How DB2 for z/OS Can Help Reduce Total Cost of Ownership

by Cristian Molaro

How can IBM® DB2® for z/OS® features reduce total cost of ownership (TCO)?

Total cost of ownership is a financial estimate that helps businesses determine the total cost of deploying a database over its life cycle, including software, hardware, and training. TCO analysis was popularized by the Gartner Group in 1987.

The goal of the following sections is to provide guidelines, orientation, and information about the elements that occur in the total cost of ownership of a DB2 for z/OS environment and to help readers explore ways to improve DB2 TCO in their existing environments. We start by setting the context in terms of today's business needs and how DB2 for z/OS can help. The following topics are then covered from a TCO saving point of view: synergy with System z®, CPU savings, better performance, storage savings, faster analytics, and improved scalability.

Business Needs and DB2 TCO

In 2011, the International DB2 Users Group (IDUG) conducted a worldwide study to understand the main IT concerns among database administrators, managers, and decision makers involved with DB2. Part of this study was a survey that gathered more than 1,100 answers from active IT professionals involved with the DB2 family of products on all platforms, but mostly on z/OS.

Figure 1 shows the distribution of the top answers to the survey question "What are the main concerns of your IT business?" The results reveal the two main concerns to be *Availability & Reliability* and *Reduce Cost*, followed by *Improve Performance* and then *Security*.

Availability & Reliability can be described as the need to assure business continuity. This aspect can be related to concerns regarding the guarantees of business survival. In today's highly competitive world, a relatively short IT outage can mean significant losses in both money and reputation. A serious outage can result in an organization being out of business permanently.

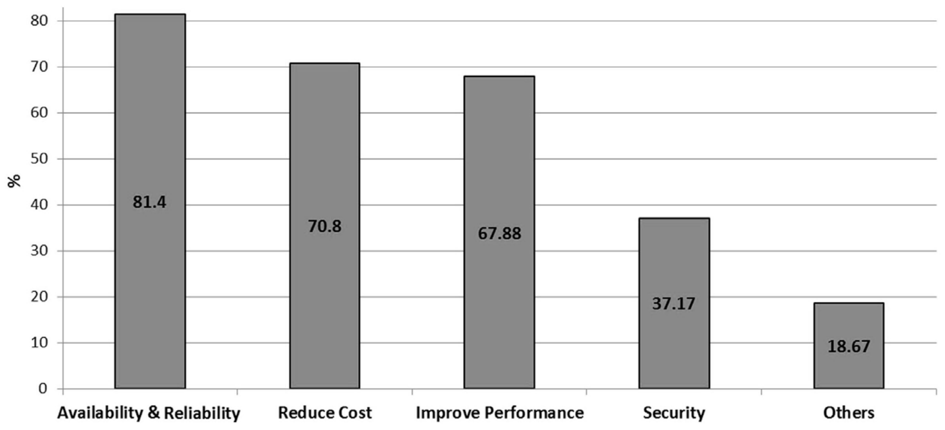


Figure 1: Top concerns of IT business

Reduce Cost also closely relates to organizational survival. The ability to provide the same or better service at a reduced total cost of ownership has been, is, and will continue to be one of the main objectives for any business evolving in today's competitive market. This point has been particularly true in recent years, during which the business continuity of many organizations had to be assured through cost reduction campaigns.

Improve Performance can be understood as the need to achieve faster response time and higher throughput to allow applications to cope with exigent and ever-growing business requirements. In many cases, good performance is a necessity for company survival and sustained growth. Improving performance relates closely to reducing costs, too.

Security has been a key DB2 industry concern in the past decade. Organizations have invested in seeking and applying technical solutions and policies to improve security. Security risks have tended to increase due to the inherent complexity and openness of Internet-exposed applications. DB2 security is becoming multi-layered, and security must be implemented at the physical, network, host, and data level. Security centralization is a trend today, and a common practice is to explore the potential of externalizing the otherwise internal DB2 security definitions.

That availability and reliability, cost reduction, performance improvement, and security are the main IT concerns is no surprise. These issues are the basis of the business continuity and business optimization definitions in the mature DB2 for z/OS industry.

How important the focus on each one of these concerns is varies with each organization. Each user differs, and so does its environment, context, and challenges. Nevertheless, reducing costs tends to be a common concern. Organizations face the challenges of aggressive business demands in terms of delivering results and better performance, but with IT budgets that tend to be under stress. IT organizations must deliver *more* business value with *fewer* resources. The ability to improve performance, increase productivity, and guarantee availability with a reduced total cost of ownership is, unarguably, a great asset.

DB2 for z/OS has been improving to help in this context. Recent releases have added functionality and features that can effectively help users improve the way they do business at a lower cost.

In this context, today's business needs are changing. We see our IT world evolving toward a near real-time, huge-amount-of-data, analytics environment. The mobile world, with its technical needs, is reaching many otherwise classic mainframe environments. A common consequence of the proliferation of mobile and other network-connected devices is an increase in online transaction processing (OLTP) workloads characterized by 24x7 availability requirements. These new workload demands are often associated with the generation of massive amounts of data and with fast analytics requirements. The market competition is highly competitive in many industries and services where DB2 is the core database. A strong need exists for fast analytics, online predictive analysis, data cleansing, and data integration, just to mention some of today's more discussed data-related topics.

DB2 for z/OS and System z provide the right platform to expand an already existing application environment to be able to cope with these new requirements. Thanks to DB2's synergy with the latest IBM zEnterprise® EC12 (zEC12) series of System z servers, the platform is able to deliver industry-leading levels of scalability, high availability, multi-tenancy, and ability to run mixed workloads.

DB2 11 for z/OS enhances DB2 analytics on System z with big data by providing connectors and the database capability that lets DB2 applications access data stored in the big data distributed file system Hadoop. The IBM DB2 Analytics Accelerator allows eligible DB2 for z/OS queries to be offloaded to an appliance to be executed fast, and sometimes quite fast.

Faster analytics, cost reduction through offloaded workloads, and straightforward deployment are accelerator characteristics that have the potential to make DB2 for z/OS and System z the best option for analytics.

DB2 and TCO

DB2 for z/OS celebrated its 30th announcement-date anniversary in 2013. When released, DB2 was intended to serve as a data warehouse database. With time and development, DB2 has evolved to become a state-of-the-art OLTP and data warehouse database engine. DB2 accelerators integrate with existing DB2 for z/OS servers to make DB2 a hybrid database. DB2 is able to deliver excellent transactional and analytic performance.

User requirements have a significant influence on the evolution of DB2 for z/OS and its features. As no exception, the strong industry need for cost reduction is guiding DB2 development toward a more cost-effective solution.

At the same time, the DB2 industry requires the usual level of reliability, high performance, and high availability that makes DB2 for z/OS and System z a business-critical, capable platform. New application development paradigms require features that enable DB2 to further support distributed access to DB2, to support new SQL, and to enable the new world of big data and analytics workloads. Helping users get more functionality makes DB2 a better but, ineluctably, more complex database.

The twin objectives of reducing costs and gaining functionality from DB2 pose an apparent conflict. Nevertheless, DB2 manages the challenges remarkably well. DB2 10 for z/OS provides a new, rich set of functionality together with CPU reductions "out of the box."

Often, the DB2 application environment hides savings opportunities that are readily accessible to DB2 users. The application of a non-used DB2 technique or a simple but forgotten best practice can help users achieve better results and reduce costs at the same time. In other cases, optimizations are the result of planning and applying DB2 functionality that requires application or system changes.

When looking for cost-saving opportunities, the main areas to investigate for potential benefits are:

- **Synergy with System z.** DB2 leverages and exploits the latest System z enhancements. This synergy may help to reduce TCO by providing better performance and more scalability to existing applications.
- **CPU savings.** CPU reductions can help to reduce TCO by lowering the software and licensing costs related to CPU utilization and by letting existing applications deliver more throughput as a consequence of reduced CPU utilization. CPU reductions can delay the need for a capacity upgrade and promote opportunities for server consolidation.
- **Better performance.** Performance enhancements unrelated to CPU improvements can help to increase application throughput, application availability, and the capacity to absorb peaks of application demand.
- **Storage savings.** In the context of large amounts of information, data compression and other techniques can dramatically reduce costs related to storing data.
- **Faster analytics.** Current business demands require complex analytics against large amounts of data. DB2's faster analytics capabilities help business to react quickly to changing conditions with reduced operating cost.
- **Improved scalability.** DB2 scalability enhancements allow better performance and throughput for existing applications. They may offer server consolidation opportunities, as well.

Reducing TCO Through Synergy with System z

Business demands and the requirements of fast-moving markets are putting data centers under stress. There is a need for smarter computing systems that innovate on efficiency, cost savings, and performance while lowering management costs and complexity.

The system design of the IBM zEnterprise System z addresses the complexity and efficiency requirements found in today's data centers. With its superscalar design, the zEC12—IBM's latest System z enterprise server—delivers a record level of capacity over prior System z servers. At its high-end configuration, the zEC12 is powered by 120 microprocessors running at 5.5 GHz and can execute more than 78,000 million of instructions per second (MIPS).

A Forrester study prepared for IBM, *The Total Economic Impact of IBM System z*, examined the total impact and possible return on investment that enterprises may realize by deploying IBM System z. In-depth interviews conducted with executives of five IBM System z customers revealed the following benefits experienced by the interviewed companies:

- **Reduction in server administration staff.** A centralized enterprise environment enabled improved staff efficiency, and cost-related reductions, compared with a distributed server environment.
- **Avoidance of costs related to maintaining a distributed platform infrastructure.** Fewer servers translated to reduced infrastructure costs, especially those related to space, power, and cooling.
- **Reduced storage cost.** System z's effective data-compression capabilities enabled reduced cost of storage. One organization reported saving up to 68 percent in storage when moving away from a distributed platform.
- **Cost saving from reduced downtime.** Companies were able to obtain a better level of availability for mission-critical applications, reducing the risk of financial penalties and image damage caused by downtime.
- **Better flexibility.** Organizations could respond more quickly to business needs and gain flexibility by exploiting the additional System z capacity.
- **Improved business credibility.** Some organizations reported that the implementation of System z improved their credibility with business partners.

DB2 Synergy with System z

DB2 for z/OS takes advantage of the latest improvements in the System z platform. With each new version, DB2 increases the synergy with System z hardware and software to provide better performance, resilience, and function for an overall improved value with a potential TCO reduction.

DB2 11 for z/OS leverages System z advances in large real memory support, faster processors, and better compression. It increases specialty engine eligibility workload, driving faster processing, improved performance, and CPU savings.

With zEC12, DB2 11 provides additional CPU reductions through the use of pageable large 1 MB real storage pages and Flash Express, and with the support of 2 GB real storage pages. DB2 for z/OS takes advantage of the following zEC12 features:

- **Faster CPU.** zEC12 provides faster CPU than previous System z processors, with a measured 1.25 percent faster speed than the previous z196 generation. The faster zEC12 processing capacity can provide these DB2 performance improvements over the z196 CPU:
  - 20 to 28 percent CPU reduction for OLTP workloads
  - 25 percent CPU reduction for query and utility workload
  - 1 to 15 percent less compression overhead with DB2 data
- **More system capacity.** zEC12 provides up to 50 percent more total capacity than the z196, making the zEC12 an excellent choice for fast-growing businesses and markets that need to fulfill ever-increasing processing capacity requirements.
- **zEC12 hardware features.** DB2 11 exploits hardware features of the zEC12 server, such as:
  - *Large frame area.* The large frame area is used for the 1 MB and 2 GB large real storage pages. Using large pages can improve performance for some applications by reducing the overhead of dynamic address translation, providing CPU savings.
  - *Flash memory and pageable 1 MB frames.* The zEC12 supports optional Flash Express memory cards. Access to flash memory is faster than access to disk but slower than access to main memory. Flash Express and z/OS 1.13 allow DB2 to allocate internal control blocks using 1 MB pageable storage, providing performance improvements.
  - *2 GB large pages.* A 2 GB page is a memory page that is 2,048 times larger than a 1 MB large page. DB2 exploitation of 2 GB pages improves performance by decreasing the number of translation lookaside buffer misses, by reducing the time spent converting virtual addresses into physical addresses, and by reducing the real storage used to maintain some internal structures.

IMPORTANT: With each new release, DB2 for z/OS leverages and exploits the new functions and capabilities introduced in the latest System z platform. DB2 11 for z/OS is no exception in providing synergy with the latest System z zEC12.

Reducing TCO Through CPU Savings

Reducing the CPU used by the DB2 subsystem or DB2 applications may result in a financial benefit. To estimate the potential financial impacts of a DB2 CPU reduction, you must combine and analyze information from different sources, such as your application CPU profile, the estimated DB2 savings for a given workload, and an understanding of TCO that relates to CPU utilization financial charges.

As an example, an estimation approach consists of the following steps:

1. Model the overall CPU utilization for a logical partition (LPAR), including DB2 and non-DB2 workloads.
2. Plot the CPU utilization and millions of service units (MSU) 4-hour rolling average.
3. Identify the MSU 4-hour rolling average peak and how it influences software charges.
4. Model the DB2-related CPU utilization (both system- and application-related).
5. Model the DB2 CPU savings.
6. Project the DB2 CPU savings on overall CPU savings.
7. Determine the DB2 CPU savings in the MSU 4-hour rolling average.

Figure 2 depicts this process. At a glance, a DB2 CPU reduction that results in a reduction of the monthly peak MSU 4-hour rolling average may result in an immediate financial benefit.

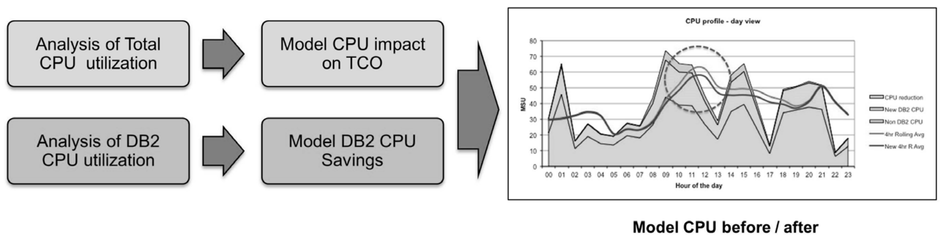


Figure 2: Estimating the financial impacts of a DB2 CPU reduction

A reduction in overall CPU utilization may impact licensing costs of non-IBM software, as well. Contact the independent software vendor (ISV) to obtain information about software charges and CPU utilization details.

How a given DB2 CPU reduction will impact the total cost of ownership is closely related to the System z software pricing model in use. Software pricing is a complex topic, and understanding it may require expert advice. The IBM webpage "IBM System z Software Pricing" at *www.ibm.com/systems/z/resources/swprice* provides an overview of IBM's mainframe software pricing, news about changes to IBM's mainframe software licensing/pricing, and downloadable tools related to System z software pricing.

The System z Software Pricing is the framework that defines the pricing and the licensing terms and conditions for IBM software that runs in a mainframe environment. An IBM Customer Agreement (ICA) contract provides the framework for the Monthly License Charge (MLC), which includes license fees and support costs that apply to IBM software products such as CICS®, DB2, IMS™, OS/390®, WebSphere® MQ, and z/OS. Software-related costs are measured by MLC pricing metrics such as:

- Advanced Workload License Charges (AWLC)
- Advanced Entry Workload License Charges (AWLC)

- Workload License Charges (WLC)
- Entry Workload License Charges (EWLC)
- Midrange Workload License Charges (MWLC)
- System z New Application License Charges (zNALC)
- zSeries Entry License Charges (zELC)
- Parallel Sysplex® License Charges (PSLC)

The MLC pricing metric is based on customer choice and/or the mainframe environment. You need to understand what the MLC pricing metric is to be able to model what the financial impact of a DB2 CPU reduction would be. Understanding how the type of MLC metric works will enable you to model the monthly license charges applicable to MLC products, such as z/OS, z/TPF, z/VSE z/VSE, middleware, compilers, and selected system management tools and utilities. Not all the IBM software running on your mainframe is necessarily an MLC product.

Organizations working with MLC metrics based on CPU utilization can benefit from immediate monthly license charges reductions as a consequence of reducing DB2 CPU.

Under sub-capacity workload license metrics, such as AWLC or WLC, the software charges are calculated based on the 4-hour rolling average CPU utilization per z/OS LPAR observed within a one-month reporting period. This information is obtained by the IBM-supplied Sub-Capacity Reporting Tool (SCRT) after processing of the related System Management Facilities (SMF) records.

Figure 3 is a representation of the CPU utilization per hour in a typical business day for a financial institution. The line in the figure illustrates the 4-hour rolling average for this LPAR.

