/OS will start using 4 K size real storage page frames.

Recommendation

Assuming you have provisioned sufficient real storage in production to fully back the total requirement of the DB2 working set:

1. Define all the local buffer pools as long-term page fixed (PGFIX=YES),

2. Sum up the total buffer pool storage requirement across all the local buffer pools defined as PGFIX=YES, and

3. Reflect that value in the LFAREA specification. (You may want to add an additional 10 percent to 20 percent in size to allow for some growth and tuning.)

Note

Make sure you have applied critical preventative z/OS maintenance before using 1 MB size real storage page frames. One of the lessons learned in the beta program is that the 1 MB size real storage page frames are relatively new and DB2 10 for z/OS is the first major subsystem to exploit them. We observed a reduction of up to 6 percent in CPU resource consumption. There is a customer requirement for a new parameter to be able to use PGFIX=YES independently from the use of 1 MB size real storage page frames. This requirement will be addressed in a future release of DB2 for z/OS.

DBM1 virtual storage constraint relief (VSCR) with a near-full 64-bit SQL runtime is available for use as soon as you go to CM. To accrue maximum benefit, you must REBIND static SQL plans and packages. We are confident that we have addressed the previous vertical scalability issue on the limited number of active threads that could be supported, and we have achieved very good results.

This support offers a "real-world" proposition of scaling up the number of active threads from, say, 500 active threads to 2,500–3,000 active threads or more per DB2 subsystem. The limiting factors now on vertical scalability (number of threads times average thread storage footprint) are most likely to be the amount of real storage provisioned on the logical partition (LPAR), followed by extended system queue area/extended common service area (ESQA/ECSA) (31-bit) storage constraints and the active log write performance (log latch contention).

Figure 1.1 shows three sets of customer measurements.

The first measurement (shown in the left column of the figure) is the virtual storage footprint of DB2 9 for z/OS.

The middle column shows the virtual storage footprint of DB2 10 for z/OS in CM without the REBIND static SQL plans and packages. The issue here is that the footprint actually increased, compared with DB2 9 for z/OS. This issue was corrected ahead of GA of DB2 10 for z/OS.

The third column shows that once you do the REBIND of static SQL plans and packages, the 31-bit virtual thread storage footprint decreases dramatically. This result illustrates the value of the DBM1 31-bit virtual storage constraint relief in DB2 10 for z/OS.

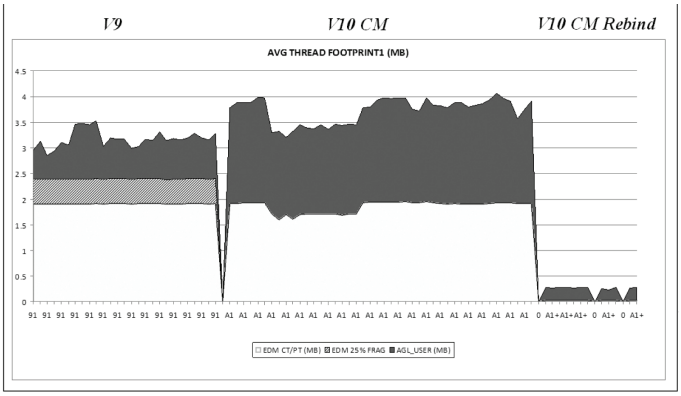

Figure 1.1. Initial DBM 1 31-bit thread storage customer measurements in DB2 9 for z/OS vs. DB2 10 for z/OS (corrected prior to General Availability [GA])

Figure 1.2 shows another group of customer measurements

Here, the first column is the DB2 9 for z/OS thread footprint. The second column is the DB2 10 for z/OS CM without the REBIND of static SQL plans and packages. In columns three and four, you can see that after the fix is applied (even without the REBIND), the thread storage footprint is greatly reduced.

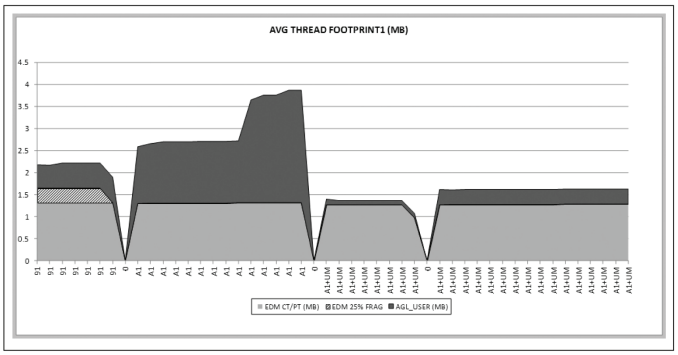

Figure 1.2. Initial DBM1 31-bit thread storage customer measurements in DB2 9 for z/OS vs. DB2 10 for z/OS [GA after fix applied])

With or without the REBIND of static SQL plans and packages, the 31-bit thread storage footprint in the DBM1 address space is reduced in DB2 10 for z/OS. However, to accrue maximum benefit in terms of 31-bit VSCR in the DBM1 address space, rebinding static SQL plans and packages is strongly recommended.

DBM1 virtual storage constraint relief with 64-bit SQL runtime

REBINDing static plans and packages maximizes the DBM1 31-bit VSCR and ensures we have a 64-bit SQL runtime. Not only does this step solve scalability issues, but it also can provide opportunities for further price/performance improvements—beyond the 5 percent to 10 percent.

For example, prior to DB2 10 for z/OS, many customers have been heavily constrained on available, 31-bit virtual storage in the DBM1 address space and, as a result, on the number of active threads that can be supported in a single DB2 subsystem or DB2 member. They have had to make compromises, trading additional CPU resource consumption to reduce the 31-bit virtual storage footprint and be able to support more active threads in a single DB2 subsystem or DB2 member.

This tradeoff involved reducing the number of persistent threads and restricting the use of the BIND option RELEASE(DEALLOCATE) for packages running on those threads. These tactics saved on DBM1 31-bit virtual storage resource consumption at the cost of incurring additional CPU resource consumption.

With DB2 10 for z/OS, provided you have additional storage provisioned over and above the 10 percent to 30 percent previously mentioned, you can use more persistent threads and make more use of the BIND option RELEASE (DEALLOCATE) with existing or new persistent threads. This capability has the potential to reduce CPU resource consumption and improve price/performance (cost reduction) beyond the previously mentioned 5 percent to 10 percent. However, it does require additional real storage to be provisioned to support the increased number of persistent threads running with RELEASE(DEALLOCATE). This is in addition to the 10 percent to 30 percent increase in real storage requirement discussed previously.

