DB2 Universal Database Performance Tuning

Berni Schiefer
IBM Toronto Lab
1150 Eglinton Avenue East
Toronto, Ontario, Canada M3C-1H7
schiefer@ca.ibm.com

Gary Valentin
IBM Toronto Lab
1150 Eglinton Avenue East
Toronto, Ontario, Canada M3C-1H7
valentin@ca.ibm.com

Abstract

DB2 Universal Database is the flagship Relational Database from IBM. By building on decades of ground-breaking IBM Database Research projects, DB2 Universal Database benefits from a pervasive and continuous infusion of advanced technology ranging from first-class optimizer and compiler technology to advanced locking, logging and recovery mechanisms, all designed with a common thread of high performance in both OLTP and complex query environments. This article will take the reader on a whirlwind tour of DB2 Universal Database, with a focus on the new auto-configuration tools. Find out how its open platform support exploits a rich repertoire of data storage techniques and how the latest Java-based administration tools can be used to quickly build and configure a DB2 Universal Database - for maximum performance.

1 Introduction

DB2 Universal Database is the flagship Relational Database from IBM. By building on the ground-breaking IBM Database Research projects, such as the relational model[CODD70], System R [ASTR76], R* [LMH+85], Starburst [HCL+90], and the SQL language itself [CHAM76], (see [DB299d]) DB2 Universal Database benefits from a pervasive infusion of advanced technology ranging from first-class optimizer and compiler technology to advanced locking, logging and recovery mechanisms, all designed with a common thread of high performance in both OLTP and complex query environments. Today DB2 Universal Database counts performance as one of its strengths.

This article will take the reader on a whirlwind tour of the tuning process in DB2 Universal Database, how its open platform support exploits a rich repertoire of data storage techniques and how the latest Java-based administration tools can be used to easily build and configure a DB2 Universal Database for maximum performance.

Also we include an introduction to the auto-tuning tools in DB2 Universal Database. These tools are geared towards simpler database administration and quick tuning - bringing forth high-performance database technology while simultaneously reducing the total cost of ownership.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

2 Configuring the hardware

2.1 Disk striping and RAID

RAID 0 is the simplest of all disk layout schemes. In RAID 0, the data is not mirrored, but simply laid out in stripes across the available disks. But often RAID 0 is not an option. This happens when the data volume is very large (100 GB+), and disk failures are prone to happen frequently. Otherwise - RAID 0 gives good read performance and optimal write performance. For databases with fast recovery time, RAID 0 can be the most cost-effective solution.

RAID 1 keeps 2 copies of each page. This means slower writes/faster reads. In DSS workloads this is acceptable because they rely heavily on read performance. In OLTP workloads with heavy write/update activity, this will cause a performance degradation.

RAID 5 is used when disk redundancy is a must - but the budget is low. RAID 5 can be slower than RAID 1 because a write requires more processing for parity calculations.

RAID 10 is a combination of RAID 1 and RAID 0. It preserves the disk reliability of RAID 1 while approaching the performance of RAID 0. Note however, that the financial implications become significant. This configuration is the most expensive of the four described here, and is often deployed in database projects which are both mission-critical and performance sensitive.

Striping is done on two levels. The first level is the striping done by the disk arrays such as RAID, the second level is the striping of data onto separate disk arrays by DB2 UDB. On the RAID level, we make the distinction between a strip size[1] and a stripe size. On the database level, we make the distinction between an extent size[2]. and a prefetch size. A good rule of thumb is to make one stripe size equal to one extent size, so that the RAID striping matches the database striping. To better grasp the concepts behind striping, we will work through a disk layout example. In this example, we take an OLTP database, and place it on 8 RAID arrays, each RAID array containing 12 disks.

The smallest element is a strip, and it represents one disk on a RAID array. A large strip size (for example, 4 KB per disk) means that a single table update will only cause writing to one disk. So this is good for OLTP. Alternatively, a thin stripe would cause the disks to fill up the bus bandwidth more readily, pushing the bus to its full potential. Continuing with our OLTP example, we make the following calculation for RAID parameters:

- One strip size for OLTP = 4 KB
- One stripe size = (number of disks) * (strip size) = 12 * 4 KB = 48 KB

Next we deal with the striping on the database level. We have at our disposal 8 such arrays, which DB2 will see as 8 distinct containers. To take full advantage of the disk arms, we will spread the data across all 8 arrays.

