Automatic Configuration for IBM® DB2 Universal Database™

Compressing years of performance tuning experience into seconds of execution

Eva Kwan (evakwan@ca.ibm.com)
Sam Lightstone (light@ca.ibm.com)
Adam Storm (ajstorm@ca.ibm.com)
Leanne Wu (leannewu@ca.ibm.com)

January 31, 2002

Abstract

This paper describes the DB2® Performance Configuration Wizard, an expert tool for the configuration of DB2 Universal Databases™. This utility has shown dramatic results for tuning and configuring DB2 servers on UNIX® and Windows® platforms particularly on OLTP (transactional) systems. The recognition of the essential need for administration and design tools has spurred renewed interest among leading RDBMS vendors. The recent proliferation of papers on index and materialized view (summary table) selection, and the development of industrial applications in this regard by leading vendors, such as Microsoft®, IBM® and Oracle®, as well as numerous 3rd party administration tools vendors, are all testament to the growing corporate recognition of this important area of investigation. The DB2 Performance Configuration Wizard is a key feature in DB2's self-managing technology portfolio. This paper discusses the purpose and features of this expert tool. Experimental results are presented along with some of the interface specifics.

Introduction and Motivation

In this paper we present an overview and experimental results for a database configuration tool for DB2 Universal Databases, known as the DB2 Performance Configuration Wizard. This "advisor" makes recommendations on database configuration parameters and memory allotment within the database management system. The performance impact of a tuned configuration versus an untuned configuration may be dramatic, with measurable and significant performance improvement. The DB2 Performance Configuration Wizard is just one of a growing set of features for DB2 that reduce the human expertise required in database tuning and administration by combining automation, artificial intelligence, and expert advice. The experimental results shown illustrate how the use of this technology can greatly simplify expert tuning work.

Database vendors like IBM DB2 are aware that the human cost in operating large database systems can grow dramatically as the database sizes and the complexity of the hardware they require grow. Understanding the priorities of administrators is critical in reducing manual administration. As the scope of functions for relational databases has expanded in recent years, the complexity of database systems has grown accordingly. The increased complexity, and volume of data (now frequently comprising tens of Terabytes per database) have increased the burden of database administrators. Often the design choices for databases are complex and nontrivial. Some of the design issues that database designers grapple with include: the choice of hardware platform; the decision to use a shared-nothing, shared-everything or SMP-cluster hardware topology; the schema design; constraints and RI design; choice of primary key and indexes; the design of summary tables; and clustering models. Once a database has a physical and logical design, the operation of the database requires substantial human attention for numerous tasks including, but not limited to: table reorganization, data statistics collection, backup control, security modeling and user administration, disaster recovery planning, performance tuning, and problem analysis.

In recent years, there have been several research and industry attempts directed at tackling this area[1][4-12][14-18]. The goal is to create intelligent software tools that will reduce the burden on database administrators by providing expert design systems, performance tuning and configuration technology, easier to use interfaces for administration, and tools for automation. A number of early research projects focussed on the selection and design of table indexes and clustering, and some initial work on optimizing memory (specifically, buffer pool) allocation.

More recent projects have examined summary table design, statistics and reorganization prioritization, and constraint modeling. However, these initial areas of interest have focused on a very small subset of the larger problem. Generally speaking, there is relative dearth of research in what has become one of the most compelling areas of industrial application in Relational Database Management Systems (RDBMSs) – algorithms for the enablement of self-designing, self-administering, and self-tuning RDBMSs. In fact, as the number of features and the complexity of RDBMSs has increased, the human cost of ownership has increased due to increased skill requirements for database administrators, and the growing salary demands of skilled database administrators in North America[2].

A Total Cost of Ownership (TCO) study by D.H. Brown compared IBM's DB2 Universal Database product to Oracle Corporation's Oracle 8i database[3]. This study examined database applications by classification, separating database warehouses from online transaction processing applications, etc. While the human administration costs in TCO varied by product and application class, they clearly represented a large component of TCO in all cases for all users. Figures 1 and 2 show the DH Brown assessment of TCO for Datawarehousing (DW) and online transaction processing (OLTP) databases respectively. The DH Brown survey indicates substantially lower TCO for DB2 UDB over Oracle, making DB2 a less expensive choice. More significantly, note that for both products, the high cost to build and maintain both decision support and transaction processing environments appears to be dominated by human costs.