The next, and new, opportunity for price/performance improvement is with regard to Distributed Relational Database Architecture™ (DRDA) and DDF server workloads. In DB2 10 for z/OS, starting with CM there is the potential to reduce CPU resource consumption for DRDA transactions by using highperformance database access threads. DB2 10 for z/OS provides the same opportunity for thread reuse with persistent threads that we have, for example, in CICS with protected ENTRY threads and/or by queuing on an unprotected ENTRY thread.

To take advantage of this improvement, the first prerequisite is that at least one of the packages associated with the transaction must be bound with RELEASE(DEALLOCATE). The second prerequisite is to issue the MODIFY DDF PKGREL (BNDOPT) command so that the BIND option RELEASE(COMMIT|DEALLOCATE) is respected.

After taking these steps, you will be able to achieve thread reuse for the same connection. At the same time, DDF will start respecting the BIND option of RELEASE (DEALLOCATE). Before DB2 10 for z/OS, you could BIND distributed packages with the RELEASE (DEALLOCATE) attribute, but the availability of this option was a moot point because RELEASE(COMMIT) was always forced at execution time (i.e., the BIND option of RELEASE(DEALLOCATE) was not respected).

Now, in DB2 10 for z/OS, we have the same possibility as with CICS and IMS/TM workloads—to have persistent threads, in this case with highperformance DBATs, and to have the BIND option of RELEASE(DEALLOCATE) respected.

The recommendation is that once you plan to start using high-performance DBATs, consider provisioning additional real storage—beyond the previously discussed 10 percent to 30 percent increase. Do not adopt a "one size fits all" strategy when using more persistent threads with the BIND option RELEASE(DEALLOCATE) with IMS/TM, CICS, or DDF workloads. Most installations cannot support making all threads persistent threads, with all the associated packages bound with RELEASE(DEALLOCATE), because of the potential for dramatic increase in the total real storage requirement.

You simply cannot afford to use the BIND option RELEASE(DEALLOCATE) for all plans and packages. Target persistent threads for thread reuse at high-volume simple transactions, and couple them with use of RELEASE(DEALLOCATE) for highuse packages with many SQL statements that are frequently executed.

For example, take your Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) packages as used by distributed client applications and BIND them twice—into two different package collections: BIND them with RELEASE(DEALLOCATE) in one collection, and BIND them with RELEASE(COMMIT) in the other collection.

In this way, you can target the high-volume transactions that would benefit the most from the use of persistent threads with BIND option RELEASE(DEALLOCATE) and connect those transactions to a data source that points to the collection where the packages are bound with RELEASE(DEALLOCATE). Packages must be bound with RELEASE(DEALLOCATE) to be eligible to use high-performance DBATs and be reused for the same connection. The remaining transactions would connect to a data source that points to the collection where the packages are bound with RELEASE(COMMIT).

The story is similar with CICS and IMS/TM. For CICS, you would choose only protected ENTRY threads for high-volume transactions and couple that with the use of BIND option RELEASE(DEALLOCATE) for frequently executed packages. Allow the rest of the transactions to run as POOL threads.

For DRDA workloads, do not overuse BIND option RELEASE(DEALLOCATE) on packages, because it will drive up the MAXDBAT requirement.

Another point to remember is that when you use persistent threads with RELEASE(DEALLOCATE), there is a tradeoff. Doing so will impact BIND/REBIND and SQL DDL concurrency. When you have a high-volume transaction that justifies use of persistent threads with RELEASE(DEALLOCATE), then BIND/REBIND and DDL activity cannot break in.

Many customers fail to see the benefit of thread reuse and avoiding the repetitive cost of thread create and thread terminate per transaction. Here is the explanation as it relates to CICS: If you are incurring the overhead of thread create and thread terminate, you cannot see the overhead in the DB2 accounting record. On the other hand, if you avoid the overhead of thread create and thread terminate, you also cannot see the overhead saved in the DB2 accounting record.

CICS uses the L8 TCB to process DB2 work, regardless of whether the application is thread safe or not. The CPU time associated with thread create and terminate (or the avoidance thereof) shows up in the CICS System Management Facilities (SMF) Record Type 110 record. Note that before the introduction of the Open Transaction Environment (OTE) in CICS, CICS did not even capture the cost of thread create and terminate in the SMF Record Type 110 record. The CPU cost of thread create and terminate was not captured. Provided successful thread reuse is achieved, the benefit of using BIND option RELEASE(DEALLOCATE) will show up in a reduction in the Class 2 TCB Time in the DB2 Accounting Record (SMF Record Type 101).

For some customer installations, DB2 10 for z/OS also has the potential to reduce the number of DB2 members in a data sharing group. Some customers had to grow their DB2 processing capacity horizontally due to the 31-bit virtual storage constraint in the DBM1 address space by growing the width of the data sharing group by adding additional DB2 members. Some of these same customers decided to run multiple members from the same DB2 sharing group on the same LPAR.

Why? They wanted to limit the number of LPARs running on the faster z10 and z196 systems because of increasing LPAR overheads. But they needed to keep the existing DB2 members, or even add DB2 members, to have enough thread processing capacity.

Now, with the generous DBM1 31-bit virtual storage constraint relief in DB2 10 for z/OS, such customers have the ability to reduce the total number of DB2 members in a data sharing group. This change can reduce the number of DB2 members from the same data sharing group running on the same LPAR down to one, and can possibly reduce the total number of LPARs as well. The ability to reduce the total number of DB2 members and/or the number of LPARs will provide further price/performance (cost reduction) improvements.

Before you consolidate DB2 members and LPARs, there are some issues to consider. For example, what will happen to the logging rate when you push more workload through a single DB2 subsystem? And, can the size of the active log configuration, the dataset placement, and the I/O subsystem cope with the load? Will log latch contention be aggravated?

By running more workload through an individual DB2 subsystem, you will drive up the aggregate logging rate for that DB2 subsystem. Also, you need to consider the increase in SMF data volume per LPAR. In DB2 10 for z/OS, you can now enable DB2 compression of instrumentation record data written to SMF (e.g., DB2 accounting trace data) to reduce the SMF data volume. DB2 instrumentation data, such as statistics trace and accounting trace records, are typically written out to SMF and can benefit from this enhancement.

A new DB2 system parameter (ZPARM) called SMSCOMP, once enabled, turns on DB2 compression of the SMF output records. This compression applies to any instrumentation record, not just statistics and accounting, that is written out to SMF. We have observed a 70 percent to 80 percent reduction in the volume of SMF data when the DB2 compression is turned on. The CPU overhead incurred is only about 1 percent— representing a very good tradeoff.

This enhancement provides an opportunity for improved problem determination (PD) and problem source identification (PSI) by offering the possibility of turning off the use of accounting roll-up for DDF and Recovery Resource Services attachment facility (RRSAF) workloads (default). We introduced this support in DB2 8 to reduce SMF data volume, but one of the drawbacks of accounting roll-up was that it compromised performance PD/PSI.