- One extent size = One stripe size = 48 KB
- One prefetch size = multiple of ((extent size) * (number of containers))
- One prefetch size = multiple of (48 KB * 8)
- One prefetch size = 384 KB

[1] A strip size refers to the number of bytes in a stripe which are written to a single disk before continuing to the next disk on the stripe

[2] An extent size is the number of database pages which are written out to a single container before moving on to the next container in the table space

Here is a summary of the results for an OLTP system with 8 RAID arrays of 12 disks:

RAID strip 4 KB

RAID stripe 48 KB

DB2 extent 48 KB

DB2 prefetch size 384 KB

2.2 Memory Layout

In the simplest sense, memory is about caching parts of your database in memory to avoid disk I/O. DB2 Univeral Database supports the concept of buffer pools, which are large memory-caches of data.

- Non-leaf pages of indexes should be in memory. This means that an index-search can be completed using just one I/O.
- DB2 uses more memory than just for buffer pools. The two most important elements are sort memory and utility memory.

  Sort memory is controlled by the Sort Heap Size and Sort Heap Threshold configuration parameters.

  Memory used by utilitites is controlled by the Utility Heap Size configuration parameter, and is important for utilities such as LOAD and BACKUP.
- The remaining free memory can be allocated for buffer pools.

2.3 Disk Read/Write Cache

Disks have carried Read caches for a long time, and the purpose of these caches was to take advantage of locality in read patterns in the system behavior. Recently, with technology advancements, disks have begun sporting write caches as well. These caches are safe because they are static, and the information is preserved even when the power is cut off to the disks. These caches can be used to great advantage.

An obvious solution is to always enable read and write caches on all disks. But some systems will require the system administrator to make a tradeoff between read and write caches, by enabling the cache as a pure read cache, a pure write cache, or some measure in between. Also, a tradeoff may be required when purchasing the disks. Here are some recommendations to take full advantage of Read/Write caches.

Write caches will help log disks, especially for workloads with a high transaction rate. Write caches are useful for workloads containing occasional updates/inserts/deletes. Read caches are not as useful because the buffer pool does a better job of determining locality in data access. A good combination is to enable maximum read cache for OS and DB2 installation image, while enabling the write cache for data and log disks.

3 Building the Database

3.1 Table Space Layout

Data layout in table spaces is a key determining factor in database performance. The first rule is to take advantage of all the disks in every case. All table spaces should be split across all disks. DB2 Universal Database supports several methods for the layout of table spaces. The first such method is called DMS (Database Managed Space). DMS is a good choice for extra performance, but reduces usability somewhat. DMS table spaces let the database bypass the filesystem function supplied by the operating system and let the database control the disks directly. The net result is faster access to the data on the disk, and no interference from the operating system for journaling, caching, re-ordering, and other tasks. SMS (System Managed Space) is a strong alternative with good performance and high managability. It gives the user the freedom to take advantage of the filesystem functionality, which means that the database will be able to share disk resources with other applications.

3.2 Loading the Data

Loading the data is often a resource-intensive operation. Load is self-tuning and will adapt the loading process for those cases where large amounts of data are involved.

LOAD is a demanding high-volume I/O operation, it is recommended to take advantage of disk parallelism. If the source files can be placed across many disks (striped or otherwise) and/or there are several CPUs on the target system, then LOAD will recognize the available resources and use them.

Another method to improve LOAD performance is to streamline the LOAD operation with the subsequent gathering of statistics and index creation. DB2 Universal Database offers the opportunity of re-building the indexes and gathering statistics during the LOAD.

By telling the LOAD operation more about the nature and requirements of the task, the performance can be improved even further. For example, LOADs can be performed in a non-recoverable fashion, or more memory can be assigned for the database utility processing. All added information provides potential for increasing LOAD performance.

3.3 Creating Indexes and Gathering Statistics

The main indexes are built, when possible, together with the LOAD. However, circumstances may require that additional indexes be created. This may happen if new, and unexpected queries are submitted for processing. When considering indexes, the database administrator is often concerned with the tradeoff of disk space versus query performance, and query performance speedup versus potential insert/update/delete slowdown.

Creation of the largest indexes might be slow if not enough resources are available to perform the task. There are several things which can help to make it faster. The most obvious of these improvements is to give the index creation process more memory to work with by increasing the buffer pool, increasing the sort heap, enabling the SMP parallelism, or ensuring that the temporary table space is large enough and spread out across several disks.

In order to function optimally, DB2 Universal Database uses reliable and up-to-date statistics. There are several levels of statistics which can be collected, beginning with general information to detailed distribution statistics. For example, when the data values are skewed, the quality of the statistics will benefit greatly from the added information on distribution.