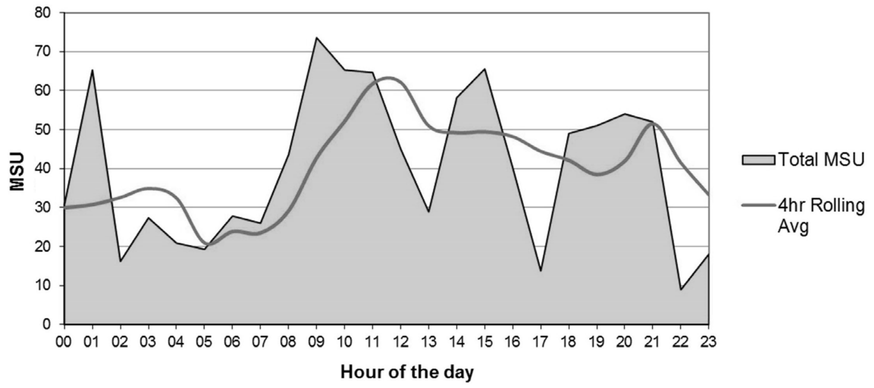


Figure 3: Hourly CPU utilization for a financial institution

The workload represented in the figure follows the common pattern of many other businesses, and we can identify two distinct workload patterns in the graph:

- During business hours, typically from 8 a.m. to 5 p.m., most of the workload is executed via transactions. This type of workload is commonly referred to as *online transaction processing, or OLTP*. An OLTP workload is typically characterized by many short, CPU-intensive transactions. This time period is also commonly referred to as *prime shift*.
- Outside of business hours, from 5 p.m. to 8 a.m. in this example, most housekeeping jobs are executed. These jobs or batches can be long-running processes designed to process large amounts of data. This time period is also commonly referred to as the *night shift* or *batch window*.

By combining the SMF and the DB2 accounting and statistics records, it is possible to model the percentage of total CPU utilization consumed by DB2-related processes.

The DB2 CPU used by applications can be obtained from the DB2 accounting records, and the DB2 CPU used by DB2 address spaces can be extracted from the DB2 statistics records. The total DB2 utilization is calculated as follows:

$DB2_{CPU} \approx DB2_{CPU\_APPLICATIONS} + DB2_{CPU\_ADDRESS\_SPACES}$

The non-DB2 CPU is calculated as

$Non\_DB2_{CPU} \approx Total_{CPU} - DB2_{CPU}$

For the same example, Figure 4 shows the DB2 and non-DB2 CPU distribution and the peak MSU 4-hour rolling average.

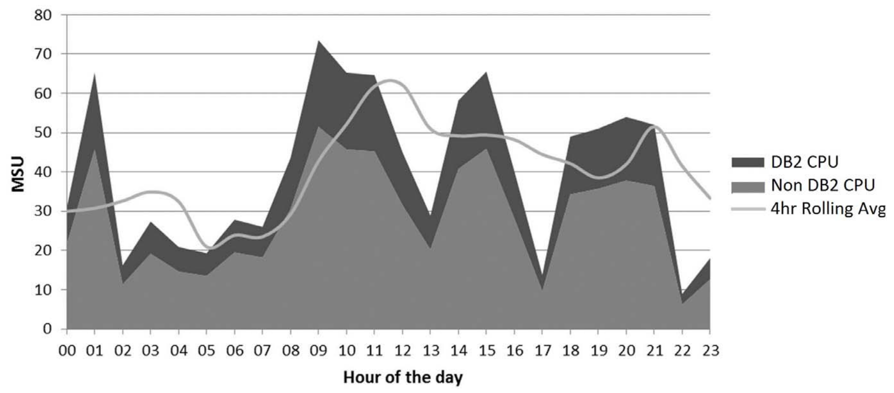


Figure 4: DB2 and non-DB2 CPU distribution and peak MSU 4-hour rolling average

Assuming a 20 percent reduction in DB2 CPU time, the new DB2 CPU can be modeled as follows:

$New\_DB2_{CPU} \approx (DB2_{CPU\_APPLICATIONS} + DB2_{CPU\_ADDRESS\_SPACES}) * (1 - DB2_{SAVINGS})$

As the CPU reduction pertains to DB2, the non-DB2 CPU utilization remains unaltered. As a result, the total CPU utilization can be recalculated as:

$Total_{CPU} \approx Non\_DB2_{CPU} + New\_DB2_{CPU}$

The new total CPU allows us to obtain a recalculated MSU 4 hour rolling average, as shown in Figure 5.

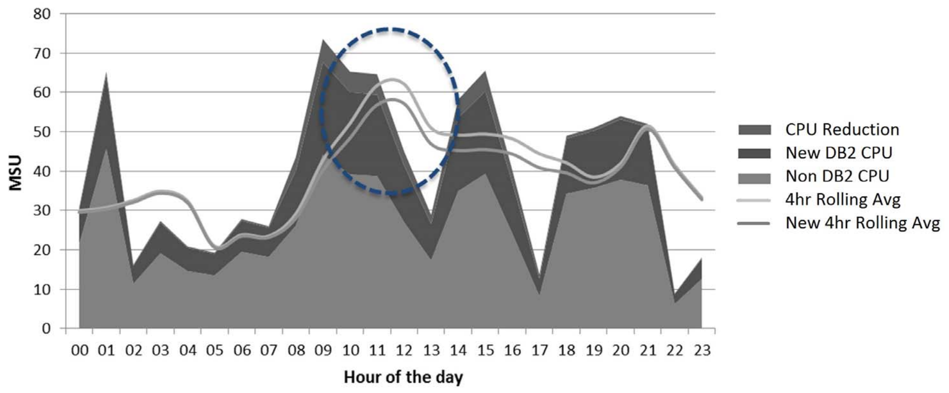


Figure 5: Recalculated MSU 4-hour rolling average

In this example, in which only 30 percent of the total CPU in the LPAR was DB2-related, a DB2 CPU reduction of 20 percent leads to a reduction of 8 percent in the peak MSU 4-hour rolling average. This reduction may be reflected in the monthly license charges.

The calculated percentage may or may not be linearly applicable to eligible software licenses; the unitary MSU price is grouped by cumulative monthly pricing levels as defined in the Advanced Workload License Charges Structure. At a glance, the more MSUs used, the less expensive they are. The focus must be the peak period. A CPU improvement, or a CPU regression, that does not affect the monthly MSU peak 4-hour rolling average may be, at least in this context, transparent from a financial point of view.

DB2 10 CPU Savings and Performance Improvements

When migrating from DB2 9, you start from DB2 9 New Function Mode (NFM). The first stage in DB2 10 is DB2 10 Conversion Mode (CM9); from here, you can fall back to DB2 9 if required. The next steps are DB2 V10 Enabling New Function Mode (ENFM9) and DB2 10 New Function Mode (NFM). DB2 10 also supports skip release migration, letting you migrate directly from DB2 8 NFM. Figure 6 illustrates these migration paths.

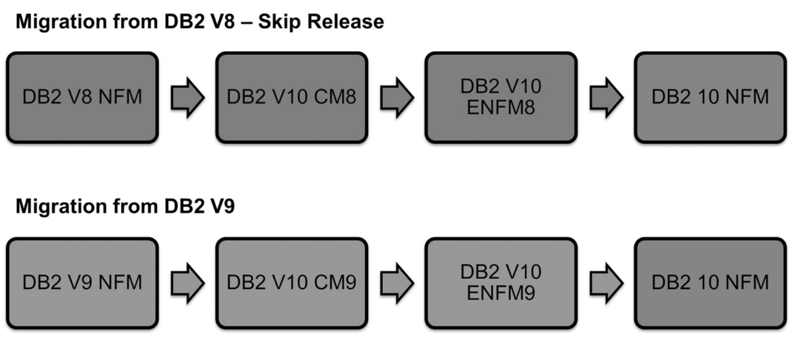


Figure 6: DB2 10 for z/OS migration paths

The maximum performance potential is available in DB2 10 NFM. However, because DB2 10 delivers many of its performance benefits in Conversion Mode, the first migration step is also interesting from a performance point of view. Some of the most relevant DB2 10 performance features grouped by migration step are listed in this section.

- DB2 10 Conversion Mode:
  - Improved performance of SQL at runtime
  - Faster single-row retrievals via open-fetch-close chaining
  - Parallel index update at insert
  - Query parallelism improvements
  - Workfile in-memory enhancements
  - Insert improvement for universal table spaces (UTS)
  - Index list prefetch
  - Memory changes exploiting more 64-bit storage, some after REBIND
  - Increased Distributed Data Facility (DDF) performance (high-performance database access threads, or Hiper-DBATs)
  - Buffer pool enhancements
- DB2 10 New Function Mode:
  - Efficient caching of dynamic SQL statements with literals
  - Faster fetch and insert
  - SQL Procedural Language (PL) performance improvements
  - MEMBER CLUSTER for universal table spaces (UTSs)
  - Utility enhancements

- DB2 10 New Function Mode features requiring changes:
  - Hash access path
  - Index include columns
  - Inline large objects (LOBs)

A detailed description of these features is beyond the scope of this document. Functional details are available in the IBM Redbook *DB2 10 for z/OS Technical Overview* (SG24-7892). Performance considerations and observations are reported in the IBM Redbook *DB2 10 for z/OS Performance Topics* (SG24-7942).

DB2 10 Performance Expectations

Because each organization, each application, and each environment is unique, it is not practical to design a single standard methodology that can forecast how much resource utilization improvement, or savings, you can get from any new DB2 version. Most workloads may show up to 10 percent CPU reduction for static SQL after REBIND. You may observe an even greater improvement with workloads that had scalability issues in previous versions of DB2 or in distributed applications targeting DB2 for z/OS and exploiting DRDA and dynamic SQL.

Some workloads are particularly susceptible to showing substantial savings. For example, those exploiting native SQL PL procedures can see up to 20 percent CPU reduction. Query workloads will also show significant improvements as a result of many positive access path changes.

Figure 7 shows the results obtained by IBM under controlled and repeatable workload conditions.

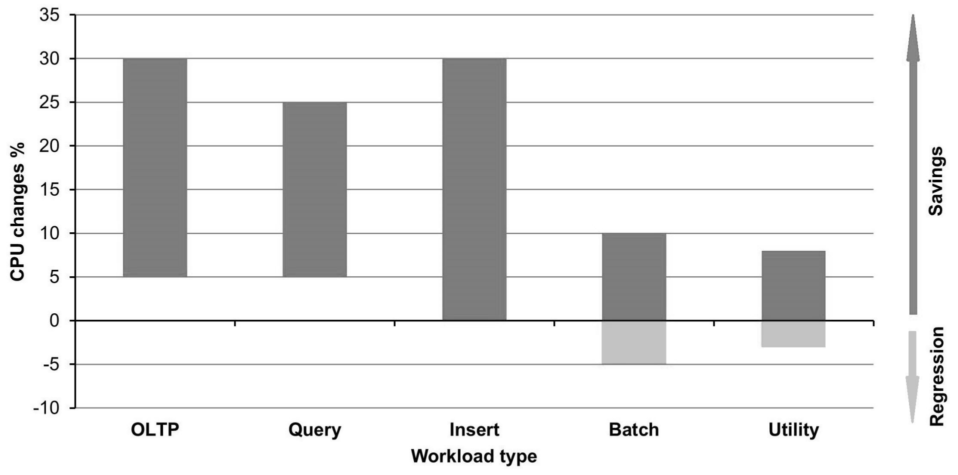


Figure 7: DB2 9 to DB2 10 migration and CPU changes – IBM benchmarks

These results are grouped by workload category. Positive changes, expressed as percentages, indicate CPU savings. Negative values indicate CPU regression. In most cases, the benchmarks were based on customer data, and these results were almost always confirmed by the customer running DB2 10.

The tests were done under controlled conditions and are repeatable, a requirement for quality statistical analysis. The benchmark results enforce the concept of variance in the expected savings: CPU improvement is workload-dependent, and your results will also vary.

DB2 11 CPU Savings and Performance Improvements

The migration to DB2 11 starts in DB2 10 NFM. There is no support for skip migration; you cannot migrate to DB2 11 from DB2 9 (Figure 8). In terms of performance enhancements, DB2 11 focuses on two areas: CPU and cost reduction and scalability.

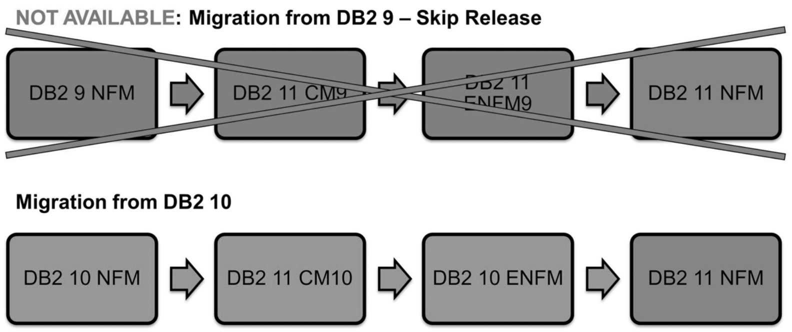


Figure 8: DB2 11 migration path

Many changes target areas that are commonly a source of performance concerns. For example, DB2 11 focuses on providing consistent performance to applications that change data, and it reduces the need for REORG utilities in some scenarios. DB2 11 has the ability to react to workload changes, requiring fewer efforts in performance tuning.

DB2 11 system performance changes that help to reduce CPU utilization include:

- Additional ad hoc machine code generation is provided for repeated operations such as SQL column processing (column proc) and RDS sort operations (sort proc).
- A new data decompression routine provides significant CPU reduction and speeds up expansion operations.
- DB2 log large RBA/LRSN support removes LRSN spin in data sharing environments. This change can provide significant CPU reduction in batch write operations.
- The DB2 output log buffer is moved to 64-bit common storage, reducing CPU by removing cross address space operation for logging. IBM reports significant CPU reduction due to this enhancement in update-intensive batch jobs.

- Data sharing availability and performance improvements, including:
  - Improved CASTOUT performance and faster CASTOUT
  - Conditional activation of group buffer pool (GBP) write-around makes pages be written directly to DASD
  - Coupling Facility DELETE NAME enhancement
  - Internal Resources Lock Manager (IRLM) enhancements

Two of the main DB2 performance changes that provide better scalability are:

- Internal DB2 processing units (zProcs) are moved above the 2 GB storage bar, providing further reduction of DBM1 virtual storage utilization below the bar. z/OS 1.13 made possible 64-bit code execution and code optimization for zProcs.
- Scalability improvement with a large number of partitions improves the performance of packages bound with the RELEASE(COMMIT) option.

DB2 11 application-level performance improvements include the following:

- When EXCLUDE NULL KEYS is specified in CREATE INDEX and ALTER INDEX operations, DB2 11 will not create an index entry for key columns with NULL. Benefits include reduced cost of index maintenance, potential FETCH performance improvement, and reduced index space utilization.
- The DECLARE GLOBAL TEMPORARY TABLES statement adds support for NOT LOGGED. This change can provide significant CPU and elapsed time reductions in intensive insert and update operations. It also provides significantly faster rollback.
- SORT improvements extend the scope of the in-memory sort operations in DB2 11. Changes can reduce workfile physical usage for large top-level sort, lowering CPU and elapsed time for sort operations.

Specialty engines provide a highly effective way to reduce CPU costs by offloading a part of CPU execution from a general-purpose processor to a special processor. DB2 11 continues the trend of increasing the zIIP value by expanding DB2 CPU zIIP eligibility for utility and system tasks workloads. (The value of zIIPs is discussed later.)

DB2 11 Performance Expectations

DB2 10 delivered strong DB2 CPU savings that drove great value for IBM users. DB2 11 follows up with even more CPU savings and improved performance. IBM's internal testing and Early Support Program results show that, depending on the specific workload, clients may achieve "out-of-the-box" DB2 CPU savings of up to 10 percent for complex OLTP workloads and up to 10 percent for update-intensive batch workloads. Complex reporting queries can see up to 25 percent CPU savings for uncompressed tables and up to 40 percent when running queries against compressed tables. These results are compared with running the same workloads on DB2 10. Additional CPU savings and performance improvements may be possible with application and/or system changes that take advantage of new DB2 11 enhancements such as log replication capture, data sharing using the extended log record format, and pureXML.

As with each DB2 release, some workloads are particularly susceptible to showing significant CPU savings in DB2 11. These workloads are often referred to as the performance "sweet spots."

In DB2 11, write-intensive batches are an excellent candidate to show CPU savings. For OLTP workloads, better performance improvements are expected for workloads involving write-intensive transactions. Transactions using RELEASE(COMMIT) and involving objects with many (more than 200) partitions are excellent candidates to obtain CPU benefits.

Distributed applications featuring workloads with heavy network traffic may show CPU benefits as a result of the synergy between DB2 11 and changes in z/OS Communications Server.

Performance sweet spots for query workload (e.g., data warehousing SQL) include those working with compressed tables and those exploiting SORT intensively. SQL access path enhancements can provide further CPU savings. Queries retrieving large result sets from an IBM DB2 Analytics Accelerator are expected to perform better in DB2 11.

Specialty Engines

An IBM System z server can be configured with optional processors known as s*pecialty engines (SEs)*. SEs are slightly modified standard processors that are designed to offload eligible workloads from general-purpose (GP) processors. Some of the CPU processing that would otherwise be executed on GP processors can, under certain conditions, be executed on an SE.

At the time of this writing, four specialty engines are available:

- System z Integrated Information Processor (zIIP)
- System z Application Assist Processor (zAAP)
- Integrated Facility for Linux® (IFL)
- System Assist Processor (SAP)

z/OS zIIPs can help to dramatically reduce the total cost of ownership for System z mainframes because the workload executed on them is not taken into account for software pricing. In addition, the purchase price of a zIIP is typically significantly lower than that of a standard processor.

zIIPs were introduced in 2006 as part of System z9® hardware, designed to work with Version 8 of DB2 for z/OS. Since then, the number of DB2-eligible workloads has grown steadily. DB2 10 for z/OS and DB2 11 for z/OS provide several zIIP-related improvements. DB2 for z/OS is not the only software that can leverage the ability to offload CPU to zIIPs; many software products from IBM and other vendors can take advantage of these specialty engines, as well.