By rolling up the transaction activity for multiple transactions into a single accounting record, we lose information about outlying, badly performing transactions. The information about the poor performance of outlying transactions gets "amortized" away by the accounting roll-up. Given the introduction of SMF data compression in DB2 10 for z/OS, SMF compression may be a better option to control SMF data volume than using the accounting roll-up.

Another consideration when migrating to any new DB2 for z/OS release is the impact of increased dump size due to growth in the total DB2 working set size (and the need to avoid partial dump capture). Make sure sufficient real storage is provisioned on the LPAR for the increased DUMPSRV and MAXSPACE requirement.

Finally, we want to re-emphasize the continued business and technical value of DB2 data sharing to differentiate the z/OS platform in terms of providing continuous availability across both planned and unplanned outages. You want to avoid large single points of failure. For example, consider a minimum configuration of four-way data sharing for true, continuous availability, assuming a twoprocessor (CEC) configuration.

By "four-way data sharing," we mean when you have two boxes (CECs) and two LPARs on each box (a total of four LPARs). A single DB2 member would run on each LPAR. That is the minimum recommendation for true, continuous availability and to maintain performance, if you want to maintain your service level agreement (SLA).

In this four-way configuration, if you were to lose a DB2 member on one LPAR, the surviving DB2 member on the alternate LPAR on the same box can take on 100 percent of the workload and use all the CPU processing capacity available on the box.

Planning for real storage

Let us discuss now, in more detail, the need to carefully plan, provision, and monitor real storage consumption. Most DB2 8 and DB2 9 for z/OS customers are properly configured and provisioned in terms of real memory. However, some are running so low on available real memory that part of the DB2 working set is often being paged out, intermittently, to auxiliary (DASD) storage.

Worse still, if a dump were to be taken on the system, the dump capture would take several minutes instead of a few seconds to complete, and it could spread "sympathy sickness" around a data sharing group. Information about real and auxiliary frames used is already recorded in the IFCID 225 record generated by DB2 for z/OS. However, although the provided information has been improved, with more details recorded in DB2 10 for z/OS, the information furnished in IFCID 225 has not allowed a customer installation to effectively monitor 64-bit shared and 64-bit common storage when running multiple DB2 subsystems on the same LPAR.

A new DB2 APAR PM24723 for DB2 10 for z/OS provides the needed capability. The new APAR uses the enhanced capability provided with MVS APAR OA35885, which provides a new callable service to Real Storage Manager (RSM) to report REAL and AUX usage for a given addressing range for shared objects. APAR PM24723 will have this new MVS APAR as a prerequisite.

The other advantage is that this same DB2 APAR provides a much-needed real storage management function within DB2 when available real storage is overcommitted and the system starts to be paged out.

Some customers have used a hidden system parameter (ZPARM) called SPRMRSMX (real storage "kill switch") when running multiple DB2 subsystems on the same LPAR. SPRMRSMX protects individual DB2 subsystems and other subsystems running on the LPAR such that if one of the DB2 subsystems were to "run away" in terms of virtual memory use, that subsystem would be "sacrificed" so that the other DB2 subsystems could continue to run.

Customers using system parameter SPRMRSMX have calculated the "normal" working set of a DB2 subsystem, multiplied that value by 2 (as a contingency), and used the resulting value as the SPRMRSMX setting. Customers currently using this system parameter will need to carefully re-evaluate the value set when migrating to DB2 10 for z/OS.

In DB2 10 for z/OS, you will need to factor in the increased use of 64-bit shared and common storage to establish the new DB2 for z/OS storage footprint. IPL amounts for the LPAR will need to be adjusted based on the number of DB2 members running on that LPAR. The following values are on a "per DB2 subsystem" (i.e., you would double these numbers when running two DB2 subsystems on an LPAR, triple them for three, and so on):

Storage area	IPL amount
64-bit private	1 TB
64-bit shared	128 GB
64-bit common	6 GB

Note carefully that these values are not indicative of real memory to be used, or even of virtual memory to be allocated; they simply represent reserving an addressing range for DB2 for z/OS to use. These large memory object areas are allocated above the 2 GB bar, and they will be sparsely populated. Virtual memory is not allocated until the pieces of storage are actually referenced.

INSERT performance

INSERT is one of the most important SQL statements in DB2 for z/OS. It is also one of the most challenging for any database management system (DBMS) to handle. Previous DB2 for z/OS releases have focused on improving INSERT performance. DB2 10 for z/OS provides some improvements for all table space types. There was particular focus on improving INSERT performance for universal table spaces, both partition by range (PBR) and partition by growth (PBG).

Over the longer term, what we want to do in DB2 for z/OS is converge all the classic table space types to be UTS and deprecate the old, classic table space types. DB2 10 for z/OS includes two specific enhancements to improve UTS performance. First, UTS now supports MEMBER CLUSTER to help where there is excessive page latch and page p-lock contention on space map pages and on data pages when using row-level locking. Second, changes were made to the space search algorithm, making the algorithm used by UTS now more like that used by the classic partitioned table space.

The performance goal for INSERT in DB2 10 for z/OS was for UTS to be equal to, or better than, the classic partitioned table space. While we are not there yet, the performance is dramatically improved. The improvement is very workload dependent. There is still a tradeoff between space reuse versus throughput and reduced contention. We still have some work to do on UTS, in the area of both PBR/PBG with row-level locking and sequential insert activity.

Three specific improvements to INSERT in DB2 10 for z/OS should help all table space types. The first is reduced log record sequence number (LRSN) spin for inserts to the same index or data page. As processors become faster, such as z10 and z196, the possibility of duplicate LRSN values and spins having to occur increases. When a spin occurs, processing loops in the DB2 code wait for the LRSN value to change. The LRSN value is used in data sharing to serialize restart/recovery actions, and it is the high-order six bytes of the store clock (STCK) value.

The LRSN is incremented every 16 microseconds. As processors get faster, there is increased potential for LRSN duplicate values and the need to spin. We already made some improvements in DB2 8 and DB2 9 for z/OS regarding this issue.

In DB2 10 for z/OS, when we have multi-row inserts (MRI) or single inserts within an application loop, we avoid the LRSN spins for the same page that would have occurred previously. The results have been very impressive. This improvement applies when you use multi-row inserts to the same page or have INSERT within an application loop to the same page, in a data sharing environment.