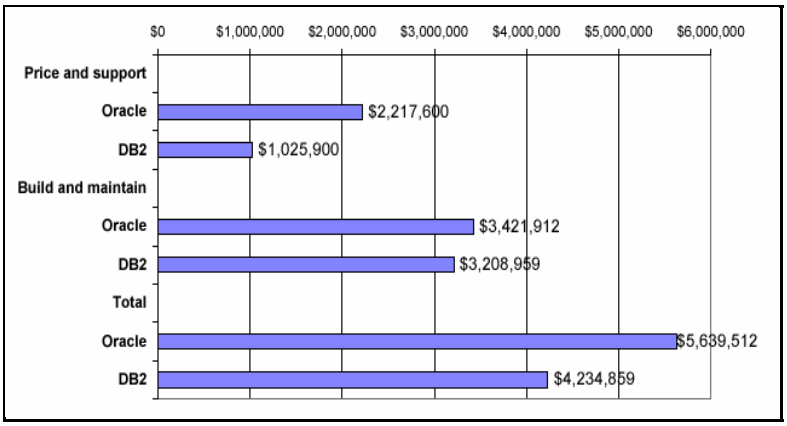

Figure 1 DH Brown study on Datawarehousing TCO.

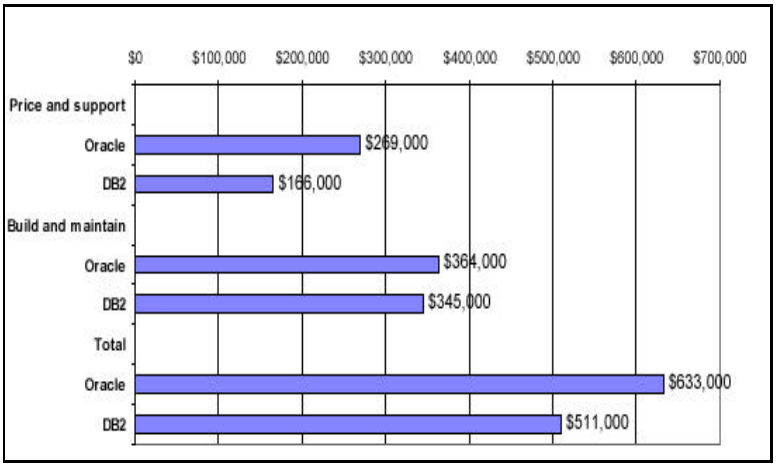

Figure 2. DH Brown study on OLTP TCO.

Recognition of the importance of ease of administration and of better design tools has spurred renewed interest in research and development of software to reduce the administration burden. The recent proliferation of papers on index and materialized view (summary table) selection, and the development of industrial applications in this regard by leading RDBMS vendors such as IBM DB2, Microsoft, and Oracle, and tools vendors such as Quest, Precise, BMC, DGI, and Computer Associates, are all testament to growing corporate recognition.

Clearly, as data sizes continue to grow, increasing the demand for large complex systems with more CPUs and more disks and disk arrays, the need for simplified administration will grow as well. The Asilomar Report on Database Research [2] projects that the growth in relational data and the growth in unstructured data stored in relational data servers will continue to grow for the next several years.

As the size and complexity of relational databases grow, the complexity in tuning these systems grows as well. This motivates the development of internal tuning and configuring technology within the product deliverable. The DB2 UDB Performance Configuration Wizard allows large database servers to be configured for performance in seconds. This utility automatically adjusts its calculations based on silent detection of system resources like CPUs, disks, and memory. While the utility has been available with DB2 for a few years, significant improvements have been performed for DB2 Universal Database version 8.1. In the following section, we provide an introduction to the utility and experimental results using the Configuration Wizard for tuning an industry standard benchmark.