In some cases, the tables and indexes which are subject to this operation do not contain the data which they will contain during normal operation, and as a result produce misleading statistics. These tables are often referred to as ”volatile tables” simply because their contents change quickly in terms of size and values. Any table can be marked as having these properties by using the VOLATILE option on the ALTER TABLE command. It is essential that statistics are gathered at a time when the table contains typical values and typical amounts of data. Lastly, statistics must be gathered regularly, and the data can be reorganized to ensure its clustering. It is best to set up these operations on a scheduled basis so as not to forget.

4 Taking Advantage of the Auto-Tuning Tools

The configuration of DB2 Universal Database is substantially simplified and accelerated due in part to the auto-configuration and auto-tuning tools. Most of these algorithms and tools have been integrated into the database engine and are invisible to the database user.

This section covers two auto-tuning tools which have been externalized in DB2 Universal Database in the form of SmartGuides. By making these tools open, the user has the opportunity to supply more information about the workload, and become aware of the configuration changes brought on by the tools.

4.1 The Database Configurator Tool

One auto-configuration tool included in DB2 Universal Database is the Database Configurator Tool [3]. The purpose of this tool is to relieve the daunting task of learning about each configuration parameter and performing measurements to find each optimal value.

The auto-configurator tool does rely on some basic questions which it asks the user in order to better tune the underlying database.

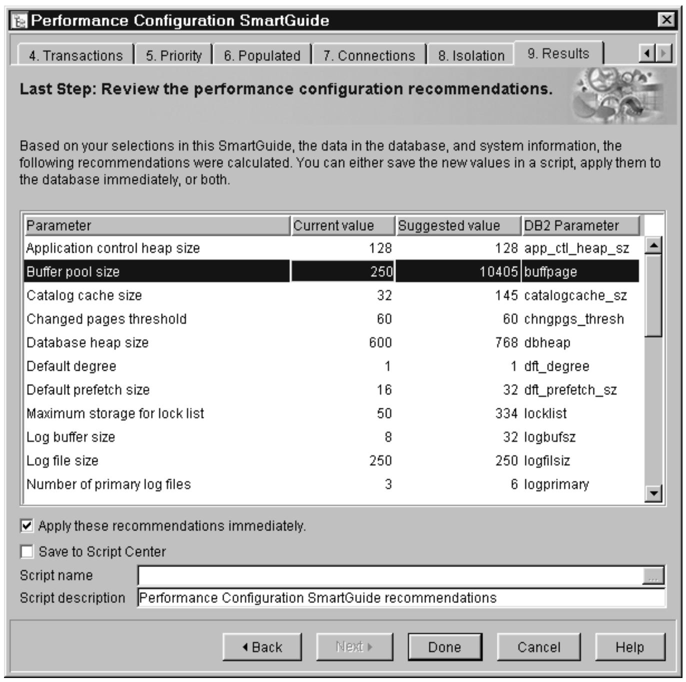


Figure 1: Analyzing the results

Figure 1 displays the resulting configuration parameters. This window reveals two things. The first is the number of configuration parameters, as you can see from the scrollbar, a modern commercial database is extremely configurable, and for many users it is a little too configurable.
The second thing made obvious in this picture is the recommendation for buffer pool size. As you can see, the default value of 500 pages (1 MB) greatly underutilizes the available resources! This machine had 128 MB of memory, and the tool decided to allocate 40 MB for buffer pool memory.

4.2 The Index Advisor

An important and time-consuming question for every database administrator is determining the right set of indexes. It requires an understanding of the dimensions of the data, of the workload exercised against the data, and

[3]To start the Performance Configuration SmartGuide, from the Control Center, click with mouse button 2 on the database for which you want to configure performance. Select 'Configure Performance' from the pop-up menu. The Performance Configuration SmartGuide opens. Follow the steps in the SmartGuide.

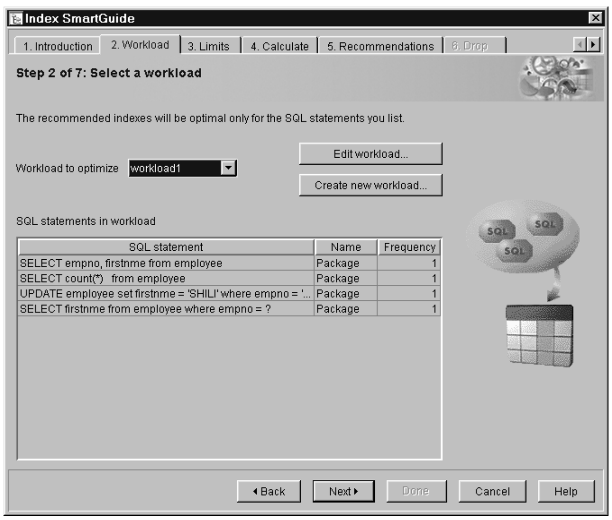
of the optimizer's methods in choosing an access plan. When the number of tables and columns runs into the thousands, no single administrator can handle this task without making a mistake along the way.