The second improvement, which works very well, is an optimization for "pocket" sequential insert activity. This is where you have multiple "hot spots"in the key range and the INSERTs are "piling in" on these hot spots. During insert, DB2 Index Manager (IM) identifies to the DB2 Data Manager the candidate row ID (RID) value (page) to be used to place the new data row. DB2 Index Manager now returns the next lowest key RID value. The end result achieved is a much better chance to find the space and avoid a space search.

The third improvement relates to parallel index read I/O, which works very well and is best-suited when it is activated where there random index key inserts. This mechanism is used when three or more indices exist on the table and you are performing random index key INSERTs. Previously, you would have had a lot of random sync read I/O. We now do parallel index read I/O when there are three or more indices on the table. This reduces the elapsed time by taking the synchronous read I/O activity out of the elapsed time.

To compensate elsewhere for the increase in CPU resource consumption, DB2 will now make the CPU resource consumption associated with prefetch engines (sequential prefetch, list prefetch, and limit prefetch) and deferred write engines eligible for zIIP offload. These types of processing are now offloaded to zIIP processors to compensate for the increase in CPU when doing parallel index read I/O for random key INSERTs.

Accounting Trace Class 3 enhancement

In DB2 10 for z/OS, there are now separate counters for IRLM Lock/Latch Wait and DB2 Latch Wait events in the DB2 accounting trace. Previously, both types of wait events were included in a single counter. When analyzing application performance problems, you had to try to figure out which type of wait activity was elevated.

The next improvement relates to data sharing. One of the disadvantages of having very large, local buffer pools with many group buffer pool (GBP) dependent objects, was that DB2 for z/OS used to scan the local buffer pool for each GBP-dependent object during DB2 shutdown. These scans added a lot of delay in shutting down the DB2 subsystem. DB2 also used to scan the local buffer pool when an object went into or out of GBP dependency. This activity could add a lot of overhead, depending on how often these transitions were made.

In DB2 10 for z/OS, we expect faster DB2 shutdown times because we avoid the local buffer pool scan per GBP-dependent object during the shutdown. We now also avoid the local buffer pool scan when an individual object (pageset/ partition) transitions into or out of GBP dependency.

Inline LOB column values are now supported in DB2 10 for z/OS. The size of the inline portion can be specified as a system parameter (ZPARM) or on an individual object basis. There is no "one size fits all" value for the use of inline LOBs. So, using a general value as a system parameter is unlikely to be a good choice.

You will get more value by setting the inline LOB value on the SQL DDL for the specific object. The performance tuning goal is to avoid access to the auxiliary table space for the majority of LOB column values. This function is aimed primarily at applications that have many, small LOB columns values (i.e., up to a few hundred bytes).

The design goal for the inline LOB value is to store the complete column value inline, in the base table row, and to avoid access altogether to the auxiliary table space. The potential exists for significant CPU and elapsed time improvement if this can be achieved by setting the right value for the inline portion.

However, if you store the LOB column value inline, in the base table row, and then very rarely reference the LOB column value, you may impact performance elsewhere because you will get fewer rows per page. In any event, you may need to consider increasing the data page size.

In the worst case, if you have made a poor choice for the inline LOB column value, you will have the first part of most LOB column values in the base table and the remaining part of the LOB column values in the auxiliary table space. So, not only will you get no benefit, but you will actually increase CPU overhead and waste DASD space. But another advantage to inline LOBs is that the portion of the LOB that is stored in the base table row is now eligible for data compression.

Another performance enhancement to DB2 10 for z/OS relates to active log writes. Before DB2 10 for z/OS, DB2 active log writes were always done serially to log copy 1 and log copy 2 when rewriting a previously written log CI that was partially filled previously. DB2 would write to log copy 1, wait, and then, when it was successful, write to log copy 2. The reason for this was that, prior to RAID devices, we had single, large, expensive disks (SLEDs). We were always concerned that, when we rewrote a previously partially filled log CI, we might destroy the previous version and its contents.

With the increased reliability provided by RAID devices, there is no longer any reason to do these rewrites of log CIs serially. DB2 10 for z/OS now always performs active log writes in parallel. This enhancement can generate significant elapsed time improvements and improvements in applications that commit frequently or when other forced writes occur (e.g., related to index leaf page splits).

Hash access vs. index-only access

Hash access basically "competes" with clustered index access, and specifically with index-only access. In an effort to reduce CPU resource consumption, hash access tries to avoid going through an index B-tree structure with many levels to access the data row. The advantage that clustered index access has is that DB2 still tries to maintain clustered data row access. Index-only access avoids access to the data row completely. DB2 10 for z/OS also provides the opportunity to have a unique index with INCLUDE columns.

Today, you may have multiple indices on a table. One index is there to enforce the uniqueness of the primary key. You may have added another index to improve performance. The leading columns may be the same in both indices. You may now include additional columns in a unique index and still use that same index as before to enforce the unique constraint.

Now, the advantage of a unique index with INCLUDE columns is that it gives you the ability to satisfy the unique constraint check and provide the performance benefits you want for query. The result is that you can reduce the number of indexes required for performance reasons. For every index you can avoid, you will improve the performance of INSERT and DELETE and possibly improve UPDATE performance, as well.

A number of customers evaluated both methods to try to find the "sweet spot." There is definite value from hash access, provided you can determine that sweet spot. However, in practice, the sweet spot has proved to be relatively small. Here are guidelines for identifying the sweet spot:

- High NLEVELS in index (more than two)
- Access by applications needs to be purely direct row access by primary key
- Truly random access
- Read-intensive, not volatile
- No range queries (minimize BETWEENs, >, <, and so on)
- Many rows per page

One of the key points about hash access performance is that you want to "tune" the space allocation of the fixed-sized hash area so that you reduce the number of rows that go into the overflow index (i.e., control overflow). If the primary fixed hash area is too small, you will have many rows in the overflow index; on the other hand, if the primary area is too large, you will have too much random I/O.

To help with sizing the fixed hash area size, DB2 10 for z/OS provides a new option on the REORG utility called AUTOESTSPACE(YES). When you perform REORG with this option, it uses information from Real Time Statistics (RTS) to resize the primary fixed hash area and reduce the number of rows in the overflow index. However, even after such a REORG, there may still be some small number of data rows in the overflow index.

Finally, when you migrate to hash access, you will see some degradation in the elapsed time for both LOAD and REORG utility executions.

Availability

There are a number of enhancements in DB2 10 for z/OS to reduce planned outages for applications and to improve the success of the online REORG utility.

Online Schema Evolution

"Deferred Alter" is a new feature in DB2 10 for z/OS. With this mechanism, when you make a schema change, the change goes "pending" and it is stored in the DB2 Catalog. The next time you perform an online REORG, the online REORG will materialize the pending changes. You can set up many deferred alters. Each of the changes will go pending in the DB2 Catalog until the subsequent online REORG, when the changes will be materialized. Why is this important? This mechanism now gives you a migration path away from the classic table space types of simple, segmented, and partitioned—which contain a single table—over to universal table spaces.