This work is part of IBM's larger self-managing technology effort known publicly as "eLiza™"[13]. eLiza aims to enable the IBM portfolio of server, storage and middleware products with self-managing features in order to create autonomic computing systems. The eLiza architecture is designed around four focal areas: self-configuring, self-healing, self-optimizing and self-protecting. Based on the eLiza focal areas, the DB2 Configuration Wizard falls squarely under the "self-configuring" classification.

Interfaces

The Performance Configuration Wizard is part of DB2 Universal Database (for UNIX and Windows)[16]. Users can access the wizard through two interfaces. The first is a graphical interface available through the DB2 Control Center, and the second interface is a functional API, db2AutoConfigure(). The graphical interface allows administrators to configure databases through a point and click paradigm, while the API allows users to embed the self-configuring technology into their database applications.

The graphical interface begins by asking the user a set of simple plain-language questions regarding how the database system will be used. In general these questions require little expertise, and relate exclusively to information that is not readily available on the database server itself (such as the approximate number of users who will typically connect to the database in the future).

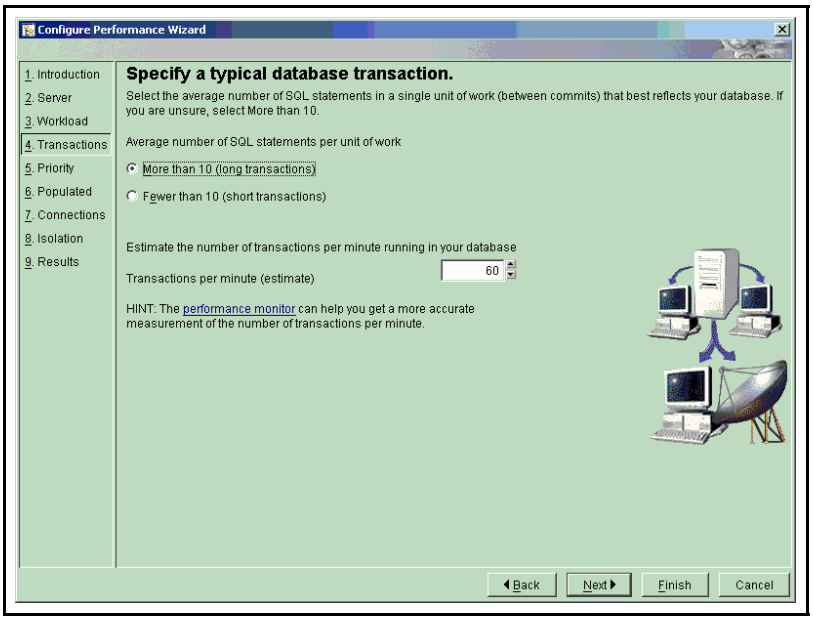

Figure 3. Performance Configuration Wizard graphical interface.

The interactive portion of the wizard gathers the following information:

- Database name
- Percentage of server memory to use for the database
- Workload type (online transaction processing, decision support, or mixed)
- Average number of statements per unit of work
- Estimated number of transactions per minute
- Whether the database administration prioritizes faster transaction performance, faster database recovery, or whether there is no priority at all
- Whether the database is currently populated with data
- Average number of connected local applications
- Average number of connected remote applications
- Isolation level (row-locking) requirements (repeatable read, read stability, cursor stability, or uncommitted read)

After stepping the user through these simple questions, the wizard combines the user's responses with information it automatically detects about the database server (described in more detail below). The complete characteristic set is combined into a configuration model inside the wizard, which biases the recommendations for setting configuration parameters and memory distribution within the DBMS. These recommendations are then presented to the user, who may opt to apply them. The modeling effort performed by the wizard for these recommendations typically requires less then ten seconds.