DB2 10 and Specialty Engines

DB2 10 extends the scope of the zIIP-eligible workload, increasing the potential to reduce TCO through the following improvements:

- The ability to offload 100 percent of prefetch and deferred write engines. This enhancement is significant for index compression and insert index I/O parallelism.
- The ability to offload 99 percent of RUNSTATS CPU (with no additional parameters).
- In z/OS V1R11, DFSORT support for additional zIIP redirect for DB2 utilities.
- Improvements to the parsing process of XML schema validation, including eligibility of 100 percent of the new validation parser and the ability to offload to zIIP, zAAP, or zAAP on zIIP.
- Increased zIIP eligibility for DRDA® workloads, from 53 percent to 60 percent.
- Certain DBM1 processes.
- Prefetch I/Os (reported as DBM1 service request block, or SRB).
- Deferred write I/Os (reported as DBM1 SRB).
- Stored procedures written in SQL PL.

Native SQL stored procedures, introduced in DB2 9, are an excellent performance choice. They perform faster that WLM address space established stored procedures. Native SQL stored procedures are zIIP-eligible, but only when called from a distributed application using DRDA and TCP/IP.

DB2 11 and Specialty Engines

DB2 11 provides more zIIP exploitation by increasing the zIIP offload for system task SRB processing in the MSTR and DBM1 address spaces. The zIIP eligibility for IBM utilities is increased as well: RUNSTATS with distribution statistics up to 80 percent and an additional 30 percent for inline statistics.

Estimating zIIP Savings

A zIIP processor is not required to evaluate its potential benefits. The SYS1.PARMLIB parameter PROJECTCPU=YES enables z/OS to collect specialty engine usage as if a specialty engine were installed. This projection capability can be run at any time, including in a production environment. PROJECTCPU can help to determine the number of zAAP and zIIP engines required to satisfy a specific customer's workload.

With the projected usage information, you can identify which portion of the CPU would be executed on a zIIP processor. For example, consider the daily CPU profile for the data warehouse LPAR shown in Figure 9. This LPAR does not have zIIP engines.

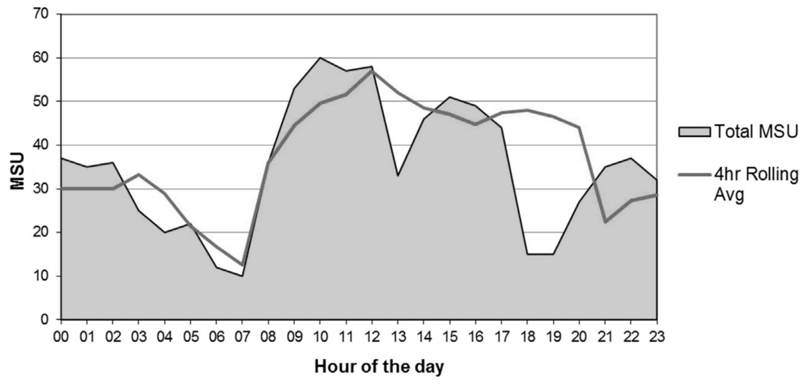


Figure 9: Daily CPU profile for a data warehouse LPAR

This chart is built using the SMF record Type 72 (RMF workload activity), and it shows the CPU used by the entire workload, DB2 and non-DB2, running in the LPAR. After activation of the PROJECTCPU z/OS parameter, the same records contains the details of the specialty engine eligible CPU executed in a general-purpose processor that would be otherwise executed in a specialty engine if available.

By subtracting the projected CPU from the total CPU used, you can model the new general-purpose CPU utilization if zIIPs with enough capacity were available in the LPAR. This information can be used to estimate a new MSU 4-hour rolling average (Figure 10). zIIP CPU time is not accounted in the MSU 4-hour rolling average.

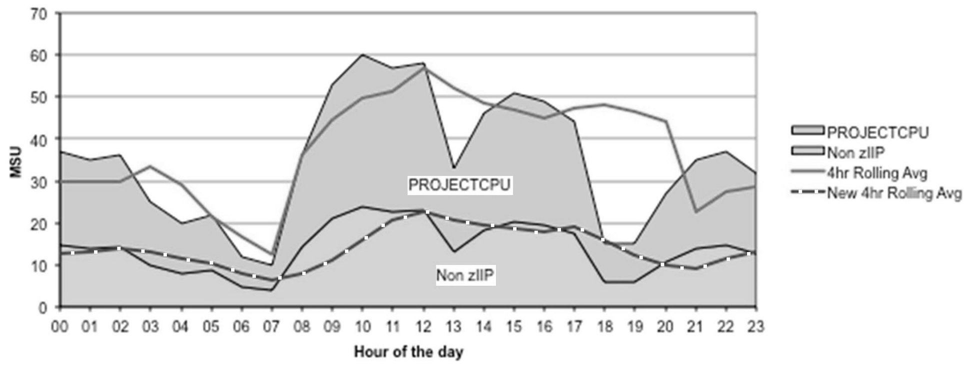


Figure 10: Projected CPU utilization with zIIPs available in the LPAR

The potential to reduce TCO by lowering CPU utilization-related costs is clear in this example. A DB2 data warehousing partition with distributed access to DB2 would be able to offload up to 60 percent of the DB2 CPU to zIIP processors.

Special Considerations for High zIIP Utilization

IMPORTANT: zIIP engines can help improve your mainframe infrastructure and reduce TCO, but they cannot be always considered exactly like general-purpose processors.

Configurations where the number of zIIPs is low compared with the number of standard processors are not uncommon. This imbalance is not necessarily a problem, provided the available number of zIIPs is enough to handle the eligible workload. Nevertheless, given the trend of ever-increasing eligible workloads, the stress on zIIP engines tends to grow over time.

z/OS dispatches eligible workload to zIIPs for execution. In cases where the zIIP is too busy, the zIIP can call for help from standard processors to complete a piece of work. When the system parameter IIPHONORPRIORITY is set to YES, a standard processor can run zIIP-eligible work if called by a zIIP processor. This portion of CPU is often referred to as specialty engine eligible processor time executed on a standard processor. This time is reported on RMF records, for example, and can be used as a capacity planning metric because it indicates that the system may benefit from more zIIP capacity. When a zIIP has to ask for help from standard processors, the cost-effectiveness of the overall system suffers.

Although zIIPs can offload eligible workloads to standard processors, calling in this assistance can slow down the workload. The system parameter ZIIPAWMT specifies an Alternate Wait Management (AWM) value for zIIPs. By default (that is, when HIPERDISPATCH=YES, a recommended system parameter value for performance), this means that a transaction may have to wait up to 3.2 milliseconds before it receives help from standard processors.

Some users have reported significant performance degradation with DB2 10 for z/OS for processes that ran smoothly in previous versions of DB2. This degradation is a consequence of too little zIIP capacity as well as the delays incurred when the zIIPs require help from standard processors. This impact would be even more dramatic if IIPHONORPRIORITY were set to NO, which prevents the zIIPs from receiving assistance from standard processors.

The potential problems related to a low zIIP capacity can be exacerbated by DB2 10's capacity to offload more CPU to zIIP engines. There are a few workarounds to this situation, but the proper design decision is to add more zIIP capacity to the LPAR.

Experience shows that it is a best practice to keep zIIP utilization at approximately 50 percent, or less, to avoid processing delays. For DB2 SQL-intensive LPARs (such as a data warehouse environment), the best performance is obtained when the number of zIIPs matches the number of standard processors.

DB2 and zAAP on zIIP

The z Application Assist Processor (zAAP) specialty engines are similar to zIIPs, but they are dedicated to running specific Java® and XML workloads on z/OS. Version 1.11

of z/OS introduced a feature that lets users run zIIP- and zAAP-eligible workloads on installed zIIP processors. This feature is enabled through the use of the ZAAPZIIP keyword in the IEASYS*xx* member of SYS1.PARMLIB. When ZAAPZIIP is set to YES, zAAP-eligible work is able to run on an available zIIP processor when no zAAP processors are present.

zAAP on zIIP is an excellent opportunity for users without enough zAAP- or zIIP-eligible workload to justify the zIIP acquisition ROI.

DB2 for z/OS can leverage the TCO advantages of zAAP on zIIP. The IBM white papers *WebSphere Application Server for z/OS: The Value of Co-Location* (WP101476) and *The Value of Co-Location: Update* (WP101476-2) describe the benefits for a workload using the cross-memory DB2 JDBC Type 2 driver compared with the DB2 JDBC Type 4 driver. This study compares DB2 9 versus DB2 10 and the DB2 JDBC Type 2 driver versus the DB2 JDBC Type 4 driver performances. This document has summarized the effect of two areas of benefit:

- The CPU benefits associated with JDBC type 2, which uses cross-memory technology. This eliminates the CPU associated with the TCP stack and DB2 DDF.
- The benefits associated with DB2 z/OS V10 as compared with DB2 z/OS V9.1.

Table 1 summarizes the comparison of CPU seconds consumed during the testing.

	Type 2 DB2 V10	**Type 4 DB2 V9.1**	**T2/V10 benefit**	**% T2/V10 benefit**
General-purpose CPU	979.92	1111.68	131.76	11.85%
Specialty engine	1605.60	2038.68	433.08	21.24%
Total CPU time	2585.52	3150.36	564.84	17.93%

Table 1: Comparison of Type 2 and Type 4 driver CPU utilization during testing

The study assumes an environment where zAAP is available or where zAAP on zIIP is enabled. The overall conclusion is that DB2 10 and DB2 JDBC Type 2 are the best option, for the scenario described in these papers, because of the CPU and co-location benefits.

IMPORTANT: IBM zEnterprise EC12 is planned to be the last System z server to offer support for zAAP processors. IBM plans to continue support for running zAAP workloads on zIIP processors.

Reducing TCO with Better Performance

Improved DB2 performance can reduce total cost of ownership by enabling applications to deliver fast, consistent response times without the need for more CPU capacity. Good performance can yield benefits that are difficult to measure but can provide priceless competitive advantages. Examples are customer satisfaction and the capacity to absorb seasonal influences without service degradation or the need for additional resources.

After migration, DB2 10 for z/OS and DB2 11 for z/OS provide substantial CPU and performance benefits. Better performance can be obtained *out of the box*, by mere migration and REBIND, as well as by investing resources in exploiting each version's new and enhanced features (Figure 11). Some of the benefits that require application changes are hash access path, index include columns, and inline large objects.

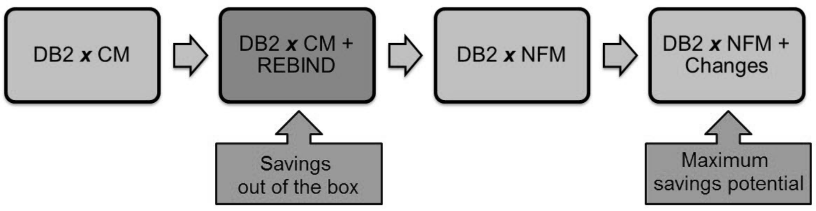


Figure 11: DB2 10 for z/OS "performance" migration path

Identifying Better Performance Opportunities

Existing applications can often be tuned to obtain better performance. When looking for performance opportunities, keep in mind the *Pareto principle*. This principle, also known as the 80–20 rule, states that roughly 80 percent of the effects come from 20 percent of the causes.

The Pareto principle usually applies well to DB2 performance. In DB2 terms, it is common to find that 80 percent of CPU utilization, locking problems, or scalability issues is produced by 20 percent of the applications. Identifying the 20 percent of applications that are of interest (from a performance point of view) lets us focus our tuning efforts. A highly effective way to define the performance focus is to look for the "top" consumers—for example, the top *x* CPU consumers, the top *y* most often executed applications, or the top *z* most active users.

The 80–20 rule usually applies to performance tuning efforts, as well. It is common to observe that 20 percent of the efforts produces 80 percent of the performance benefits. This fact is often used to look for "quick wins" or "sweet spots"—performance opportunities that have the potential to provide substantial benefits with low investment.

The *law of diminishing returns* also applies to DB2 performance tuning. This law states that a process will eventually yield to a point where additional efforts result in diminished benefits. In other words, over-tuning is not beneficial. Even though DB2 performance tuning and monitoring constitute a never-ending process, you have to stop tuning an application when the optimization efforts overcome the performance benefits.

It is essential to understand your workload profile before trying to calculate how total cost of ownership can be reduced by performance improvements. Using DB2 statistics and accounting information, in combination with RMF type 72 records, for example, it is possible to understand the resource utilization during a business period.

Figure 12 shows how CPU, measured in millions of service units, is used in a typical day in a financial institution.

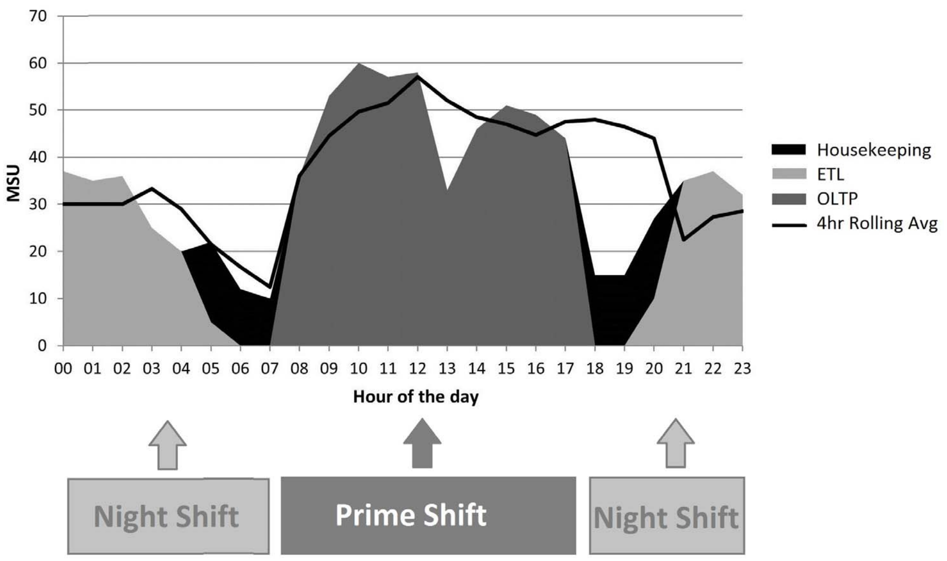


Figure 12: Daily CPU profile in a financial institution

The chart distinguishes the three main DB2 workloads: online transaction processing, housekeeping jobs, and extract, transform, and load (ETL) processes. Each one of these workloads requires specific considerations when looking for performance opportunities. Their contributions to the MSU 4-hour rolling average are also different.

Because of these inherent differences, it is a common, and good, practice to split the analysis into two or more periods. For example, the figure shows the workload split between night shift, which is characterized mostly by batch processes, and prime shift, which consists of mostly online transactions.

DB2 instrumentation is essential in this context. The DB2 accounting and statistics records are the basis of performance analysis. Consider starting these DB2 traces for continuous monitoring:

- Accounting classes 1, 2, 3, 7, and 8 with destination SMF
- Statistics classes 1, 3, 4, 5, and 6 with destination SMF

Specific performance monitoring or problem determination may require additional traces. Some performance traces may incur a performance impact and should be started only when needed.

A top-down DB2 accounting analysis approach is often effective in finding performance opportunities. Start by looking at the highest level, and then increase the level of detail. An accounting report by CONNECTION TYPE is a convenient starting point. Follow up by looking at more details for the relevant connection type—at plan or authorization ID level, for example.

In DB2 accounting, the Class 1 elapsed time shows the duration of the accounting interval. Also known as application time, it includes the time spent in DB2 and in the application.

The Class 2 elapsed time counts only the time spent in the DB2 address space during the accounting interval. It represents the sum of the times from any entry into DB2 until the corresponding exit from DB2. Class 2 elapsed time is also referred to as the time spent in DB2.

The Class 3 elapsed time counts wait time, including I/O, lock, and latch wait time. "Not accounted" time is the non-identified time spent in DB2.

Common DB2 performance bottlenecks are CPU, I/O (including synchronous I/O and logging), concurrency (locking), and storage (insufficient virtual storage and DBM1 constraint below the 2 GB bar).

As a starting point, calculate the "not in DB2 time" as follows:

Not in DB2 time = Class 1 – Class 2

The result will indicate whether further investigation should be done in DB2 or in the application logic, as illustrated in Figure 13.

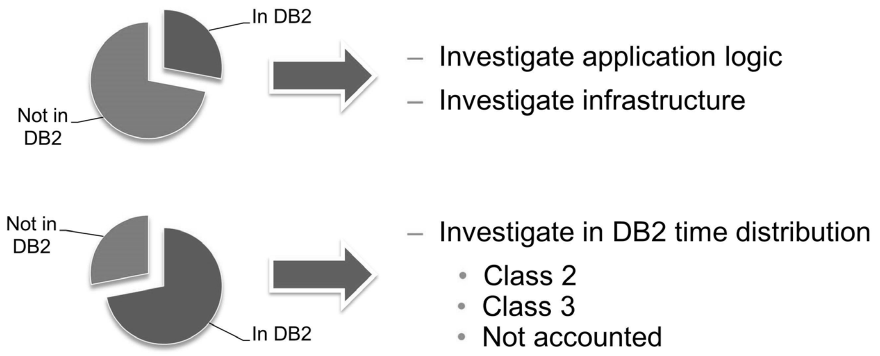

In DB2
Not in DB2
– Investigate application logic
– Investigate infrastructure
Not in DB2
In DB2
– Investigate in DB2 time distribution
• Class 2
• Class 3
• Not accounted

Figure 13: Investigating "not in DB2" versus "in DB2" time distribution

If most of the time is in DB2, you need to understand where the time is spent. This is a basic requirement for effective performance tuning. The time in DB2 is spent on using CPU, on waiting, or on not accounted time. The next steps in the investigation depend on the distribution of these three times.