Note

UTS is a prerequisite for some of the DB2 10 for z/OS functions, such as hash access, inline LOB, and currently committed. It is also a prerequisite for the cloned table function in DB2 9 for z/OS. If a table space is a simple table space or a segmented table space, you can have only one table per table space to be able to use this migration path to UTS, because UTS supports only one table per table space.

Note

This migration path to UTS is a "one-way ticket" only. Once you migrate to UTS, you cannot go back using the same Deferred Alter mechanism to simple, segmented, or partitioned table spaces. To return to using the classic table space types, you would have to unload the data, drop the table space, redefine the table space as it was before, and reload the data.

Note also that point-in-time recovery to a point before a successful materializing online REORG is not possible. If, for example, you have incorrect results from REORG, possibly because the wrong rows were discarded or an application change needs to be rolled back, you cannot recover to a point before the online REORG.

Now, once you have migrated to UTS PBG/PBR, you can change attributes such as DSSIZE and index page size. You can turn MEMBER CLUSTER on and off or migrate to and from hash access. These abilities are all provided by the Deferred Alter mechanism, followed by the online REORG. This function works very well and can help reduce the number of destructive database changes that previously caused database downtime.

To summarize, the benefits of Deferred Alter are:

- Streamlining the move to UTS
- Reducing the administrative time and cost associated with moving to UTS
- Helping minimize errors
- Reducing outages

Another new option is the FORCE option of online REORG. In the last part of the REORG, when you are in the final attempt to drain the object and are about to make the switch, if there are "active" threads blocking, the FORCE option allows DB2 10 for z/OS to kill the active threads.

Early beta customers found limited value to this function because if the threadswere active in DB2, DB2 would cancel the threads (good). But if the threads were inactive, the FORCE function did not kill them, and the online REORG failed. Then, when the inactive threads came back to life after the online REORG failed, the threads were canceled on their way back in. So the FORCE option is not a guaranteed way to kill all blocking threads and allow the online REORG to always make the switch.

Also new with DB2 10 for z/OS, the online REORG of LOB table spaces provides a DISCARD option. Early customers thought this feature was of limited value because it cannot handle LOB column values greater than 32 K.

Other Issues

First, there is the ability to create classic partitioned table spaces (PTS). In DB2 10 for z/OS, the classic PTS is now deprecated, meaning that, by default, you will not be able to create any new PTS. An attempt will be made to honor the request by creating a UTS PBR. However, a CREATE of UTS will support only the table-based controlled partitioning syntax. The legacy index-based control partitioning syntax is not supported for UTS.

So, by default, you may not be able to create any new, classic PTS. However, customers demanded the continued ability to create classic PTS because there are still a few areas where classic PTS has value over UTS.

The good news is that you can still create classic PTS in DB2 10 for z/OS, and these table spaces are still officially supported. There are two ways to continue to create classic PTS:

1. Specify SEGSIZE=0 on the CREATE TABLESPACE statement.
2. Set new system parameter (ZPARM) DPSEGSZ to zero (the default is 32).

Either of these methods will let you create classic PTS in DB2 10 for z/OS. For customers who still have old COBOL and PL/1 programs, the DB2 7 lookalike precompiler (DSNHPC7) for COBOL and PL/I is still provided in DB2 10 for z/OS.

The concurrency issues with parallel SQL DDL execution are not absolutely solved in DB2 10 for z/OS, despite the DB2 Catalog restructure. While the restructure was eventually successful for parallel BIND/REBIND activity, most customers still experience deadlocks when running parallel jobs with heavy SQL DDL against different databases within the same commit scope. Therefore, some customers will still have to run their SQL DDL jobs single-threaded.

BIND/REBIND issues

With single-thread BIND/REBIND, early customers have reported degraded CPU and elapsed time performance on entry into DB2 10 for z/OS CM. There are two reasons for this experience:

- PLANMGMT is now ON by default, and its default value is EXTENDED.
- New indexes defined for post–Enable New Function Mode (ENFM) processing, when hash links are eliminated, are being used even in CM.

Because we have a single code path (no dual path processing) across the different modes of DB2 10 for z/OS, those indices are now used even in Conversion Mode. For most customers, single thread BIND/REBIND performance remains important because there are no concurrency improvements until after the DB2 Catalog restructure is completed at the end of ENFM.

With parallel BIND/REBIND jobs, particularly in data sharing mode, we identified and addressed a number of concurrency and performance problems prior to general availability, including:

- Performance problems related to the repetitive DELETE/INSERT process
- Space growth in SPT01 for both LOB table spaces and base table spaces

The concurrency of parallel BIND/REBIND jobs is now working well. There are several relevant APARs:

APAR	Description
PM24721	Inefficient space search for out-of-line LOB in data shar
PM27073	Inline LOB with compression for SPT01 to address SPT01
PM27973	More efficient space reuse for base table and UTS

With these APARs applied, concurrent BIND/REBIND activity in data sharing mode works well after you get past ENFM processing.

Once beyond ENFM processing, we recommend that customers change existing procedures to run BIND/REBIND activity in parallel (but you should not do this until after ENFM). Doing so gives customer installations the opportunity to get back to and improve upon the elapsed time performance levels experienced in DB2 8 and DB2 9 for z/OS and to reduce application downtime when implementing new enterprise application releases.

Incompatible Changes

The most important incompatibility relates to the CHAR() scalar function. As an application programmer, you may want to use this function and apply it against a decimal column value to pull out a numeric value to assign to particular fields.

The incompatible change is documented in the install guide. The challenge for customers is how to identify what the rogue applications are that need be corrected. How do you identify what the exposure is? How can you support a phased migration?

By working with customers in the beta program, we were able to identify the issue. APAR PM29124 was created to restore the compatible behavior of pre–DB2 10 for z/OS, by default, for the CHAR() scalar function. In a subsequent APAR, wewill give you the capability to put on a new trace that will identify those applications that are potentially exposed and require investigation. You will then be able, at the individual package BIND level, to indicate whether you want the new behavior.

The next incompatibility issue is with SQL stored procedures. If you have a native SQL procedure that was implemented and/or regenerated under DB2 10 for z/OS and you need to fall back to DB2 9 for z/OS, that native SQL procedure will not run. The workaround is to run ALTER PROCEDURE REGENERATE on the DB2 9 for z/OS member. APAR PM13525 will deal with this issue automatically for you.

Finally, there is an issue with Create Trigger for triggers that are created on DB2 10 for z/OS. If you fall back to DB2 9 for z/OS, such triggers will not work. The workaround is to drop and re-create these triggers under DB2 9 for z/OS after fallback.