The Index SmartGuide [4] is a visual tool in DB2 Universal Database which helps the administrator choose indexes, and in some cases it relieves the administrator of the task altogether.

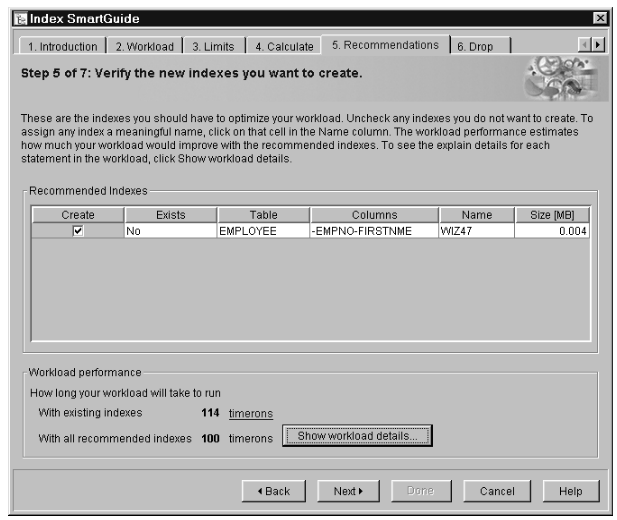
The main concept behind index advising is to suggest indexes to the administrator based on a representative SQL workload, and the underlying statistics of the data. The recommendation engine will take advantage of the optimizer and use a cost-based approach instead of heuristics or rules.

The index recommendation algorithm also considers the cost of maintaining indexes. In other words, if the workload contains many insert/update/deletes then some indexes can have a negative impact on performance because they need to be maintained. These impacts are taken into account. So it is better to include insert/update/delete statements as part of the workload description, to guarantee optimal recommendations.

The following snapshots show the two basic steps in the Index SmartGuide:



(a)



(b)

Figure 2: Index Advisor

Figure 2(a)illustrates collecting the workload. The Index SmartGuide, when started, already knows about a workload by searching through the SQL cache, which stores recently executed dynamic statements. Other sources of SQL may be tapped, such as SQL stored in application packages and SQL stored in event monitor files. SQL can also be entered manually.

Figure 2(b) illustrates making recommendations. The recommendations are shown with details on tables, columns, size in megabytes, and direct improvement on total workload time. In this example, the workload was improved by only 10%.

5 Advanced Topics

As we have seen, some basic principles for the physical machine and physical database design as well as the use of intelligent tools to guide the tuning of the database can go a long way to providing an efficient well-tuned DBMS system. However, in certain application environments with unique requirements, or to achieve an even

[4] To start the Index SmartGuide click mouse button 2 on the Indexes folder and select Create - Index using SmartGuide from the pop-up menu. The Index SmartGuide opens. Follow the steps in the SmartGuide.

more highly tuned system, the DBA can use a wide range of additional functionality to control the behavior of the database very precisely. In this section we will explore some of this additonal functionality.

We begin with the physical database design. The use of multiple table spaces can give fine-grained control over the placement of data. For example, separating data and indexes into different table spaces or placing table data on only a subset of the disks available. The use of multiple table spaces also allows the database to be managed at a more granular level since many of the database utilities operate at a concurrency granularity of a table space (e.g. load, backup/restore). The characteristics of each table space can also be controlled, including defining a page size ranging from 4 KB to 32 KB, the size of an extent placed on each disk, the number of pages to prefetch from the table space, and a summary of the performance characteristics of the underlying disk devices.

Once multiple table spaces are in place, multiple buffer pools can be created to specifiy the caching behavior. Groups of tables representing one workload can be given priority over another group by allocating a proportionately larger percentage of available memory to one group. Alternatively, certain large tables can be restricted to a small buffer pool. In this case, large table scans will not compete with the buffer pool memory used by indexes, which is needed to avoid expensive random accesses. The result is better concurrency and more reliable performance characteristics.

The ALTER TABLE statement can also be used to more precisely define the role of each table and the access to it. For example, read only tables can choose to use a less granular level of locking using the LOCKSIZE option. Tables which log changes via continuous INSERT activity can benefit from specifying this using APPEND MODE. Space for future update activity can be reserved by using PCTFREE which controls how densely the LOAD utility populates the table pages.