Don't forget to verify the impact of external resources on the DB2 performance analysis. A z/OS LPAR is commonly shared by many applications, processes, and users. Effective DB2 tuning is possible only if the underlying system is well equipped to support the workload. A system issue, such as lack of CPU capacity or an incorrect Workload Manager definition, can have a significant impact on applications that can hardly be compensated for by proper tuning, good DB2 physical design, or application logic changes.

An impact analysis can help to put the performance opportunities in perspective and to focus on those with more potential for an organization. The example in Figure 14 shows a classification of opportunities by business impact versus implementation effort.

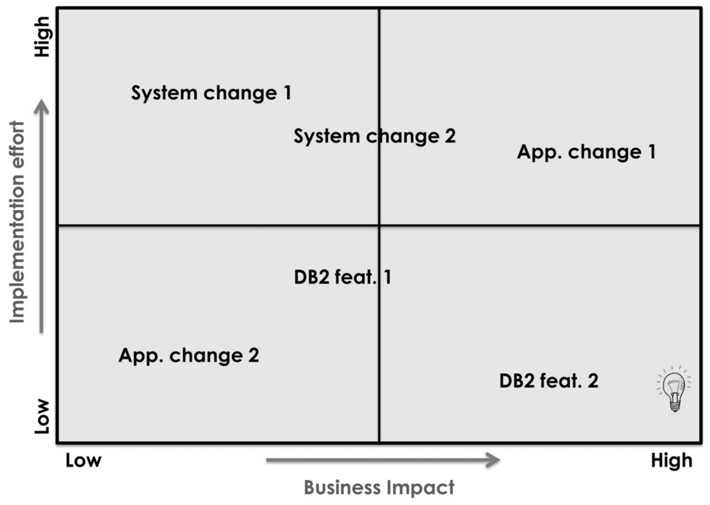


Figure 14: Impact analysis

Opportunities with low implementation effort and high business impact, also known as *quick wins* or *sweet spots*, have the potential to provide a high value in a short time with fewer efforts. Those with high implementation effort and high business impact are opportunities that can provide significant value but may not be not directly applicable.

Getting Better Performance with REBIND

The advantages of REBIND are many. Better performance is probably the most important. For example, a runtime overhead is involved in running a package bound in a previous version of DB2. Obtaining the CPU and memory advantages of a new DB2 release, as well as the benefits that may come with software maintenance, requires REBIND. You also need REBIND to gain the potential benefits of a new access path with updated DB2 statistics.

In addition, some system restrictions may impose the need for REBIND. For example, only packages bound after or in DB2 9 can be used with DB2 11.

Nevertheless, the advantages of and the need for REBIND must be balanced with the risks of access path degradation. Many organizations have adopted a conservative approach and avoid REBIND unless mandatory. Unfortunately, this policy implies a missed opportunity to get the most out of DB2 for z/OS.

DB2 has been adding features that help users to BIND and to REBIND in a safe way. For example, DB2 9's PLAN MANAGEMENT support (PLANMGMT subsystem parameter) helps users "fall back" to a previous instance of a package should an access path change cause a problem. DB2 10 enhances this support with the addition of new DB2 catalog tables and new BIND/REBIND options. DB2 11 continues the trend in this area with these enhancements:

- Further PLAN MANAGEMENT capabilities with the introduction of APREUSE(WARN)
- BIND and DDL break-in option on persistent DB2 threads with RELEASE(DEALLOCATE) packages
- The option to specify the package compatibility level behavior for static SQL (APPLCOMPAT)

DB2 Plan Management

PLAN MANAGEMENT, also referred to as access path stability, specifies whether DB2 saves historical information about the access paths for SQL statements, so you have the option to fall back to a previous version of the access path in case of performance degradation after REBIND.

PLAN MANAGEMENT imposes no performance impact at run time. It can be used when migrating to DB2 10. It provides safer mass REBIND strategies and can promote safer and easier mass REBIND campaigns when migrating to a new DB2 release.

At REBIND PACKAGE, DB2 for z/OS saves old PACKAGE copies in the DB2 SPT01 and catalog tables. Monitor SPT01 space utilization, especially with the EXTENDED option. DB2 10's APRETAINDUP(NO) REBIND option saves old copies only when they differ from the active copy.

The PLAN MANAGEMENT policy value can be:

- **OFF.** DB2 does not save access path information.
- **BASIC.** DB2 keeps information about the current access path and one additional access path, known as the previous access path. BASIC is the default in DB2 9.
- **EXTENDED.** DB2 saves information about the current and two additional access paths, known as the previous and original copies. EXTENDED is the default in DB2 10.

Figure 15 shows the BASIC policy in action. At REBIND, the CURRENT package copy becomes the PREVIOUS copy. The prior PREVIOUS copy is lost, and the newly created package becomes the CURRENT copy.

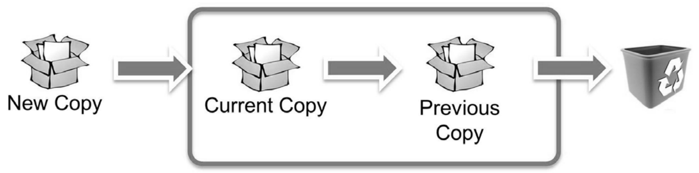


Figure 15: PLANMGMT(BASIC) policy

Figure 16 shows the EXTENDED policy in action. At REBIND, the CURRENT package copy becomes the PREVIOUS copy. The prior PREVIOUS copy is lost. If an ORIGINAL copy does not exist, the old CURRENT copy becomes the ORIGINAL copy as well; otherwise, the ORIGINAL copy is not altered. The newly created package becomes the CURRENT copy.

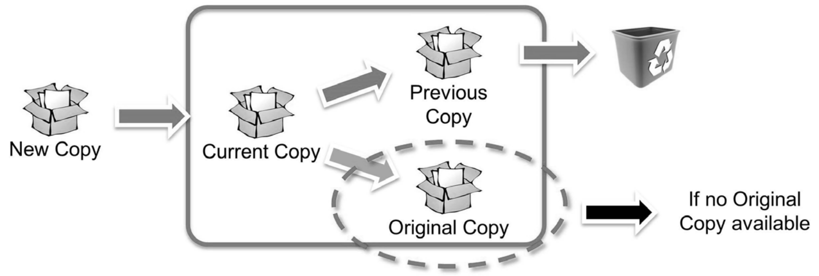


Figure 16: PLANMGMT(EXTENDED) policy

The following REBIND command shows the use of the PLANMGMT option EXTENDED:

```
REBIND PACKAGE(CRISCOLL.PCKGCR0)  -
  OWNER(PDB2)                    -
  QUALIFIER(PDB2)                -
  EXPLAIN(YES)                   -
  REOPT(NONE)                    -
  PLANMGMT(EXTENDED)
```

To revert to a previous package copy, you use the REBIND option SWITCH. Possible SWITCH values are:

- **SWITCH(PREVIOUS).** DB2 toggles the CURRENT and PREVIOUS packages. The CURRENT copy takes the place of the PREVIOUS copy, and the existing PREVIOUS copy takes the place of the CURRENT copy. Any existing ORIGINAL copy remains unchanged.
- **SWITCH(ORIGINAL).** DB2 replaces the CURRENT copy with the ORIGINAL copy; the CURRENT copy replaces the PREVIOUS copy. The existing PREVIOUS copy is lost. The existing ORIGINAL copy remains unchanged.

Figure 17 illustrates these two options.

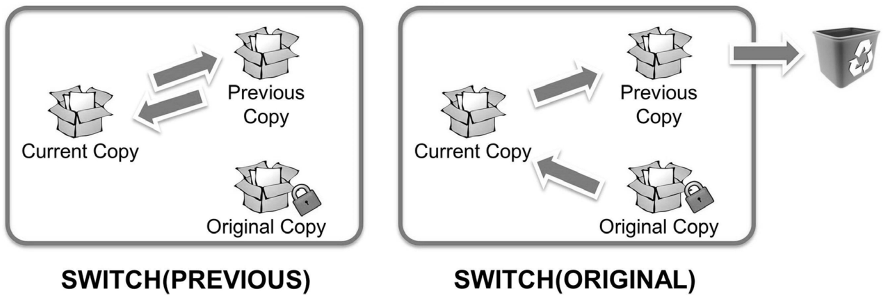


Figure 17: SWITCH(PREVIOUS) vs. SWITCH(ORIGINAL)

This example shows the use of SWITCH(ORIGINAL) in a REBIND command:

```
REBIND PACKAGE(CRISCOLL.PCKGCR0)  -
  SWITCH(ORIGINAL)
```

In DB2 10, the DB2 catalog table SYSIBM.SYSPACKCOPY contains a row for every saved package. The following SQL can be used to investigate SYSPACKAGE-like information for previous and original package copies.

```
SELECT
SUBSTR(COLLID,1,10) AS COLLID ,SUBSTR(NAME,1,10) AS NAME ,LASTUSED
 ,VALID ,OPERATIVE ,COPYID ,PLANMGMT
FROM SYSIBM.SYSPACKAGE
WHERE COLLID = 'CRISCOLL' AND NAME = 'PCKGCR0'
UNION ALL
SELECT
SUBSTR(COLLID,1,10) AS COLLID ,SUBSTR(NAME,1,10) AS NAME ,LASTUSED
 ,VALID ,OPERATIVE ,COPYID ,PLANMGMT
FROM SYSIBM.SYSPACKCOPY
WHERE COLLID = 'CRISCOLL' AND NAME = 'PCKGCR0'
WITH UR;

---------+---------+--------+---------+---------+-------+---------+-------
COLLID      NAME       LASTUSED    VALID  OPERATIVE     COPYID  PLANMGMT
---------+---------+--------+---------+---------+-------+---------+-------
CRISCOLL    PCKGCR0    2012-12-28  Y      Y                  0  E
CRISCOLL    PCKGCR0    2012-07-27  Y      Y                  1  E
CRISCOLL    PCKGCR0    2012-03-01  Y      Y                  2  E
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
```

The COPYID value indicates which package copy was used:

- 1 = PREVIOUS copy
- 2 = ORIGINAL copy
- 0 = CURRENT copy

DB2 11 APREUSE(WARN) Enhancement

DB2 10 introduced the BIND/REBIND option APREUSE. This option tells DB2 to try to reuse a previous access path for an SQL statement during BIND or REBIND. DB2 10 allows two different values:

- **APREUSE(NONE).** DB2 does not try to reuse previous access paths for statements in the package. This value is the default.
- **APREUSE(ERROR).** DB2 tries to reuse the previous access paths for SQL statements. If statements in the package cannot reuse the previous access path, REBIND fails, and no new package is created. Processing continues for the next package.

This feature is similar to having DB2 automatically use optimization hints. As such, there is no guaranty of success for every case. For example, if an index that is needed to apply a previous access path is missing, access path reuse will not work.

APREUSE provides an effective way to have packages gain the benefits of the DB2 10 run-time and real storage advantages while keeping a well-known access path. APREUSE exploits the information stored in the Explain Data Block (EDB) that was introduced in DB2 9. Only packages bound in DB2 9 or later can take advantage of APREUSE.

APREUSE(ERROR) is not always the best solution, and users have wanted an intermediate option between APREUSE(NONE) and APREUSE(ERROR)—one that would result in a new package even if the previous access path could not be used, and that would provide diagnostic information about access path reuse failures.

In DB2 10, APREUSE hints are applied at the query block level. If hints could not be applied for all the query blocks in a given SQL statement, the resulting access path could be less than optimal or even show access path regression. DB2 11 changes the granularity at which hints are applied. If DB2 cannot apply a hint for a statement, the entire hint is discarded at the statement level, and DB2 reoptimizes the statement in its entirety. This change lets DB2 11 extend the APREUSE functionality by adding the option WARN.

With APREUSE(WARN), DB2 tries to reuse the previous access paths for SQL statements in the package, but REBIND is not prevented when they cannot be reused. Instead, DB2 generates a new access path for the SQL statement and ignores the previous access path. While APREUSE(ERROR) operates on a package boundary, APREUSE(WARN) can be seen as operating on a statement boundary.

With DB2 11, users can mass REBIND a DB2 environment and be notified about those packages on which access path reuse was not possible. As with HINTS, it is to be expected that a small percentage of the packages will not apply. With APREUSE(WARN), you get a new package and the notification that reuse was not possible. Not being able to reuse an access path is not necessarily an issue, because the new access path could actually be better than the previous one. An exploration of the PLAN_TABLE information can help you decide whether to let the package run with the new access path or to switch back to a previous package. The latter possibility requires the use of the PLANMGMT bind option.

DB2 11 improves the diagnostic information made available for cases in which reuse cannot be applied. When you use APREUSE in combination with EXPLAIN(YES) or EXPLAIN(ONLY), DB2 populates the PLAN_TABLE with access path information. In DB2 11, the PLAN_TABLE accurately describes the valid new access path even in case of reuse failure, regardless of whether APREUSE(ERROR) or APREUSE(WARN) is specified.

DB2 11 RELEASE(DEALLOCATE) Optimization

The BIND option RELEASE(DEALLOCATE) can reduce CPU for some applications, but there are some concerns related to its utilization. It is therefore a performance best practice to apply RELEASE(DEALLOCATE) selectively and to measure the impacts on application performance and concurrency. For example, REBIND and DDL operations may experience timeout with persistent threads using RELEASE(DEALLOCATE), especially when applied to local connections such as CICS or IMS transactions. DB2 11 alleviates this concern by supporting BIND and DDL operations to break in persistent threads.

Before DB2 11, for applications running with RELEASE(DEALLOCATE), the accumulation of objects referenced in storage could impact thread footprint and CPU usage as the number of objects per thread increased. DB2 11 provides optimizations in the RELEASE(DEALLOCATE) processing and automatically tracks resource and lock usage. To provide consistently good performance, DB2 11 frees resources and locks held by packages that touch a large number of objects once internal thresholds are reached. This change relaxes the concerns around the use of RELEASE(DEALLOCATE).

DB2 11 Application Compatibility and APPLCOMPAT

Some applications might behave differently or receive an error when they try to use DB2 11 functions and features. For this reason, DB2 11 provides the ability to run individual applications as in DB2 10 once you migrate to DB2 11 NFM. The BIND/REBIND option APPLCOMPAT specifies the package compatibility level behavior for static SQL. For dynamic SQL, the behavior is determined by the APPLICATION COMPATIBILITY special register. The compatibility level can be:

- V10R1. Applications run with the features and behavior of DB2 10.
- V11R1. Applications run with the features and behavior of DB2 11.

DB2 11 application compatibility gives users a safer migration path. It relaxes the need to review every application's functionality before migrating to DB2 11 and helps guarantee that no behavior changes will happen with the new DB2. While this option isolates applications from release incompatibilities, it also prevents them from getting some of the SQL advantages of DB2 11.

Application compatibility can be applied at the package level. When a potential application or SQL incompatibility is fixed, you can change the APPLCOMPAT value to V11R1 to let an individual application be able to exploit the new options of DB2 11.

Case Study: Performance Benefits of REBIND

REBIND can provide performance benefits in many ways, but organizations often do not REBIND as a way to protect applications from possible access path degradation. The risks of degraded performance can be minimized by exploiting PLAN MANAGEMENT. Working with PLAN MANAGEMENT provides a fallback option to a previous access path in case of issues. There is no runtime impact for packages bound with PLAN MANAGEMENT.

A financial institution avoiding REBIND decides to exploit PLAN MANAGEMENT to explore the potential benefits of a mass REBIND campaign. As a first exercise, the user decides to REBIND all packages used in the OLTP workload window.

In DB2 9, the following query provides the distribution of the packages by RELBOUND, the release when the package was bound or rebound.

```
SELECT
RELBOUND, COUNT(*)
FROM  SYSIBM.SYSPACKAGE
WHERE VALID <> 'N'
  AND OPERATIVE <> 'N'
GROUP BY RELBOUND
WITH UR;
```

DB2 10 introduced the LASTUSED column in SYSIBM.SYSPACKAGE to track the date when a package was last used. This information can be extremely handy to safely free unused packages or to decide that a REBIND of a package may be useful. The following example uses the LASTUSED column in a query.

```
SELECT
RELBOUND, COUNT(*)
FROM  SYSIBM.SYSPACKAGE
WHERE VALID <> 'N'
  AND OPERATIVE <> 'N'
  WHERE LASTUSED >=
  (CURRENT_DATE - 1 MONTH)
GROUP BY RELBOUND
WITH UR;
```

The RELBOUND distribution for this company revealed that although the organization is running DB2 10, only 35 percent of the packages were bound in this release. Almost 49 percent of packages were last bound in DB2 9, and 16 percent in DB2 8.

The organization decided to mass REBIND all the packages using PLANMGMT(EXTENDED), executing a mass REORG and RUNSTATS campaign in preparation for the changes. Figure 18 shows the performance results, comparing CPU utilization and number of SQL transactions per hour before and after REBIND.