Migration and Planning Considerations

This section reviews key migration and planning considerations to take note of in planning for DB 10 for z/OS.

Migration Strategy

As in previous releases, we recommend a short time for mixed-release coexistence in data sharing. A short period for ENFM is also highly recommended. Support from vendors may affect the migration staging. One concern for CM is that some new performance improvements cannot be used.

The timing for moving from Test to QA to Production involves more options to consider. There are better controls for preventing the use of new functions, but a long gap between Test and Production levels is not advisable. You now have more granularity in the migration process and can move through mode by mode. Some customers migrate both Test and Production to CM and then change to New Function Mode (NFM) in a short time.

The chart shown in Figure 1.3 summarizes the history of DB2 releases. The top line tracks the year when each release became generally available (GA). The arrows show that the only releases where it was possible to skip a release were from DB2 5 to DB2 7 and from DB2 8 to DB2 10 for z/OS.

The lower part of the chart indicates the steps within the upgrade path from DB2 8 or DB2 9 for z/OS to DB2 10 for z/OS. The double-headed arrows indicate where you can "go back" a step, if necessary.

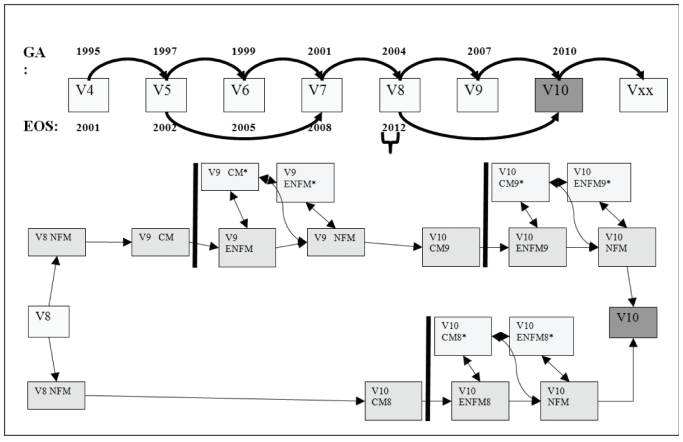

Figure 1.3. Timeline of DB2 releases and upgrade paths.

Note

If you are migrating from DB2 8, you have a decision to make. Should you go to DB2 9 for z/OS, or skip it and go directly to DB2 10 for z/OS? Once you decide to migrate to DB2 10 for z/OS CM8, you can still return to DB2 8. But you cannot then try to migrate to DB2 9 for z/OS CM.

Planning considerations

In general, the DB2 10 for z/OS migration process is very similar to that for both DB2 8 and DB2 9 for z/OS. It works well, with few customers experiencing problems with migration fallback. The ENFM process in DB2 10 for z/OS runs a lot longer than it did for DB2 9 for z/OS and even longer than it was on DB2 8.

You can migrate to DB2 10 for z/OS CM from either DB2 8 for z/OS NFM or DB2 9 for z/OS NFM. You cannot migrate through either of the following two scenarios:

- Once you migrate forward from V8 NFM to DB2 10 for z/OS CM8, you can always fall back to V8 NFM, but you cannot then migrate forward to DB2 9 for z/OS CM.
- Once you migrate forward from V8 NFM to DB2 9 for z/OS CM, you can always fall back to V8 NFM, but you cannot then migrate forward to DB2 10 for z/OS CM8.

Here are some important APARs to remember:

- Fallback Toleration SPE:
  - APAR PK56922
- Early Code for DB2 V8/V9:
  - APAR PK87280 (supersedes APAR PK61766)
- Information APARs:
  - II14474: V8 to V10
  - II14477: V9 to V10

If you are migrating from DB2 8 NFM, the bootstrap data set (BSDS) must be reformatted for the larger number of active/ archive log tracking.

For those who operate DB2 Connect™, the minimum level supported is DB2 9.1 FP1. DB2 9.7 FP3A is required to support the new DB2 for z/OS functions.

Many customers still use DDF Private Protocol under DB2 8 and DB2 9 for z/ OS. There is zero tolerance for DDF Private Protocol in DB2 10 for z/OS. You must absolutely eliminate all use of DDF Private Protocol before starting DB2 10 for z/OS in CM.

Many customers have local plans and packages (CICS, IMS™, batch, and so on) that have been accidentally mistagged as requiring the use of DDF Private Protocol. These plans and packages, which have been mistagged, will be tolerated. However, if any of these packages really do perform an external call that uses DDF Private Protocol, the call will be prevented and the application will fail immediately.

In DB2 10 for z/OS, database request modules (DBRMs) bound directly into plans are no longer supported. However, if any DBRMs bound into plans are found at execution time, DB2 will automatically trigger AUTOBIND to generate packages on first allocation after entry into DB2 10 for z/OS. We choose a standard collection name to put these packages in, but the recommended best practice is to deal with DBRMs bound directly into plans before migrating to DB2 10 for z/ OS. Any old plans and packages bound prior to DB2 V6 will also be invalidated and go through an AUTOBIND.

During ENFM processing on DB2 10 for z/OS, all the new indexes and new table spaces in the DB2 Catalog and Directory will be created as SMS-controlled, requiring extended addressability (EA) and extended format (EF) attributes. Some customers still do not use SMS management for the DB2 Catalog and Directory. Once you get to the ENFM process in DB2 10 for z/OS, the datasets of the DB2 Catalog and Directory must be SMS-managed.

For those of you coming from DB2 8, partitioned data sets extended (PDSEs)—as opposed to partitioned data sets (PDSs)—are required for SDSNLOAD, SDSNLOD2, and ADSNLOAD libraries.

The environment created by the DSNTIJSS job is only for DB2 Catalog and Directory data sets, which must be SMS-controlled in DB2 10 for z/OS. Other DB2 subsystem data sets, such as logs and the BSDS, are not accounted for in this environment.

The DSNHDECP module supports the NEWFUN parameter with the following options: V10, V9, or V8. This provides a way of stopping both static and dynamic SQL applications from using new SQL functions.

Many customers have old EXPLAIN table formats. DB2 10 for z/OS brings some changes in this space. First, if you have any plan tables that use a format prior to DB2 8, they will not work with EXPLAIN in DB2 10 for z/OS. The format and the ASCII/EBCDIC Coded Character Set Identifier (CCSID) from previous releases are deprecated in DB2 10 for z/OS. They will fail with an SQLCODE –20008. If you have plan tables in DB2 8 or DB2 9 for z/OS format, youcan still use them, but they will generate a warning SQLCODE +20520, regardless of whether they are CCSID EBCDIC or UNICODE.