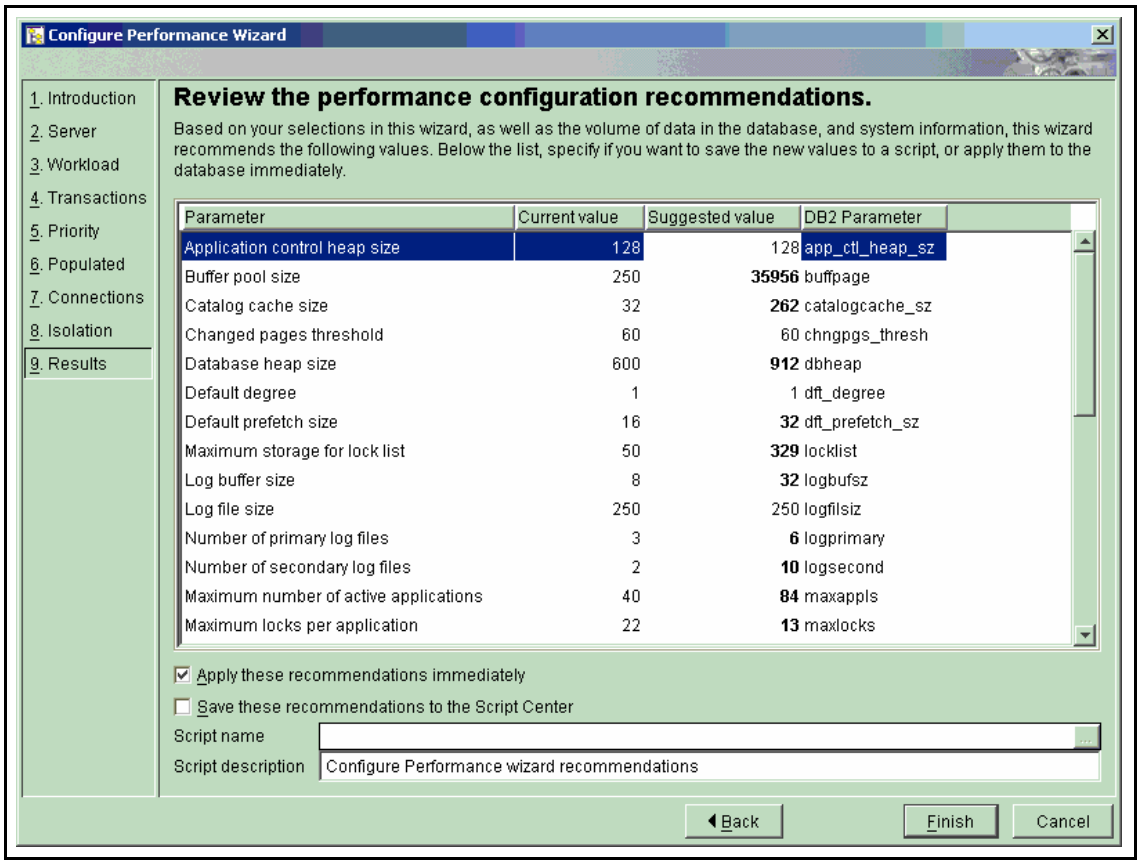

Figure 4. Performance Configuration Wizard results.

Internal Objectives and Features

The objective of this wizard is to define the database configuration parameters and memory assignments to optimize system performance. The complexity of the memory topologies for enterprise database management systems (including non-IBM solutions) makes these assignments a difficult task, even for skilled administrators deeply aware of their workload characteristics.

Similarly the complex process models deployed by enterprise server software in order to support hundred or thousands of attached users with efficient I/O and networking demands substantial complexity. Immature systems do a poor job of not only configuring parameters to support these process models, but many non-IBM products fail to provide sufficient means for monitoring and control to allow users to adapt configuration elements as required by their specific system use.