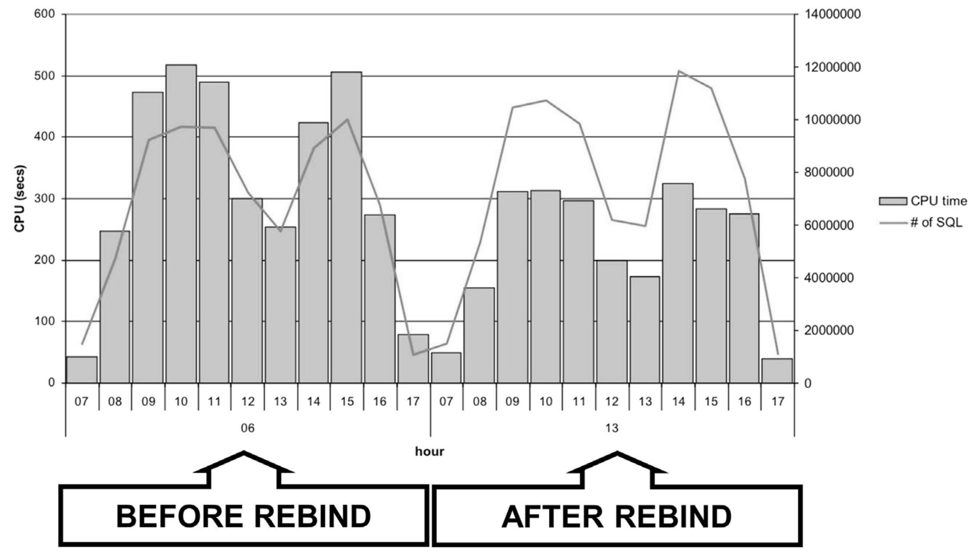


Figure 18: Effect of mass REBIND on total DB2 CPU time

As a result of the mass REBIND of more than 2,000 DB2 packages, the CPU utilization is visibly reduced and the transaction throughput is higher.

Two frequently used packages went wrong. The new packages included a multiple-index access path that had row ID (RID) list failures. This performance issue was quickly fixed by switching back to the previous packages. Further investigation ended up adding the REOPT option to REBIND for these packages.

DB2 EXPLAIN At a Glance

DB2 EXPLAIN provides valuable information to help you understand the optimizer choices and tune application performance. Among the many available EXPLAIN options are:

- **BIND/REBIND with EXPLAIN(YES).** Generates a new access path, populates PLAN_TABLE, and creates a new package
- **BIND/REBIND with EXPLAIN(ONLY).** Generates a new access path, populates PLAN_TABLE, but *does not* create a new package
- **EXPLAIN PLAN (i.e., SPUFI).** Generates a new access path and populates PLAN_TABLE
- **EXPLAIN PACKAGE.** *Does not* generate a new access path; extracts the existing access path from the package and populates PLAN_TABLE

DB2 10 High-Performance DBATs

DB2 10 provides valuable performance improvements for distributed applications, including improved return to client result sets, enhanced support for native SQL PL procedures, extended correlation token, virtual and real storage improvements, and LOBs and XML materialization avoidance.

One of the most relevant changes is the introduction of high-performance database access threads (DBATs). DB2 10 high-performance DBAT support reduces CPU consumption by supporting the BIND option RELEASE(DEALLOCATE) for distributed access to DB2. These enhancements save CPU by avoiding repeated package allocation/ deallocation and avoiding acquiring and releasing parent (IS, IX) locks frequently.

The more noticeable CPU reduction is expected for short transactions, typically OLTP workloads. No benefit is achieved for ACTIVE threads (CMSTATS=ACTIVE), nor for applications running with the REBIND option KEEPDYNAMIC YES.

Table 2 summarizes the observed performance results during controlled OLTP benchmark tests. It compares DB2 9 with DB2 10 results out of the box, then shows further savings achieved when running DB2 10 with RELEASE(DEALLOCATE). The times shown are the total CPU time (in microseconds) per transaction, that is:

Total CPU = DB2 System Services Address Space + Database Services Address Space + IRLM + DDF Address Space CPU

Workload	**DB2 9 CPU (ms)**	**DB2 10 CPU (ms)**	**Savings**	**DB2 10 DEALLOCATE CPU (ms)**	**Savings**
SQL ODBC/CLI: Dynamic SQL	2114	1997	5.5%	1918	9.3%
JDBC: Dynamic SQL	2152	2017	6.3%	1855	13.8%
SQLJ: Static SQL	1899	1761	11.9%	1668	16.6%
Stored procedures in SQLJ with static SQL	1768	1642	6.7%	1550	11.9%

Table 2: OLTP benchmark test results

DB2 provides the –MODIFY DDF PKGREL command to activate or deactivate the distributed RELEASE(DEALLOCATE) option support. The –MODIFY DDF command's PKGREL option specifies whether DB2 ignores the BIND options of packages used for remote client processing. PKGREL can be modified by two options:

- **BNDOPT.** The rules of the RELEASE bind option that were specified when the package was bound are applied. This is the default option in DB2 10.
- **COMMIT.** The rules of the RELEASE(COMMIT) bind option are applied. This is the DB2 9 (and earlier) behavior.

This example shows how to enable the honoring of the RELEASE option for distributed packages:

```
-MODIFY DDF PKGREL(BNDOPT)
```

After executing this command, DB2 outputs the following feedback in the MSTR address space.

```
STC12396  DSNL300I  -DB0A DSNLTMDF MODIFY DDF REPORT FOLLOWS:
          DSNL302I PKGREL IS SET TO BNDOPT
          DSNL301I DSNLTMDF MODIFY DDF REPORT COMPLETE
```

To disable, use the following syntax; this overlays the distributed package's RELEASE option and causes DB2 to apply the RELEASE(COMMIT) behavior.

```
-MODIFY DDF PKGREL(COMMIT)
```

The –DIS DDF DETAIL command contains the DSNL106I message, which reports the current status of the PKGREL option. The following –DIS DDF DETAIL command output shows that the current status for PKGREL is COMMIT in this system.

```
DSNL080I  -DBCM DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION           LUNAME            GENERICLU
DSNL083I DBCM               OVERIJS.TOTDBCM   -NONE
DSNL084I TCPPORT=12345 SECPORT=12346 RESPORT=12347 IPNAME=-NONE
DSNL085I IPADDR=::10.50.1.1
DSNL086I SQL    DOMAIN=www.molaro.be
DSNL090I DT=I  CONDBAT=  10000 MDBAT=  200
DSNL092I ADBAT=    0 QUEDBAT=      0 INADBAT=      0 CONQUED=      0
DSNL093I DSCDBAT=      0 INACONN=      1
DSNL105I CURRENT DDF OPTIONS ARE:
DSNL106I PKGREL = COMMIT
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Running distributed applications with RELEASE(DEALLOCATE) improves performance in many cases. However, it could become difficult to execute some operations, such as DDL, on objects allocated by the workload.

IMPORTANT: The BIND/REBIND option RELEASE(DEALLOCATE) may create concurrency problems. Plan for a gradual implementation and monitor impacts.

Reducing TCO Through Storage Savings

The worldwide growth of information has been increasing steadily for many years. In 2007, the amount of information created, captured, or replicated exceeded available storage for the first time. With big data, this scenario is certainly not going to revert.

As an illustration, consider the storage requirements to support the insert of a simple 9-byte DB2 table row as illustrated in Figure 19.

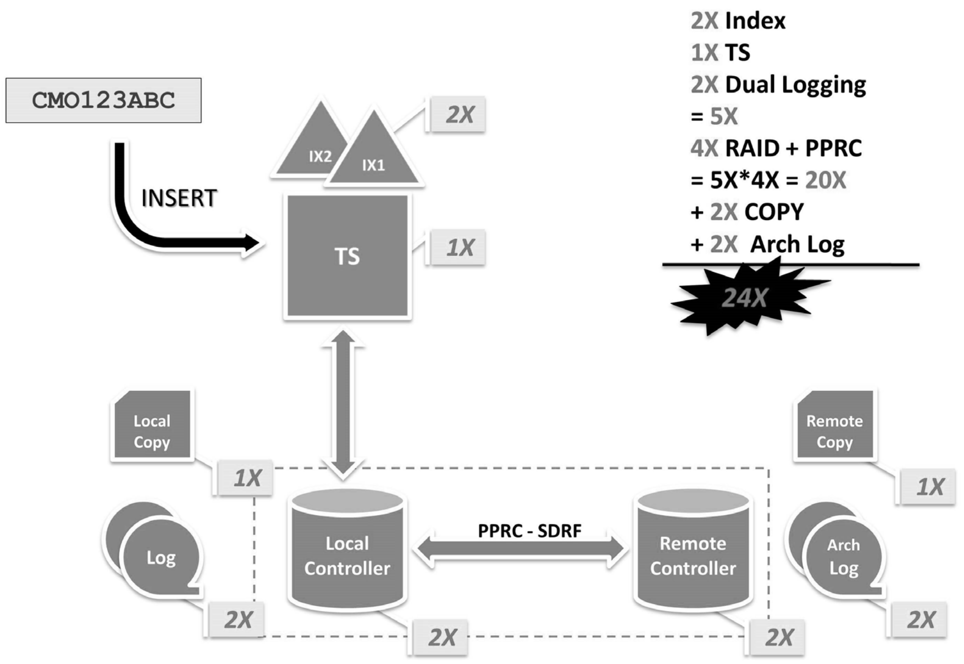


Figure 19: Storage required for a simple table insert

The data has to be saved once in the table space containing the table. Assuming that the row's columns are part of two indexes, the data has to be stored two times more. The usual dual-logging system then requires storing the data two times more. At this point, the original data has to be written five times already. Normally, disk storage is replicated to a remote data center using replication techniques such as peer-to-peer remote copy (PPRC). Local storage is mirrored with some form of redundant array of independent disks (RAID). As shown in this example, a commonly found configuration, the inserted data is stored 24 times.

Several DB2 catalog tables can be used to analyze storage space utilization. For example, you can create reports about the number of kilobytes of DASD storage allocated using the SPACE and SPACEF columns of table SYSIBM.SYSINDEXES, SYSIBM.SYSTABLESPACE, SYSIBM.SYSINDEXPART, SYSIBM.SYSTABLEPART, and SYSIBM.SYSSTOGROUP.

The STOSPACE DB2 utility refreshes these columns. It determines the amount of space allocated for DB2 storage groups and their related table spaces and indexes. STOSPACE does not set a utility restrictive state on target objects, and it can run concurrently with any utility. It is also cheaper and faster to execute than RUNSTATS, so it is a better option if you are looking only for space analysis information.

This example shows how to execute the STOSPACE utility:

```
//JOBCARD
//* ------------------------
//STOSPACE EXEC PGM=DSNUTILB…
...
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
   STOSPACE STOGROUP *
/*
```

The DB2 realtime statistics tables are also a good information source. For example:

- **SYSIBM.SYSINDEXSPACESTATS.** The LASTUSED column contains the date when the index was last used for SELECT, FETCH, searched UPDATE, searched DELETE, or to enforce referential integrity constraints.
- **SYSIBM.SYSTABLESPACESTATS.** The SPACE column shows the amount of space, in kilobytes, allocated to the table space or partition; for multi-piece, linear page sets, this value is the amount of space in all data sets. The DATASIZE column provides the total number of bytes that row data occupy in the data rows or LOB rows. The UNCOMPRESSEDDATASIZE column shows the total number of bytes that a data row would have occupied if the data were not compressed.

Even if the storage cost per unit of data is decreasing, ever-increasing space demands make storage a substantial investment. The opportunities to reduce total cost of ownership related to storage can be classified in two main categories: optimizing storage utilization and dividing the data.

Optimizing storage aims to reduce the storage requirements for valuable business data. DB2 for z/OS provides several techniques to achieve this objective, including data compression, DB2 managed disk space allocation, and index compression. Rationalization of object-related storage can provide big savings as well, by removing redundant objects and reclaiming unused space, for example.

Dividing the data means exploiting DB2 features such as partitioning or creating history tables to reduce the amount of data that has to be processed by applications and housekeeping processes. The smaller the data scope, the faster and less expensive the processing.

DB2 Data Compression

In many cases, compressing the data in a table space significantly reduces the amount of disk space needed to store data. This effect can reduce TCO by lowering space requirements. Data compression can provide performance benefits, as well.

DB2 data compression and decompression are hardware-assisted. This approach helps to reduce CPU costs related to accessing and manipulating compressed data. In general, compression and decompression get faster with each new generation of System z servers, as processors get faster.

DB2 compression is a lossless compression implementation. That is, no details from the data are lost when compressing/decompressing. On the other hand, lossy compression techniques, such as that used to create JPEG images, involves losing some information each time you compress the data. With DB2 for z/OS, data is compressed in the DB2 logs, in the buffer pools, and in the data pages in both table spaces and copies.

Data compression is dictionary-based. Before getting compression results, you must create a dictionary using the LOAD or REORG utility. A missing or inadequate dictionary results in a less effective compression ratio.

Figure 20 depicts the internal changes in a table space after compression. Note the addition of the dictionary in the AFTER section of the figure.

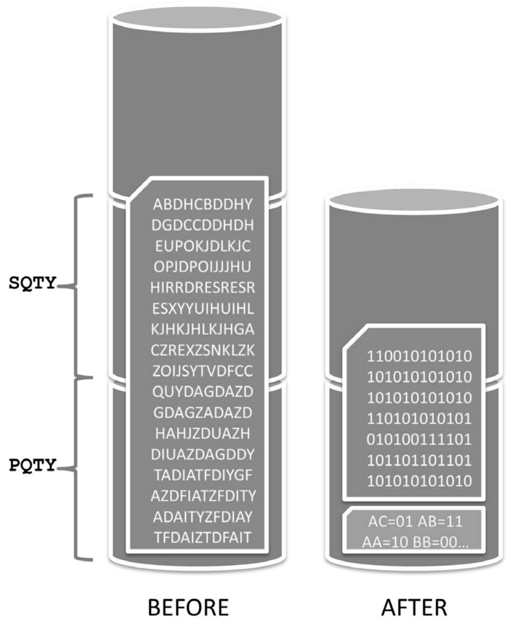


Figure 20: Table space changes following compression

One dictionary is maintained per compressed table space partition. Since DB2 8, the dictionary is loaded in DB2 storage above the 2 GB bar. Nevertheless, with up to 4,096 partitions per table and sixteen 4 KB pages per dictionary, memory requirements can increase quickly when monitoring real storage utilization and system paging.

The compression ratio depends on the data characteristics. Compression can work well for large table spaces. But with small ones, compression may result in more space utilization due to the introduction of the compression dictionary. You can use the DB2-provided DSN1COMP standalone utility to estimate compression effects before compressing a table space. The following example shows the output of an execution of DSN1COMP; in this case, 25 percent of the bytes will be saved.

```
DSN1940I DSN1COMP COMPRESSION REPORT
             301  KB WITHOUT COMPRESSION
             224  KB WITH COMPRESSION
              25  PERCENT OF THE BYTES WOULD BE SAVED

           1,975  ROWS SCANNED TO BUILD DICTIONARY
           4,665  ROWS SCANNED TO PROVIDE COMPRESSION ESTIMATE
           4,096  DICTIONARY ENTRIES

              81  BYTES FOR AVERAGE UNCOMPRESSED ROW LENGTH
              52  BYTES FOR AVERAGE COMPRESSED ROW LENGTH

              16  DICTIONARY PAGES REQUIRED
             110  PAGES REQUIRED WITHOUT COMPRESSION
              99  PAGES REQUIRED WITH COMPRESSION
              10  PERCENT OF THE DB2 DATA PAGES WOULD BE SAVED
```

You can query the DB2 catalog to obtain information about the actual effectiveness of compression. For example, column PAGESAVE of SYSIBM.SYSTABLEPART provides the percentage of pages that are saved by compressing the data.

Figure 21 shows the PAGESAVE distribution for the table spaces of a DB2 for z/OS data warehousing environment. To build this chart, only COMPRESS=Y table spaces were considered. In general, this kind of environment often shows compression ratios of 50 percent or more, which can safely be considered as compression working okay for the data in these table spaces. A compression ratio below 50 percent may require attention; compression may not be effective for that data, or the dictionary may need to be re-created. PAGESAVE < 1 requires immediate review; this may indicate either that compression results in more space requirements or, more commonly, that a dictionary is missing.

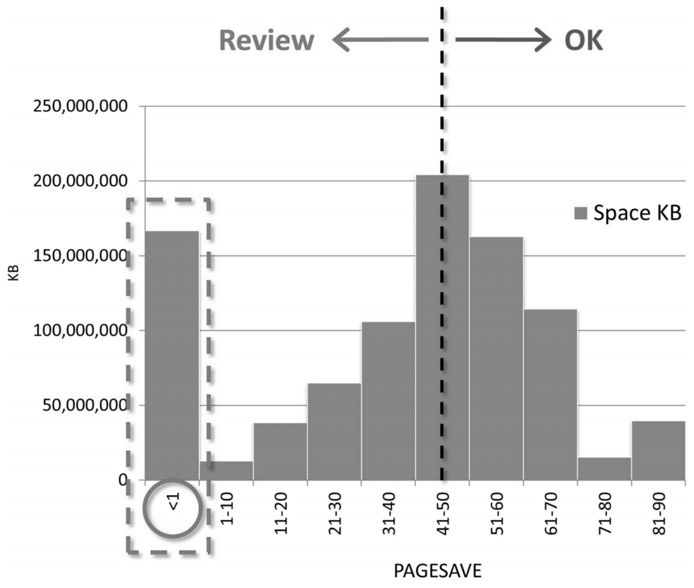


Figure 21: PAGESAVE distribution for the table spaces of a data warehousing environment

On top of space savings, DB2 data compression can provide other kinds of performance advantages. Because data is compressed in buffer pools, compression may enable storing more information within the same buffer pool size. This can make the buffer pool scanning of large amounts of data more effective or increase the buffer pool hit ratio. Compression can reduce DB2 logging and speed up taking copies.

In many cases, compressing the data in a table space significantly reduces the amount of disk space needed to store data. Nevertheless, data compression may not be the best choice for every object. Because DB2 compresses the data one record at a time, compression may be not efficient with short rows because 8 bytes of overhead are required to store each record in a data page. Even if rows compress well, there is a limit of a maximum 255 rows per page. This may result in data pages with lots of empty space, and you may end up with compression overhead but no space savings.

DB2 Managed Disk Space Allocation

Traditionally, a database administrator designs the storage characteristics of a table space and defines the primary and secondary quantities in the DDL USING block, as shown in this example:

```
>>-USING-------------------------------------------------------->

                        (1)
>--+-VCAT--catalog-name-----------------------------------+----><
   |                          .---------------------.     |
   |                          V                     | (2) |
   '-STOGROUP--stogroup-name----+-----------------+-+-----'
                                +-PRIQTY--integer-+
                                +-SECQTY--integer-+
                                | .-ERASE NO--.   |
                                '-+-----------+---'
                                  '-ERASE YES-'
```

PRIQTY specifies the minimum primary space allocation, and SECQTY specifies the minimum secondary space allocation for a DB2 managed data set. This approach has been used successfully, but it requires administration and monitoring efforts. In addition, big SECQTY values may result in large quantities of unused allocated storage space.

With DB2 managed disk space allocation, DB2 can calculate the amount of space to allocate to secondary extents by using a sliding-scale algorithm. The first 127 extents are allocated in increasing size, and the remaining extents are allocated based on the initial size of the data set. If the environment is not suffering from severe DASD fragmentation, a table space should reach its dataset space limit before reaching its maximum number of extents.

Sliding scale for secondary space allocation is an optional DB2 feature. It can help to reduce TCO by simplifying the administrative tasks related to the monitoring of DB2 table space and index space extents. Because the size of the secondary extents follows the growing of the data, it can provide storage savings by keeping unused space to a minimum.

Figure 22 shows the initial space utilization of, and the effects of sliding on, an enterprise development environment. Objects used to be created using large space allocations in preparation for receiving data. With sliding allocation, unused space is reduced to a minimum.

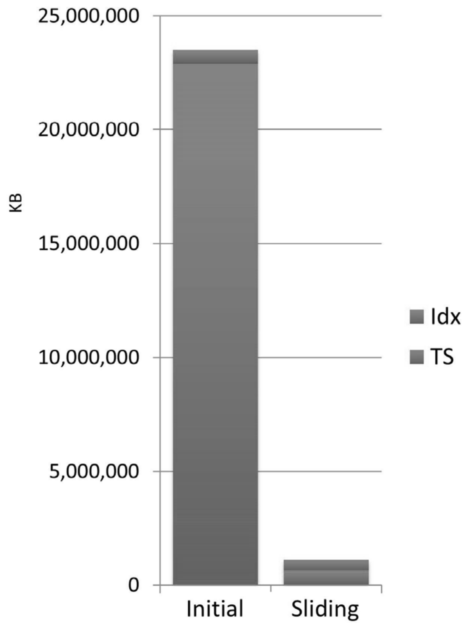


Figure 22: Effects of sliding-scale allocation of secondary space

Sliding allocation can help to save significant quantities of storage by optimizing the data pages allocated. There is no effect on data. Figure 23 shows that sliding does not change the amount of data in the object. Savings come from DB2's smarter and automatic extent management.

Case Study: Combined Effects of Data Compression and DB2 Managed Disk Space Allocation

DB2 data compression and DB2 managed disk space allocation can be combined to obtain a maximum storage space savings. This case study shows an example.

The starting point is a DB2 table holding about 10,000,000 rows. The table is supported by a partitioned table space with three partitions. Because of business needs, the data cannot be evenly distributed at the initial stage.

Figure 24 illustrates how to use the DB2 catalog tables to obtain space information about this object. SYSTABLES gives the total number of rows under CARD and the number of pages that include rows of the table under NPAGES. SYSTABLEPART shows the details per partition: the number of rows per partition under CARD, storage space utilization under SPACE, and number of data set extents under EXTENTS.

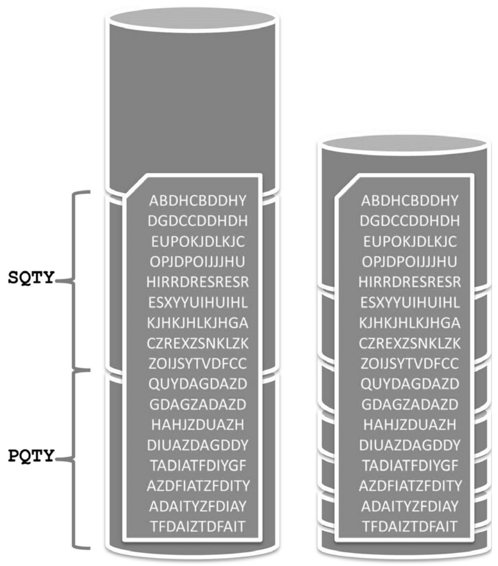
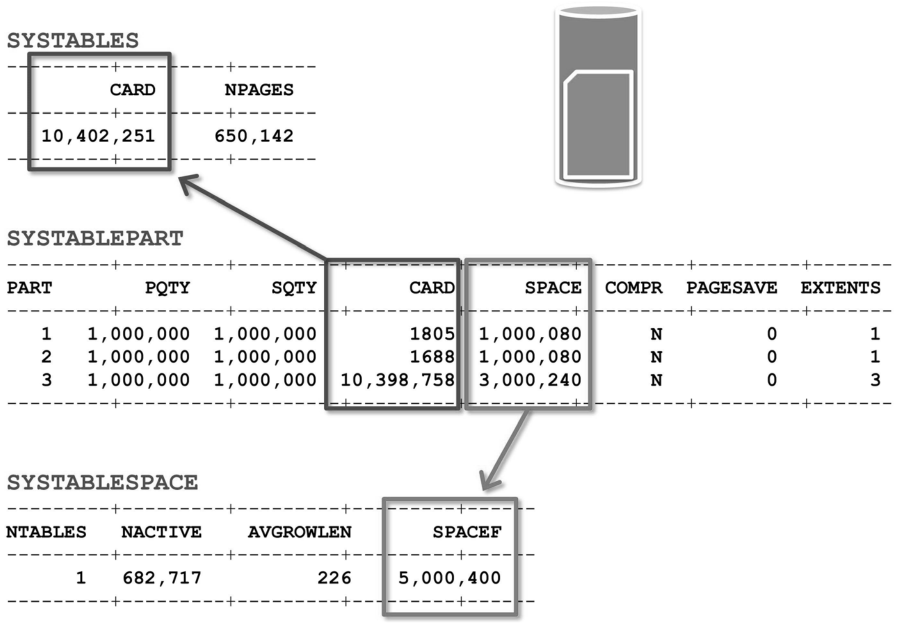


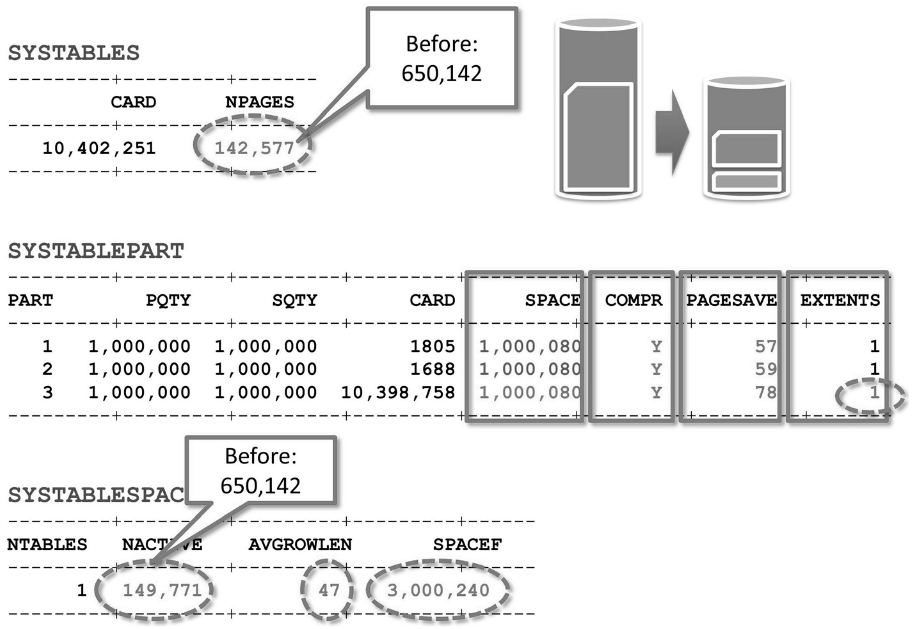
Figure 23: Object data unaffected by sliding-scale allocation



```
SYSTABLES
---------+---------+-------
      CARD      NPAGES
---------+---------+-------
  10,402,251     650,142
---------+---------+-------

SYSTABLEPART
---------+---------+---------+---------+---------+---------+---------+-------
PART        PQTY       SQTY         CARD       SPACE  COMPR  PAGESAVE  EXTENTS
---------+---------+---------+---------+---------+---------+---------+-------
   1   1,000,000  1,000,000         1805   1,000,080     N          0        1
   2   1,000,000  1,000,000         1688   1,000,080     N          0        1
   3   1,000,000  1,000,000   10,398,758   3,000,240     N          0        3
---------+---------+---------+---------+---------+---------+---------+-------

SYSTABLESPACE
---------+---------+---------+---------+----
NTABLES   NACTIVE    AVGROWLEN      SPACEF
---------+---------+---------+---------+----
      1   682,717          226   5,000,400
---------+---------+---------+---------+----
```

Figure 24: Using the DB2 catalog tables to obtain space information about an object

Figure 25 shows the effects of altering the table space to COMPRESS=Y and creating a compression dictionary with a REORG utility. The number of rows does not change, but the average row length goes from 226 to 47 bytes as a result of compression. The number of data pages containing rows drops from 650,142 to 142,577. And the kilobytes of DASD used by the table space is reduced from 5,000,400 to 3,000,240. In this example, compression reduced the space requirements by approximately 40 percent.

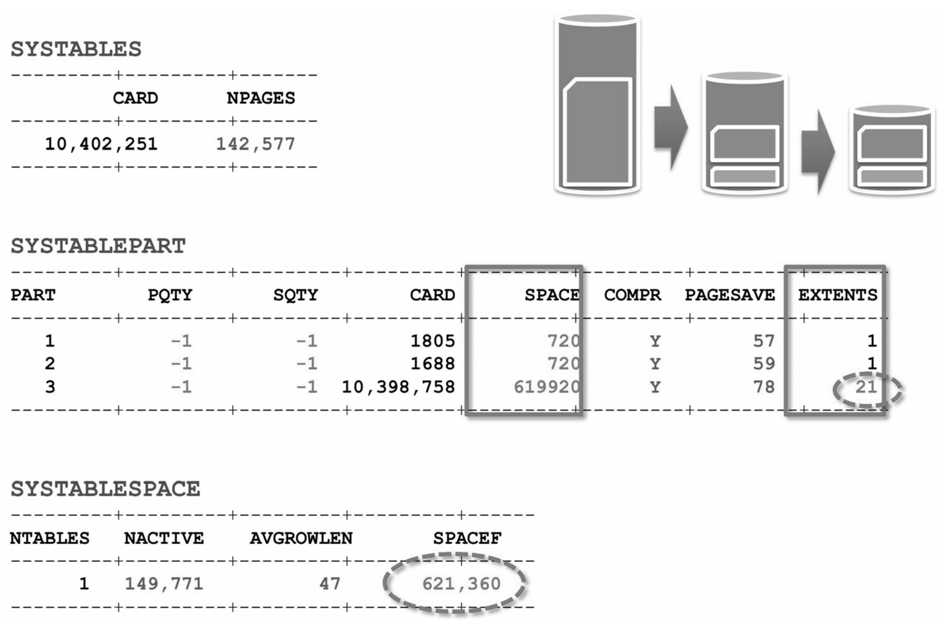


```
SYSTABLES
---------+---------+-------
        CARD     NPAGES
---------+---------+-------
  10,402,251    142,577
---------+---------+-------

SYSTABLEPART
---------+---------+---------+---------+---------+---------+---------+---------
PART        PQTY      SQTY        CARD      SPACE  COMPR  PAGESAVE  EXTENTS
---------+---------+---------+---------+---------+---------+---------+---------
   1   1,000,000 1,000,000        1805  1,000,080      Y        57        1
   2   1,000,000 1,000,000        1688  1,000,080      Y        59        1
   3   1,000,000 1,000,000  10,398,758  1,000,080      Y        78        1
---------+---------+---------+---------+---------+---------+---------+---------

SYSTABLESPAC
---------+---------+---------+---------+------
NTABLES   NACT  VE    AVGROWLEN       SPACEF
---------+---------+---------+---------+------
      1    149,771          47    3,000,240
---------+---------+---------+---------+------
```

Figure 25: Using the DB2 catalog tables to show space savings after compression

Because partitions 1 and 2 are over-allocated for the amount of data they contain, altering the table space to use a sliding scale for secondary space allocation can help further reduce storage needs. PQTY and SQTY are altered to –1, and a REORG utility is required to adjust the extents. Figure 26 shows the final results. There is no impact on the number of pages, but the DASD space is reduced to 621,360 KB.

Figure 27 gives a graphical summary of these changes. In this example, the combined effects of DB2 data compression and sliding scale for secondary space allocation reduced the storage requirements by almost 90 percent without affecting the data. In addition, applications accessing the data may benefit from some of the performance advantages of data compression. DB2 utilities such as REORG and COPY may execute faster as a result of the reduced table space DASD space.



```
SYSTABLES
---------+---------+-------
         CARD      NPAGES
---------+---------+-------
   10,402,251     142,577
---------+---------+-------

SYSTABLEPART
---------+---------+---------+---------+---------+---------+---------+-------
PART         PQTY       SQTY         CARD       SPACE  COMPR  PAGESAVE  EXTENTS
---------+---------+---------+---------+---------+---------+---------+-------
   1           -1         -1         1805         720      Y        57        1
   2           -1         -1         1688         720      Y        59        1
   3           -1         -1   10,398,758      619920      Y        78       21
---------+---------+---------+---------+---------+---------+---------+-------