Once the tables are defined and an initial set of indexes exist, the behavior of the indexes can also be specified using options on CREATE INDEX. For example, the freespace on each index page can increase or decrease depending on table usage using the PCTFREE option. Also, the set of columns present in unique indexes can be partitioned between those required for uniqueness and those columns included to obtain index only access to the table for certain queries by using the INCLUDE clause.

After the physical database design is complete, the execution behavior can be controlled both at the database manager (all databases in an instance) and the database level using a combination of database manager configuration parameters, database configuration parameters and registry settings. All of these are documented in the Administration Guide and allow you to override many of the default behaviors of DB2 Universal Database. In many cases DB2 offers a hierarchy of ways in which the behavior can be controlled. One example is the query optimization class which controls the intensity and scope with which the query optimizer explores the search space and considers different query rewrite and access path strategies. Prior to the execution of any individual DML statement the query optimization class can be specified using the SET CURRENT QUERY OPTIMIZATION statement. However, session level defaults for ODBC/CLI users can be set in the .ini file. Finally, a global database level default can be provided with the default query optimization class (dft_queryopt) database configuration parameter.

SQL statements come in different shapes and sizes, and sometimes contain idiosyncracies. For example, some statements are aggregations which return a small amount of data. Other queries return large volumes of data, but only the first few results are actually needed. Some queries are read-only, and others will be updated using WHERE CURRENT OF CURSOR processing. The optimizer can exploit this additional knowledge if the intent is signalled within the syntax of th SQL Query. For example, defining a cursor as FOR READ ONLY allows the optimizer to select access plans that do not have the ability to keep position on the base row. Similarly the FETCH FIRST N ROWS ONLY clause permits the optimizer to optimize sorts and result set processing for a specified and limited number of rows. Similarly the OPTIMIZE FOR N ROWS clause indicates that priority should be give to retrieving the first N rows as opposed to the entire answer set.

Finally, an important but often neglected area of performance tuning is the actual application design. There are many interfaces available for sending SQL statements to the database engine, ranging from static embedded SQL, through ODBC and JDBC, all the way to command interfaces like the command line interface and db2batch.

For further information on the ways in which access to DB2 Universal Database can be optimized for each of these interfaces refer to the DB2 Application Programming Guide and Reference.

6 Conclusion

Tuning DB2 is not difficult. Features have been added to make the database administrator's job easier. For the sizing and configuration phases, there are rules which help to provide a good starting point. Configuring and building the database is also made easier with the presence of both hidden auto-tuning and visual SmartGuides. Both help the user reach optimal database performance earlier in the deployment cycle, on all sizes of machines, from a personal database on a workstation to an enterprise solution on a parallel cluster. DB2 UDB offers these advantages without compromizing its configurability and flexibility, which have allowed expert administrators to implement advanced database layout and configuration. The auto-management technology has benefits for all levels of administrators. It lets the database user concentrate on his original concern: the safe storage and fast processing of enterprise data.

References

[ASTR76] Astrahan, M., M. Blasgen, D. Chamberlin, K. Eswaran, J. Gray, P. Griffiths, W. King, R. Lorie, P. McJones, J. Mehl, G. Putzolu, I. Traiger, B. Wade, and V. Watson, System R: Relational approach to database management, ACM Transactions on Database Systems 1:2 (june 1976).

[CHAM76] "SEQUEL2: a unified approach to data definition, manipulation, and control," IBM J. Research and Development 20:6, pp 560-575.

[CODD70] "A Relational model for large shared data banks," Comm, ACM 13:6, pp 377-387.

[DB299a] DB2 IBM DB2 Universal Database Administration Guide, Design and Implementation (SC09-2839), IBM Corp, 1999.

[DB299b] DB2 IBM DB2 Universal Database Administration Guide, Performance (SC09-2840), IBM Corp, 1999.

[DB299c] DB2 IBM DB2 Universal Database Command Reference (SC09-2844), IBM Corp, 1999.

[DB299d] DB2 IBM DB2 Universal Database SQL Reference, Volume 1 and Volume 2, (SBOF-8923), IBM Corp, 1999.

[HCL+90] L.Haas, W.Chang, G.M. Lohman, J. McPherson, P.F. Wilms, G. Lapis, B. Lindsay, H. Pirahesh, M. Carey, and E. Shekita Starburst mid-flight: As the dust clears. IEEE Trasactions on Knowledge and Data Engineering, March 1990.

[LMH+85] G. M. Lohman, C. Mohan, L. M. Haas, D. Daniels, B. G. Lindsay, P. G. Selinger and P. F. Wilms. Query processing in R*. In Query Processing in Database Systems, (W. Kim, D. S. Reiner, and D., S. Batory, eds.), Springer-Verlag, 30-47, 1985.