IBM's DB2 Universal Database product provides a set of configuration parameters that can be used to configure and tune the DB2 environment. The following configuration parameters specific to DB2 are recommended and set by the DB2 Performance Configuration Wizard. The parameters control memory distribution, parallelism, I/0 optimization and many aspects of logging and recovery:

- APP_CTL_HEAP_SZ
- BUFFPAGE
- CATALOG_CACHE_SZ
- DBHEAP
- LOCKLIST
- LOGBUFSZ
- LOGFILSIZ
- PCKCACHESZ
- SORTHEAP
- STMTHEAP
- UTIL_HEAP_SZ
- BUFFER POOL
- SHEAPTHRES
- CHNGPGS_THRESH
- DFT_DEGREE
- LOGPRIMARY
- LOGSECOND
- MAXAPPLS
- MAXLOCKS
- MINCOMMIT
- NUM_IOCLEANERS
- NUM_IOSERVERS
- SOFTMAX
- FCM_NUM_BUFFERS
- FCM_NUM_RQB
- INTRA_PARALLEL
- MAX_QUERYDEGREE
- MAXAGENTS
- NUM_POOLAGENTS
- NUM_INITAGENTS

To complete the system characteristics model used by the DB2 Performance Configuration Wizard, the utility automatically detects a number of system characteristics, including the following:

- System Information
  - Number of physical disks
  - Physical memory size (RAM)
  - CPU information
    - number of online CPU
    - number of configured CPU
  - Operating System features
    - Operating System (OS/2®, Windows NT®, Windows 2000, UNIX, etc.)
- Database Information
  - Size of database
  - Number of tables
  - Number of indexes
  - Number of tablespaces
- Bufferpool Information
  - Name, size for each bufferpool
  - Number of bufferpools

These automatically detected characteristics combine with the user specified characteristics to bias the internal configuration model used by the advisor. The experimental results of this modeling are described in the following section.

Experimental Results

To test the efficacy of the latest enhancements to the Performance Configuration Wizard, we conducted rigorous tests in an OLTP benchmarking environment.

We conducted our tests using an industry standard OLTP benchmarking environment. The environment is meant to simulate transaction-dominated systems. In such systems databases are updated by customers or employees connected to the database. Each operation (transaction) is generally very short and the benchmark measures how many transactions can be processed within a given unit of time. The environment resembles a company that has a database to track their inventory.

The database can be modified in the following ways:

- New orders for items can be placed
- Payments for items can be recorded
- A batch of orders can be queued for delivery
- The status of an order can be checked
- The level of inventory can be queried

The test environment consisted of one IBM pSeries™ 640 Model B80 server with 4x375 MHz POWER3-II processors. The server had 8 GB of physical memory, 96x16 GB Serial Storage Architecture (SSA) disk drives and was running AIX® Version 4.3.3. The benchmark was configured to have 200 concurrent users and the total database size was 25 GB.

The benchmark was run with three separate configurations. Initially the benchmark was run with DB2's default set of configuration parameters to obtain a baseline result for later comparison. After baseline results were obtained, the benchmark was run with the parameters assigned by the Performance Configuration Wizard. To gauge the value of the wizard's assigned parameters, the benchmark was run a final time with configuration parameters that were chosen after hand tuning. Figure 5 shows the relative results of the three runs.

The default settings on all systems are meant to provide a configuration that can be utilized by all customers without any tuning. They can be used even for systems which are extremely memory constrained (such as a laptop environment). For this reason, the configuration cannot take advantage of the fact that on the test system there was 8 GB of main memory (since some environments will have much less memory). It is not surprising that while the benchmark is capable of running reasonably well with the default configuration, the performance is far from that of the hand tuned configuration.