SYSTABLESPACE
---------+---------+---------+---------+------
NTABLES   NACTIVE    AVGROWLEN       SPACEF
---------+---------+---------+---------+------
      1   149,771           47      621,360
---------+---------+---------+---------+------
```

Figure 26: Using the DB2 catalog tables to show the impact of sliding scale

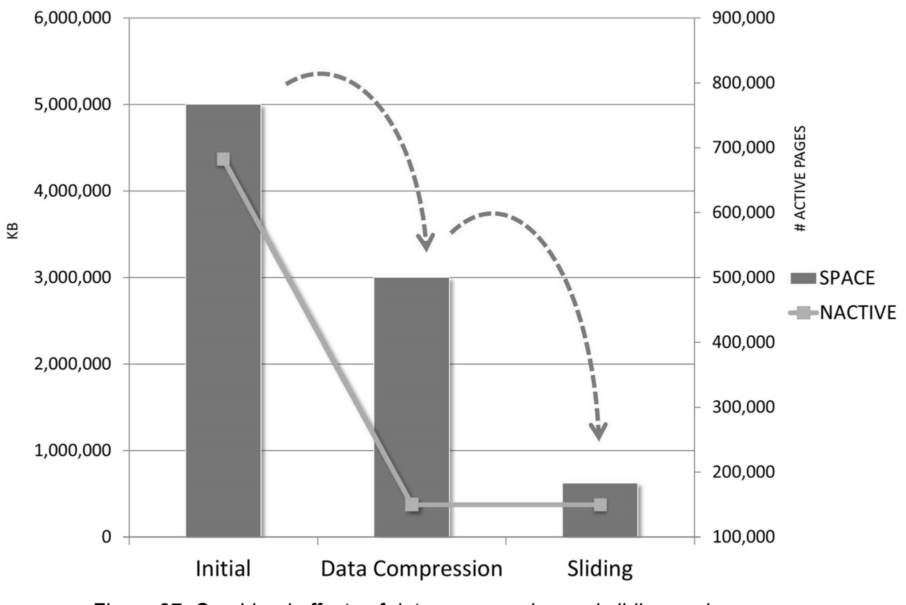


Figure 27: Combined effects of data compression and sliding scale on space

Index Compression

In DB2 data warehouse environments, it is common to take advantage of DB2 data compression to save DASD storage and reduce costs. To improve performance, extensive use of indexes is common. As a consequence of these two factors combined, it is usual to see environments where the disk storage space used by indexes is bigger than that for table spaces.

Figure 28 shows the space utilization, in kilobytes, of the table spaces and index spaces for the five biggest databases in manufacturing organization's DB2 data warehouse environment.

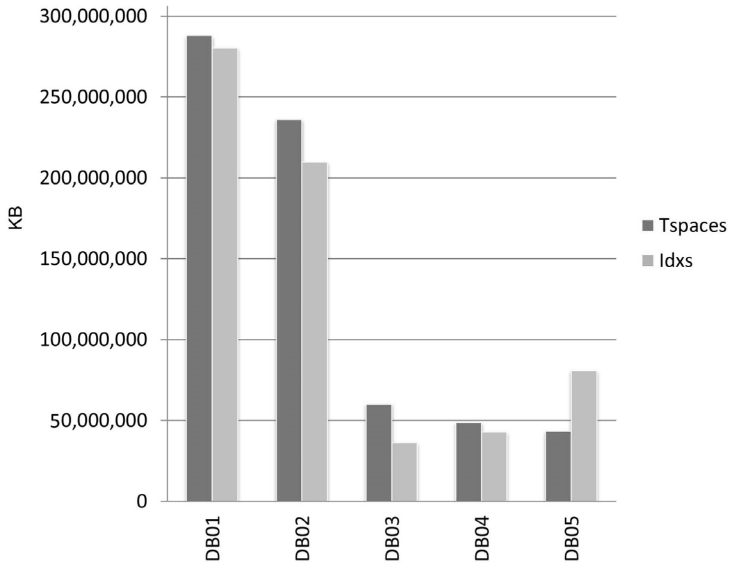


Figure 28: Space utilization in a DB2 data warehouse environment

In this case, DASD space used by indexes is almost as much as that used by table spaces, with indexes representing 48 percent of the total space requirements.

To further reduce storage requirements, DB2 introduced index compression in DB2 9. DB2 uses a hybrid compression algorithm that employs prefix compression to compress index pages. Index compression is not the same as data compression. The goal of index compression is to save index storage on disk. Compression of index data occurs on disk only and is not hardware-supported. With index compression, CPU overhead is incurred at read and write I/O execution time only. Table 3 outlines the differences between data and index compression.

	Data compression	Index compression
Technique	Ziv-Lempel hardware assisted	Prefix compression
Dictionary	Must be built before getting data compressed	Not needed, compression starts immediately
Data on disk	Compressed	Compressed
Data on DB2 log	Compressed	Not compressed
Data in buffer pool	Compressed	Not compressed
Image copies	Compressed	Not compressed

Table 3: Data compression vs. index compression

The compression ratio depends on the data and on how the columns are distributed in the index. The buffer pool choice determines the maximum disk space savings and the efficiency of memory utilization.

With index compression, DB2 stores index data on disk in 4 KB pages. The compressed data is expanded into buffer pool pages of 8 KB, 16 KB, or 32 KB. The choice of the buffer pool page size is a design decision. This selection, along with how well the index data compress, defines the efficiency of the process and the storage savings. The maximum amount of disk storage saving you can achieve when using index compression is limited by the index buffer pool:

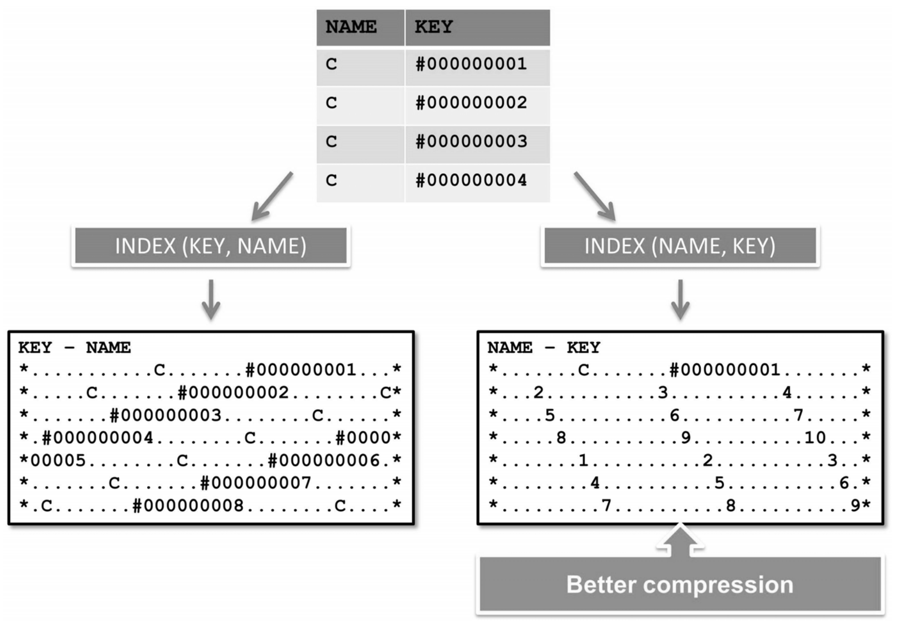
- For 8 KB, up to 51 percent compression
- For 16 KB, up to 76 percent compression
- For 32 KB, up to 88 percent compression

The choice of an inappropriate buffer pool page size can result in either wasted buffer pool memory or wasted DASD storage.

The DB2-provided standalone utility `DSN1COMP` supports index compression. It can be used to estimate the space savings on disk due to index compression and to select the buffer pool size that best fits the index. This example shows the output of one execution:

```
  8  K Page Buffer Size yields a
 40  % Reduction in Index Leaf Page Space
 60  % of the original index's Leaf Page Space
 16  % of Bufferpool Space would be unused to
     --------------------------------------------
 16  K Page Buffer Size yields a
 41  % Reduction in Index Leaf Page Space
 59  % of the original index's Leaf Page Space
 57  % of Bufferpool Space would be unused to
     --------------------------------------------
 32  K Page Buffer Size yields a
 41  % Reduction in Index Leaf Page Space
 59  % of the original index's Leaf Page Space
 78  % of Bufferpool Space would be unused to
```

The decision to deploy index compression should be made carefully and should be implemented on a case-by-case basis as much as possible. Because of the nature of the prefix compression algorithm, two indexes composed of the same columns but arranged in a different sequence could compress remarkably differently. In Figure 29, the printout of the VSAM pages of two compressed indexes shows that less information is stored in the page for the column sequence NAME, KEY. This index compressed more than the one with the sequence KEY, NAME.



NAME	KEY
C	#000000001
C	#000000002
C	#000000003
C	#000000004

```
KEY - NAME
*...........C.......#000000001...*
*.....C.......#000000002........C*
*.......#000000003........C......*
*.#000000004........C.......#0000*
*00005........C.......#000000006.*
*.......C.......#000000007.......*
*.C.......#000000008........C....*
```