If you use the DB2 10 for z/OS format, you must use UNICODE as the CCSID. If you try to use CCSID EBCDIC with DB2 10 for z/OS format, you will get the following errors:

1. EXPLAIN fails with RC=8 DSNT408I SQLCODE = –878.
2. BIND with EXPLAIN fails with RC=8 DSNX200I.

We recommend using the DB2 10 for z/OS extended format of the plan tables with a CCSID value of UNICODE. APAR PK85068 can help you migrate existing plan tables in DB2 8 or DB2 9 for z/OS table format over to the new DB2 10 for z/OS format with a CCSID of UNICODE.

Should you "skip" DB2 9?

If you decide to migrate from DB2 8 directly to DB2 10 for z/OS, you are, by definition, an early adopter of the new DB2 10 for z/OS release. This is because the end of support for DB2 8 is the end of April 2012. Quite clearly, the DB2 8 to DB2 9 for z/OS migration is the safer path to take because DB2 9 for z/OS has been in the field for almost four years and is quite stable.

Early customer adopters of DB2 10 for z/OS, whether migrating from DB2 8 or DB2 9 for z/OS, should expand their plans and take extra care to mitigate the risk of instability. This is not a statement of, nor an implication that, the DB2 10 for z/OS release has any endemic problems of instability. These same recommendations would apply to any release of DB2 or any other major software product.

First, you should perform application regression and stress testing to keep problems away from production. Next, plan to be proactive with regard to the continual application of preventive service maintenance. Plan to stay more current than two full, major preventive service maintenance drops per year. Regular, full, major preventive service maintenance drops, including HIPERs/ PEs, are essential and required for about a year.

We strongly recommend planning for four major preventive service maintenance drops in the first year, based on the quarterly RSU. Then, you can move to two major and two minor preventive service maintenance drops as the release passes through the early adopter curve. In between these drops, be vigilant and take advantage of the Enhanced HOLDDATA on a regular basis to find out which critical HIPERs/PEs are available.

One of the advantages of the CST/RSU process for recommended service maintenance, as opposed to the PUT route, is that it enables you to stay current on HIPERs/PEs that have gone through more testing but lets you stay further back on non-HIPERs/PEs maintenance. This capability provides some level of protection against PTFs in Error (PEs).

Finally, you have to be able to accept some level of risk and be able to handle some "bumps in the road" during the migration.

Security Considerations When Removing DDF Private Protocol

As previously mentioned, there is zero tolerance in DB2 10 for z/OS for DDF Private Protocol. Ahead of migrating to DB2 10 for z/OS, you need to plan for and work on eliminating all use of DDF Private Protocol and converting it to DRDA before you leave DB2 8 NFM or DB2 9 for z/OS NFM. There are fundamental differences in how authorization is performed, based on which distributed protocol you use and whether the protocols are used in combination.

Private Protocol is unique to the DB2 for z/OS requester and supports static SQL statements only. The plan owner must have authorization to execute all SQL requests executed on the DB2 for z/OS server. The plan owner is authenticated on the DB2 for z/OS requester and not at the DB2 for z/OS server.

Now, let us compare that with the DRDA Protocol. DRDA supports both static and dynamic SQL statements. The primary auth ID and associated secondary auth IDs must have authorization to execute both static SQL packages and dynamic SQL at the DB2 for z/OS server. The primary auth ID authenticated and secondary auth IDs are associated at the DB2 for z/OS server.

Until DB2 10 for z/OS, Private Protocol and DRDA protocols can be used by the same application within the same commit scope. You can "mix and match." Private Protocol security semantics are used due to possible inconsistent behavior, which is dependent on how the programs are coded and executed. That is a brief history of the differences between Private Protocol and DRDA Protocol.

Things have changed with APAR PM37300, which applies to DB2 8 and DB2 9 for z/OS. It provides control over the authorization checks performed when migrating from Private Protocol to DRDA Protocol. In DB2 10 for z/OS, Private Protocol security semantics are no longer used because the default is to use DRDA Protocol for access from a DB2 for z/OS requester.

DB2 8 and DB2 9 for z/OS will now use DRDA authorization checks and will use the DB2 system parameter PRIVATE_PROTOCOL to determine what security checks should be performed. This system parameter was previously introduced for customers to prevent new use of Private Protocol after all the previous use was eliminated. To do this, a customer would set PRIVATE_PROTOCOL to NO.

So, before you disable Private Protocol by setting PRIVATE_PROTOCOL to NO, ensure that all the appropriate grants are in place by granting execute privileges to any user who plans to run a package or stored procedure package from a DB2 for z/OS requester at the DB2 for z/OS server. It will be treated like any other DRDA client application at the DB2 for z/OS server.

Clearly, this is a major change that could have a big impact. To help customers migrate to DRDA Protocol and the changes in security checking, both DB2 8 and DB2 9 for z/OS still provide the option to continue to prevent the introduction of new Private Protocol requests, but now provide the option to continue to use the Private Protocol authorization checks. This is achieved by changing the setting of the DB2 system parameter PRIVATE_PROTOCOL from NO to AUTH.

Save critical access paths and accounting data

BIND REPLACE and REBIND activity can cause unwanted access path changes. You should identify important queries, plans, and packages. Be sure that plan tables contain access paths and costs. ALTER current plan tables to add new DB2 10 for z/OS columns. REBIND may change access paths, so extract plans and run REBIND with EXPLAIN under a dummy collection or a different application or program name.

Keep accounting reports for crucial queries and applications. If you have a problem and send in accounting layout long reports and the plan table data, we will be able to troubleshoot the problems more quickly. If you do not have the reports and the data, then we must guess.

Items Planned for Post-GA Delivery

The first item to mention is APREUSE and APCOMPARE. These features are introduced with APAR PM25679. These options of BIND REPLACE and REBIND provide a way to generate a new SQL runtime but, at the same time, ask DB2 10 for z/OS to give you the old access path wherever possible. So, if you have previously re-bound under DB2 9 for z/OS, this will mitigate the risk of access path change on the first BIND REPLACE or REBIND in DB2 10 for z/OS. Additional items planned for post-GA delivery include the following:

- In DB2 10 for z/OS, you will be able to delete a data sharing member. This function was introduced by APAR PM31009. Deletion of a DB2 member will require a quiesce of the data sharing group.
- Inline LOBs will be introduced for SPT01 to gain the benefits of data compression and improve BIND/REBIND performance. This function is introduced with APAR PM27811.
- Enhancements for new DBA authorities are introduced with APAR PM28296:
  - Prevent privileged users from stopping audit traces
  - No implicit system privileges for DBADM
- Online REORG concurrency for materializing deferred ALTERs is introduced with APAR PM25648.
- Temporal enhancements:
  - TIMESTAMP WITH TIMEZONE support (APAR PM31314)
  - Enhancement for data replication (APAR PM31315)
  - ALTER ADD COLUMN, propagate to history table (APAR PM31313).
- New system profile filters based on “client info” fields is introduced with APAR PM28500:
  - Three new columns for userid, appname, and workstation
  - Wildcard support: if column is ‘*’ then all threads pass that qualification.
- A new DB2 system parameter (ZPARM) to force deletion of coupling facility (CF) structures on group restart (APAR PM28925). This feature is aimed at disaster recovery. We want to avoid a situation during a disaster restart of using “stale” information in the CF structures. When the DB2 member starts and it is the first member to connect to the structure, it wipes out those structures and forces a group restart.
- Relief for the incompatible change in CHAR of decimal data by using APAR PM29124 to restore the previous behavior that existed prior to DB2 10 for z/OS.
- Real storage monitoring enhancements to be provided in APAR PM24723; this APAR also provides protection for over-commitment of available real storage.
- Hash LOAD performance (APAR PM31214)
- DSSIZE greater than 64GB (APAR yet to be announced).
- REORG REBALANCE SHRLEVEL CHANGE (APAR yet to be announced).

Note

RSM APAR OA35885 is a prerequisite to the enhanced storage monitoring capability provided by DB2 APAR PM24723. DB2 APAR PM24723 is strongly recommended for production use of DB2 10 for z/OS.

We strongly advise customers not to go into a major production environment without the proper monitoring of real and auxiliary storage usage as provided by this APAR and DB2 APAR PM24723. Together, these two APARs provide DB2 10 for z/OS with statistics on real and auxiliary storage use in relation to the 64-bit memory object allocated by DB2 for z/OS above the 2 GB bar.

DB2 10 for z/OS can request z/OS to provide information about real and auxiliary storage, based on a particular addressing range. It provides proper monitoring when you have multiple DB2 subsystems running on the same LPAR. It also provides some protection against the system paging, overcommitting real storage, or running out of AUX storage. DB2 can free unused memory back to the z/OS operating system.

When should you migrate to DB2 10 for z/OS?

A "normal" migration is moving one version at a time every three years. For customers with even earlier versions, the ability to skip a migration cycle will be attractive, but this ability is not "something for nothing." Customers need to consider the tradeoffs and challenges in a "skip version" migration. Most customers who migrate to a new version by three years after the general announcement (GA) of the respective new release are already on DB2 9 for z/OS.

The project for skipping a release is larger. While the testing and rollout are only a little greater than a single version migration, the education and remediation work is roughly double the size; most project plans estimate 150 percent. Consider the timing carefully. Improvements in DB2 9 for z/OS are delayed with a "skip" release migration plan. You may need to have extended service on DB2 8.

You will find more details about the "when to migrate" decision, the IBM Technote at http://www.ibm.com/support/docview.wss?uid=swg21006951.

In summary:

- We recommend the regular application of preventive service maintenance. It should be a continual process.
- Testing should be performed over and above that performed by DB2 for z/OS Development.
- CST testing still does not replace customer regression/stress testing.
- You must be prepared to tolerate some "bumps in the road."
- Customers who are not prepared to take mitigating actions and have no tolerance for "bumps in the road" should not be early adopters and should migrate directly to DB2 9 for z/OS.

For customers who are still running DB2 V7, the option to skip from DB2 8 to DB2 10 for z/OS is very attractive and makes the current path clear. Customers who have just migrated to DB2 8 may like this alternative, for the short term. DB2 10 for z/OS supports migration from DB2 9 for z/OS NFM or from DB2 8 NFM. Customers not yet running DB2 8 or DB2 9 for z/OS should plan to migrate first to DB2 for z/OS V8, as preparation for an eventual migration to DB2 10 for z/OS.

We estimate that about one in five customers migrated using a "skip version" technique from DB2 V5 to DB2 V7, and we expect to see a similar proportion this time. The savings for skipping a version migration are less than 50 percent, since the education and needed application and administration changes are about the same. Customers who do skip migration report that the project takes longer, about 50 percent longer than a normal migration path.

Changing from DB2 8 or earlier to DB2 10 for z/OS will require a cultural shift that some describe as "culture shock." If customers spend the bulk of their migration project time in testing, savings could be up to 40 percent. But most customer plans should expect 20 to 25 percent reduction, compared with two migrations.

The tradeoff for skipping is primarily the later delivery of DB2 9 for z/OS improvements, namely CPU savings, especially in utilities and disk savings via compression for indexes, improved insert and update rates, improved SQL, and pureXML for developer productivity, as well as better availability.

Summary

To summarize, DB2 10 for z/OS is a very good release in terms of the opportunities for price/performance and scalability improvements. There is significant DBM1 31-bit VSCR after rebind as soon as DB2 10 for z/OS CM. You can use 1 MB size real storage page frames on z10 and z196 processor. But the key is the use of long-term page fix on local buffer pools. There are also improvements in terms of reduced latch contention and latch management overhead.

Over and above the "out-of-the-box" performance improvements as a result of BIND/REBIND and the use of 1 MB size real storage page frames, there are opportunities for further price/performance improvements *provided you have enough real memory*. It is a classic tradeoff between increased real storage provision in order to reduce CPU resource consumption. This includes making more use of persistent threads both for legacy CICS and IMS/TM applications, as well as the use of high-performance DBATs for DDF workloads.

If you have enough real memory, you can make greater use of the BIND option RELEASE(DEALLOCATE) with these persistent threads. But you must recognize that increased use of the BIND option RELEASE(DEALLOCATE) is a tradeoff; it will lead to increased storage consumption, and you will need to plan for additional real memory over and above the required 10 percent to 30 percent increase just to stand still when migrating to DB2 10 for z/OS.

The use of the BIND option RELEASE(DEALLOCATE) with persistent threads can also reduce concurrency because BIND/REBIND and SQL DDL activity will not be able to break in to work.

DB2 10 for z/OS also provides opportunity for the greatly enhanced vertical scalability of an individual DB2 member in data sharing and the potential for LPAR/DB2 consolidation.

You must carefully plan, provision, and monitor real storage consumption. Early customer adopters of DB2 10 for z/OS, migrating from either DB2 8 or DB2 9 for z/OS, should make plans and take extra care to mitigate the risk of instability. Those steps include:

- Plan regular full "major" maintenance drops.
- Use CST/RSU recommended maintenance.
- Perform application regression and stress testing to keep problems away from production
- Be prepared to tolerate some "bumps in the road."

IBM Global Services
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
October 2011

IML14294-USEN-00