```
NAME - KEY
*.......C.......#000000001.......*
*...2..........3..........4......*
*....5..........6..........7.....*
*.....8..........9..........10...*
*.......1..........2..........3..*
*........4..........5..........6.*
*.........7..........8..........9*
```

Figure 29: Impact of index column order on index compression

In general, a data warehouse environment is an excellent candidate for index compression. You may consider index compression where a reduction in index storage consumption is more beneficial than a possible increase in CPU consumption.

Index compression may not be a convenient choice for OLTP environments. However, larger index pages can be used independently of index compression. A larger index page can provide performance benefits to OLTP transactions, reducing the number of index levels and preventing index page splits.

Reducing TCO with Faster Analytics

DB2 and System z are platforms of choice for decision support environments worldwide. DB2 for z/OS has been adding warehousing- and analytics-specific functionality with each new release. Today, it is possible to build an end-to-end enterprise warehousing solution on System z that includes data extraction, transformation, load, query, and reporting functionalities.

Analytics and warehousing queries are complex, highly resource-intensive workloads that require organizations to process massive amounts of data. Big data is a reality today, and data volumes will only continue to grow. In addition, the dynamics of global competition mean that companies must leverage the information in their systems quickly to help them make smart, informed strategy decisions.

DB2 for z/OS provides capabilities that can deliver the analytics that businesses require from existing DB2 data. It can leverage existing data into faster analytics. These capabilities save the need for third-party database or new infrastructure developments and, therefore, contribute to total cost of ownership savings.

IBM DB2 Analytics Accelerators

In many situations, the speed at which analysts can understand and react to changes in a dynamic business environment is limited by the IT infrastructure's ability to process complex queries against large volumes of data. Ideally, business analytics would be unconstrained by infrastructure limitations. Organizations could leverage near-instantaneous analytics and run reports when they need them, even in real time. This is where DB2 query accelerators come into the picture.

A DB2 query accelerator is a transparent-to-the-user tool designed to boost database speed and performance. Accelerators can provide dramatic improvements in response time and reduce CPU utilization by offloading eligible queries to specifically designed hardware. They can help deliver faster, more reactive business insight by executing analytics when they are required and at high speed.

The IBM DB2 Analytics Accelerator can dramatically improve performance and reduce the cost of analytics in DB2 for z/OS environments. Available as an add-on appliance built on IBM Netezza® technology, the accelerator is designed for easy, rapid deployment. Users simply instruct DB2 to consider the query accelerator the new access path for eligible queries on eligible objects. The appliance requires very little configuration, and accelerator performance information is integrated with the usual DB2 traces.

For candidate queries, the results can be astonishing. Queries often run significantly faster than they ever have before. Response times for well-tuned queries running on current-generation traditional hardware can shrink from hours to seconds. Queries that previously ran too slowly to be useful can often be completed in minutes. As response times approach what you might expect from an OLTP environment, realtime analytics can become an everyday reality.

Potentially, a good portion of the CPU consumed by queries running in DB2 can be eliminated by offloading the query processing to the accelerator. Nevertheless, offloaded queries returning a large result set could consume a noticeable amount of CPU time in DB2. This scenario is improved with a performance enhancement introduced in DB2 11 for z/OS.

Figure 30 shows the CPU utilization and evolution of the MSU 4-hour rolling average during the execution of a data warehousing workload. This scenario is the combination of short, medium, and long intensive queries executed entirely in DB2.

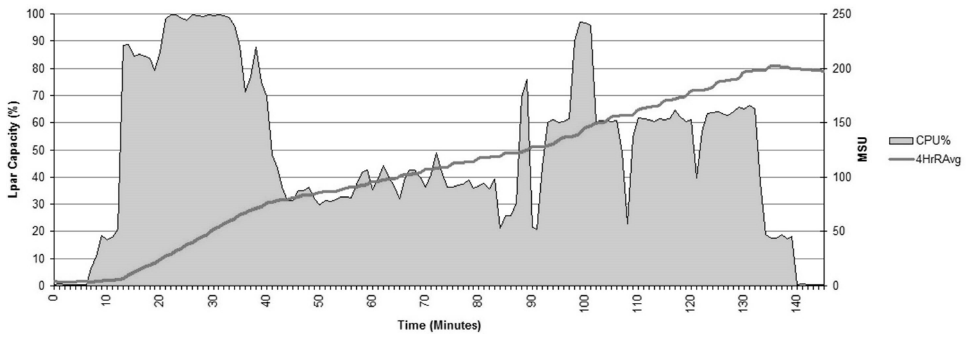


Figure 30: Workload before acceleration – Total LPAR CPU utilization

Figure 31 shows the same scenario after adding an accelerator appliance to the environment. The most complex queries were offloaded to the accelerator. As a result of the faster accelerator performance, the whole scenario was accomplished in a fraction of the original elapsed time. Most of the z/OS CPU was removed from the LPAR as queries were executed in the accelerator, reducing the impact in the MSU 4-hour rolling average, which is the usual software license cost metric.

For clarity, the chart scale is unchanged. This example shows how eligible workload can be executed faster and less expensively in the accelerator.

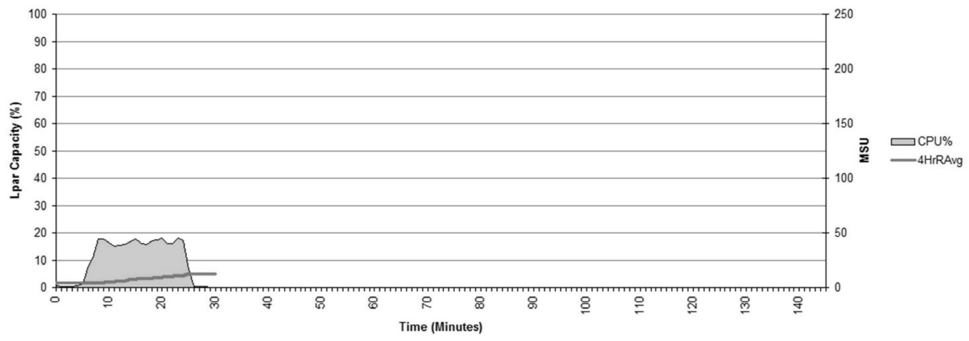


Figure 31: Workload after acceleration – Total LPAR CPU utilization

The secret of this speed resides in the highly specialized hardware and software that is tuned for serving analytics queries. The DB2 Analytics Accelerator appliance exploits massive parallel processing on dedicated CPUs, disks, and memory in a highly and linearly scalable architecture.

Figure 32 depicts the relationship between an application, DB2 for z/OS, and the accelerator. The accelerator appliance connects to z/OS and DB2 using a private network. The DB2 objects to be accelerated are defined and loaded in the accelerator using DB2 stored procedures and a graphical user interface. Once an accelerator is installed and enabled to work, the DB2 optimizer considers the appliance as a new access path and will offload SQL processing transparently when it judges that doing so would be more efficient.

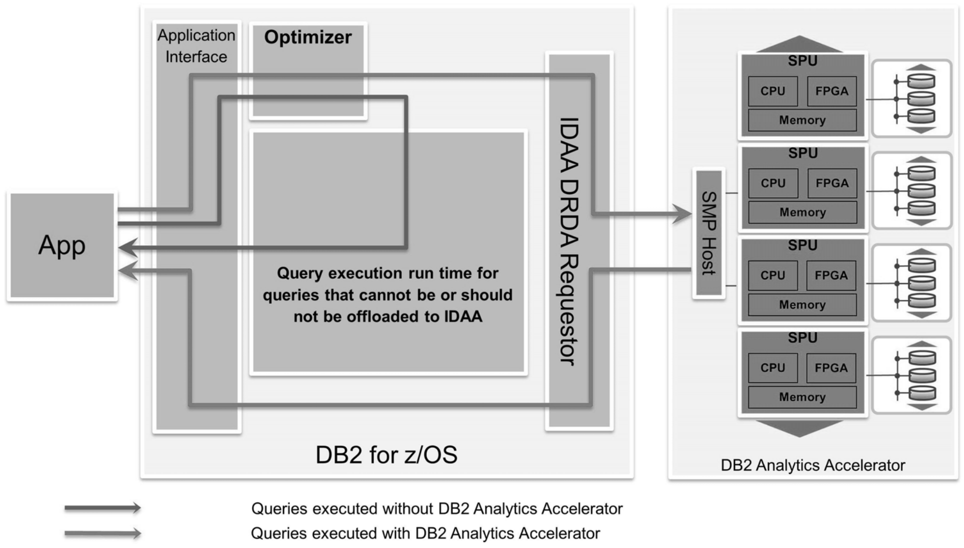

Application Interface
Optimizer
IDAA DRDA Requestor
App
Query execution run time for queries that cannot be or should not be offloaded to IDAA
DB2 for z/OS
SMP Host
SPU
CPU
FPGA
Memory
DB2 Analytics Accelerator
Queries executed without DB2 Analytics Accelerator
Queries executed with DB2 Analytics Accelerator

Figure 32: IBM DB2 Analytics Accelerator and DB2 for z/OS

The business value of a query accelerator resides in its close and transparent integration with DB2 and System z. You get hyper-speed analytics in the highly reliable, secure, and stable mainframe environment that you already know and love. DB2 Analytics Accelerator can not only help reduce total cost of ownership but also change how analytics are executed on DB2 and System z, the platform of choice for enterprise warehousing infrastructures.

Leverage Legacy QMF Objects

Query Management Facility™ (QMF™) was part of the initial DB2 announcement, 30 years ago in 1983. Many users have been working with QMF since the beginning of its history with DB2 for z/OS. As a result, it is common to find organizations with a long history, and an extensive catalog, of 3270-created QMF queries and reports. Nevertheless, the classic 3270 interface may not be up to the level of the graphical reporting capabilities that today's modern business require.

The evolution of the QMF target user base could be summarized as follows:

- Database administrators: QMF for TSO and High Performance Option (HPO)
- Technical users: QMF for Windows®, an extension of QMF to the desktop
- Data analysts, IT professionals: QMF for Windows/WebSphere, an extension of QMF to a web browser that also introduces graphical reporting

- Business users: QMF for Workstation/WebSphere, which provides an intuitive visual solution with a personalized, nontechnical graphical user interface and supports enhanced security, OLAP, and a variety of databases on top of DB2 for z/OS

The QMF Enterprise Edition product interoperability lets users leverage legacy QMF queries into a graphical user interface with modern reporting capabilities. For example, users can develop, edit, and run QMF queries, forms, reports, and procedures via the 3270 user interface. These QMF objects are stored in the QMF catalog within DB2 for z/OS. Users working with QMF for Workstation and WebSphere can access the same QMF catalogs and use the same object format. Queries, reports, and procedures created in QMF for TSO can be accessed, edited, and executed in QMF for Workstation and WebSphere, and vice versa. All changes made in one QMF product are immediately accessible in all other QMF products. Users can also initiate the z/OS-based execution of QMF objects from the QMF for Workstation and/or WebSphere products.

QMF advantages that can help reduce TCO include:

- QMF for Workstation eliminates the overhead of a TSO session and a QMF for TSO session during development of queries and reports, resulting in reduced z/OS CPU utilization.
- QMF High Performance Option converts QMF queries, forms, and procedures to compiled COBOL programs and DB2 COBOL stored procedures. This option can provide better performance, reduced CPU utilization, and predictable performance.
- Existing reports can be reused by the QMF family rather than being re-created in another tool, providing smooth extension of QMF reports to other areas of the business.
- The ability to use one interface and one tool to access many data sources (including DB2 for z/OS, DB2 for iSeries®, DB2 LUW, Oracle®, Microsoft SQL Server®, Microsoft Excel®, and others) increases productivity.

QMF Enterprise Edition is currently available in the form of these solutions:

- **QMF for TSO/CICS.** Supports end users who work entirely from traditional mainframe monitors to access databases in the IBM DB2 family
- **QMF HPO.** The High Performance Option for QMF for TSO/CICS is a multifaceted tool that helps database administrators manage QMF objects and increase performance in TSO and CICS environments.
- **QMF for Workstation.** Extends the key functionality of QMF for TSO/CICS on distributed platforms via a rich desktop development environment. Runs on Microsoft Windows, Linux, and Apple computer systems.
- **QMF for WebSphere.** Provides a feature set similar to that of QMF for Workstation, using a thin-client, browser-based solution and mobile device support. Runs on Windows, Linux, Solaris, AIX®, HP-UX, iSeries, z/Linux, and z/OS.

QMF for Workstation and WebSphere features that may be of interest are:

- Ad hoc and prepared queries
  - Create, edit, and reuse QMF queries
  - Apply groupings, aggregations, conditional formats, and more
  - Perform one-click export to Excel, data files, or database table
  - Perform one-click generation of reports from the data
- Tabular and graphical reports
  - Generate QMF tabular reports or graphical visual reports
  - Create highly customizable, page-based layouts
  - Draw data from any number of data sources
- QMF dashboards
  - Customized, interactive data visualizations
  - Rapid authoring model
  - Technical details

Reducing TCO with Improved Scalability

DB2 10 provides impressive reductions in memory requirements below the 2 GB bar. In prior releases, these requirements limited the concurrent number of threads and DB2 scalability. DB2 10 supports five to ten times more concurrent users per DB2 subsystem. DB2 logging and the internal DB2 serialization mechanisms (latches) have also been improved, giving overall increased throughput per DB2 subsystem.

These are examples of changes that increase the workload that a single DB2 subsystem can handle. This situation provides new consolidation opportunities where users may reduce the number of DB2 subsystems for the same workload.

DB2 subsystem consolidation and better DB2 scalability can help to reduce total cost of ownership by simplifying the IT infrastructure and reducing the data center footprint. DB2 consolidation can yield lower administrative and staff costs.

DB2 10 Throughput Enhancements

When data becomes unpredictably volatile, or when the amount of data increases, there can be performance problems related to throughput and bottleneck issues. This situation is often observed during nightly batch jobs. Many organizations are constrained in that they must wait for critical nightly batch processing to end before being able to begin their business day processing (OLTP). If the end of the batch process is delayed, it means a late start to the daytime activity. Depending on the company and its business activity, this service degradation can have a large financial impact.

DB2 10 can improve the throughput of data-intensive processes. Changes such as logging enhancements, latching contention relief, dynamic prefetch enhancement, and

I/O parallelism for index updates provide better performance, resulting in an improved throughput without the need to change existing applications.

From a performance point of view, mass insert applications need particular attention. The DB2 10 insert performance improvements help improve mass insert performance. The observed performance benefits vary with the workload characteristics.

Some of the biggest improvements were observed for high-volume, concurrent insert processes in data sharing environments. MEMBER CLUSTER support for universal table spaces is made available in DB2 10 NFM. This feature can help improve insert performance in data sharing environments.

Out-of-the-box enhancements related to insert performance include space search improvement, index I/O parallelism, log latch contention reduction and faster COMMIT processing, support for MEMBER CLUSTER in universal table spaces, and LRSN spin loop avoidance.

DB2 latching is used for short-term serialization of DB2 resources such as storage or control blocks, pages in buffer pools, or log write output buffer pages. Latches are usually very inexpensive to acquire. Under heavy load, they can become a bottleneck. The most common source of problems are DB2 log contention (in latch class 19) and prefetch latch or EDM LRU chain latch (included in latch class 24).

Most DB2 latches that could impact scalability were enhanced in DB2 10. This list summarizes some of the areas that were improved:

- **LC12.** Global transaction ID serialization
- **LC14.** Buffer manager serialization
- **LC19.** Log write in both data sharing and non data sharing
- **LC24.** EDM thread storage serialization
- **LC24.** Buffer manager serialization
- **LC25.** EDM hash serialization
- **LC27.** WLM serialization latch for stored procedures/user-defined functions
- **LC32.** Storage manager serialization
- **IRLM.** IRLM hash contention
- **CML.** z/OS Cross-memory local suspend lock
- **UTSERIAL.** Utility serialization lock for SYSLGRNG (NFM)

Logging performance is enhanced in DB2 10. Log synchronous writes performance is improved. In some scenarios, DB2 logging suspension time drops by 50 percent compared with DB2 9. DB2 10 log asynchronous writes performance changes, in combination with disk storage enhancements, help increase logging asynchronous throughput that benefits massive batch processing.

All these improvements have the potential to reduce the elapsed time required by batch processing. Batch jobs might end sooner after migration to DB2 10. Having the overnight processing end sooner provides an additional buffer before the opening of the OLTP window. This extra buffer can sometimes compensate for unexpected peaks in batch processing, seasonal treatments, or operational problems during night processing.

The value of this extra buffer is difficult to measure in financial terms. Nevertheless, it can help avoid service disruptions and delays, which are quantifiable depending on the activity. The extra buffer can provide a better quality of service, increasing customer satisfaction.

DB2 Storage and Scalability

The addressable storage below the 2 GB bar, also known as *BTB*, has been the scalability limit of DB2 for z/OS in many scenarios. At a glance, the available storage BTB limits the number of concurrent threads that a single DB2 subsystem can handle. DB2 for z/OS has been improving its scalability with each new release by moving internal structures from BTB to above the bar. Figure 33 summarizes this evolution.

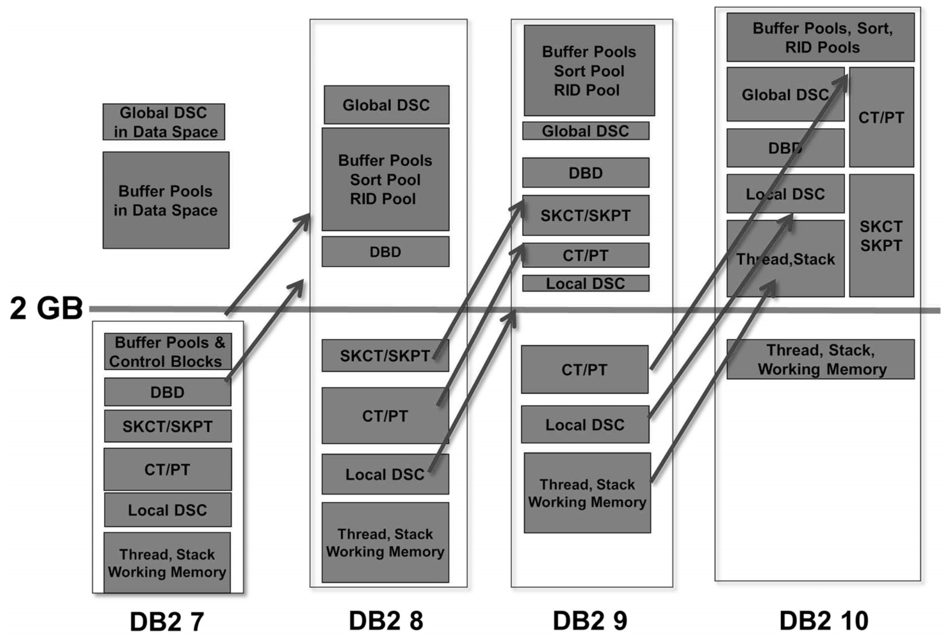

2 GB
Global DSC in Data Space
Buffer Pools in Data Space
Buffer Pools & Control Blocks
DBD
SKCT/SKPT
CT/PT
Local DSC
Thread, Stack Working Memory
Global DSC
Buffer Pools Sort Pool RID Pool
DBD
SKCT/SKPT
CT/PT
Local DSC
Thread, Stack Working Memory
Buffer Pools Sort Pool RID Pool
Global DSC
DBD
SKCT/SKPT
CT/PT
Local DSC
CT/PT
Local DSC
Thread, Stack Working Memory
Buffer Pools, Sort, RID Pools
Global DSC
CT/PT
DBD
Local DSC
SKCT SKPT
Thread,Stack
Thread, Stack, Working Memory
DB2 7
DB2 8
DB2 9
DB2 10

Figure 33: DB2 storage scalability evolution

Before DB2 10, a safe number of concurrent threads per DB2 subsystem was counted as a few hundred. Although recent DB2 versions have made improvements in this area, virtual storage BTB remains the most common constraint. This constraint is commonly known as Virtual Storage Constraint (VSC).

DB2 10 for z/OS provides a significant step forward in scalability by moving a large portion (from 50 to 90 percent) of the BTB storage to 64-bit virtual storage. These benefits are available to dynamic SQL immediately after migrating to DB2 10. Static SQL using packages will benefit after a REBIND.

Figure 34 compares the DB2 storage BTB between DB2 9 and DB2 10 for the same workload. The DB2 10 memory changes substantially increase the number of concurrent threads that can be supported by a single DB2 subsystem. DB2 10 has the potential to support five to ten times more concurrent users and up to 20,000 concurrent users in a single DB2 subsystem.

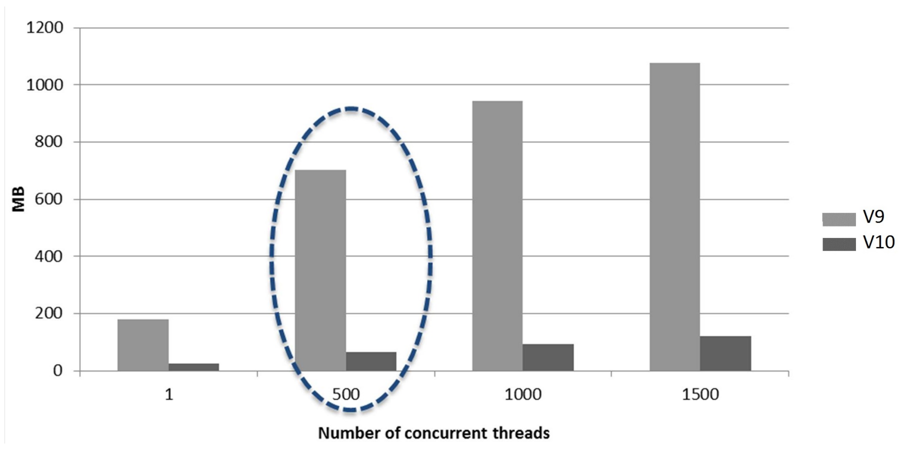


Figure 34: DB2 storage (DBM1) below the 2 GB bar

Memory enhancements BTB also allows for BIND and REBIND options that may provide CPU reduction benefits at the expense of a bigger thread footprint. This CPU versus storage tradeoff, which was not affordable before DB2 10, becomes a performance opportunity. These options include a larger utilization of RELEASE(DEALLOCATE) and larger MAXKEEPD values for KEEPDYNAMIC=YES.

Nevertheless, storage enhancements BTB comes at the expense of higher overall real storage requirements. That is, for the same workload, DB2 10 uses a lot less memory BTB, but it may require up to 10 to 30 percent more real storage. *Real storage* refers to the actual physical memory available in the LPAR. Over-commitment of real storage, which could just be a consequence of migrating to DB2 10, will result in system paging. System paging is a less desirable situation and may degrade LPAR performance overall. To an extent, virtual storage BTB concerns before DB2 10 may translate to real storage concerns after DB2 10. The monitoring focus shifts from virtual to real storage. Fortunately, the acquisition cost of real storage has been dropping substantially with the latest System z generation of servers.

IMPORTANT: DB2 11 continues this trend by further moving DB2 structures from BTB to above the bar.

Data Sharing Member Consolidation

With DB2 releases prior to DB2 10, organizations may install data sharing environments to be able to absorb a larger number of concurrent threads than allowed with a single DB2 subsystem. Figure 35 depicts the case of a DB2 data sharing environment with co-location of data sharing members in the same LPAR.

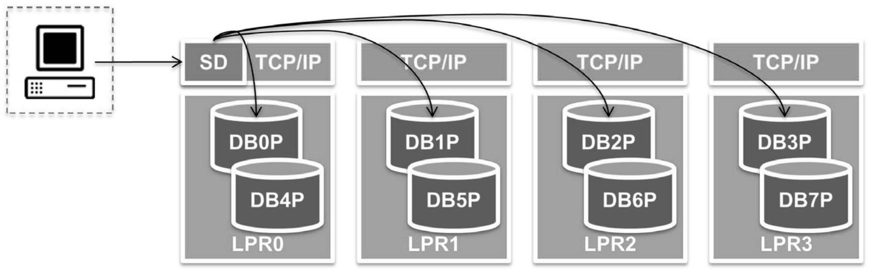


Figure 35: DB2 data sharing environment with co-location of data sharing members

Organizations running DB2 9 and experiencing virtual storage constraints can consider consolidating DB2 subsystems after moving to DB2 10. DB2 10 also improves in other areas, making it possible to run more work in a single DB2. These changes include enhancements in the logging subsystem and latching contention relief. Figure 36 shows how the previous example would look like after data sharing member consolidation of the co-located DB2 subsystems.

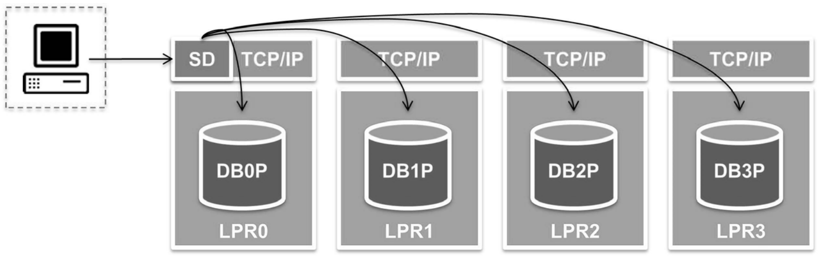


Figure 36: DB2 data sharing environment after data sharing member consolidation

Data sharing member consolidation provides a simpler infrastructure and has the potential to reduce administration and maintenance costs. In addition, consolidation may provide 0.5 percent CPU savings for each member removed from the group and more savings on real storage.

In complex installations with multiple LPARs, there is a potential for even further consolidation by eliminating now-redundant LPARs. DB2 10's combination of CPU reduction and memory enhancements makes it possible to consider an LPAR consolidation scenario.

As an example, Figure 37 shows a four-LPAR, eight-member DB2 data sharing group that consolidated from a 16-member data sharing group after migrating to DB2 10. Here, the last two LPARs could be removed and their workload transferred to the remaining partitions. This scenario has the potential to further reduce total cost of ownership by data center simplification.

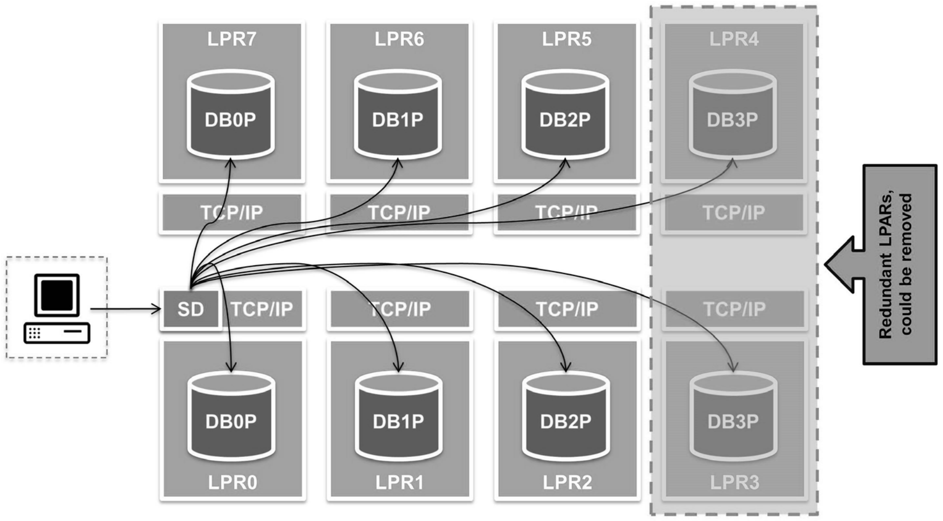


Figure 37: Data sharing member consolidation after migration to DB2 10

Nevertheless, before planning member consolidation, keep in mind that DB2 data sharing is a high-availability solution, as well. Multiple DB2 members, in combination with DB2 Sysplex Support, provide applications with seamless high availability and workload balancing. From an availability point of view, the ideal minimum number of DB2 data sharing members is four, with two at a remote location.

Conclusion

DB2 for z/OS has features that can help to fulfill today's needs and reduce total cost of ownership, a key area of concern for today's organizations. As has been demonstrated in this paper, the main areas to investigate for potential benefits are synergy with System z, CPU utilization, performance, storage, analytics, and scalability. DB2 for z/OS has the power to deliver for the business in all of these areas.

Acknowledgments

Thanks a lot to the following people for their invaluable contributions to this paper:

- Katrin Noack, Belgium
- Surekha Parekh, World-Wide Marketing Program Director – DB2 for z/OS
- Terrie Jacopi, Program Director, DB2 for z/OS Information Management
- Katie M. Tipton, Book Editor, MC Press Online, LLC