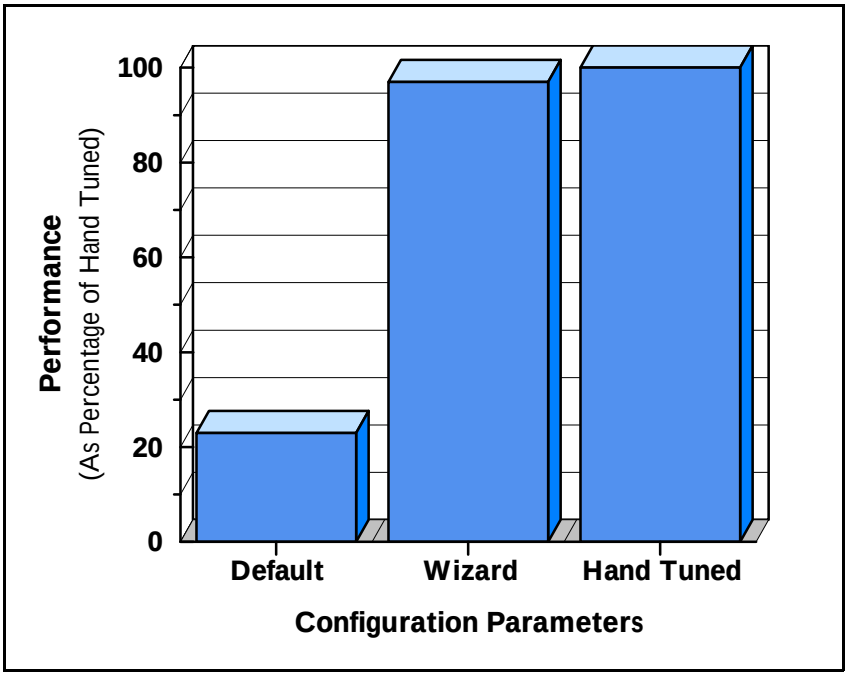

Figure 5. Performance configuration results.

With the configuration parameters recommended by the wizard the benchmark achieved performance just 3% away from the hand tuned configuration. This is remarkable given that

significantly less time was required to achieve this result. The ability to achieve excellent workload performance in a few minutes is preferable to the time consuming task of manually tuning configuration parameters.

Since the performance of the wizard's recommendation and the hand tuned configuration differ slightly, it is interesting to compare the configuration parameters that were chosen for the two runs. Table 1 contrasts the memory related parameters of the two runs.

	Wizard	Hand Tuned
Application Control Heap	128	128
Application Heap	128	328
Buffer Pool	252581	400000
Catalog Cache	596	64
Database Heap	4084	512
Lock List	4006	1000
Log Buffer	128	128
Package Cache	20000	1000
Sort Heap Threshold	12947	20000
Statement Heap	512	2048
Utility Heap	84193	5000
Total	**379303**	**430208**

Table 1. Memory related configuration parameters (in 4 KB pages**)**

Of the eleven memory related configuration parameters that were tuned, we can observe the following:

- In two cases, the configuration parameters were left at their default values (Application Control Heap. Log Buffer)
- In five cases, the parameters chosen by the wizard were larger (Catalog Cache, Database Heap, Lock List, Package Cache, and Utility Heap)
- In four cases, the parameters chosen by the performance tuning experts were larger (Application Heap, Buffer Pool, Sort Heap Threshold, and Statement Heap)

While the two configurations differed in nine cases, it is interesting to analyze the cases where the two configurations were significantly different. There are three configuration parameters that are vastly different: Package Cache, Utility Heap, and Buffer Pool.

In the case of the package cache, the wizard set the configuration parameter to 20000 pages while the hand tuned system was set to 1000 pages. It is easy to see why the wizard may have set this parameter higher than an expert who was hand tuning the system. This is because it is difficult for the wizard to determine the number of distinct queries that are to be run (and indeed, this is difficult to determine for any production system). The wizard makes a basic differentiation between the number of distinct queries (the user is asked whether the queries are OLTP, DSS, or

mixed). However, out of necessity, the granularity of this information is quite large. This is the desired behavior, since the wizard is designed so that its questions can be answered quickly and the recommendations can be made with as little disruption to the user as possible. With that in mind, the wizard's selection seems reasonable.

In the case of the utility heap being set larger by the wizard, we observe the difference between benchmarking situations and production systems. In production systems, utilities such as load, backup, and restore are run frequently and require large amounts of memory if the users wish them to run quickly. In a benchmark environment however, utilities are much less important. This highlights the fact that the wizard is tuning for a production environment while the expert who has hand tuned the system can tune for a single run in a relatively stable environment. For production systems, where utilities are important, it is clear that the wizard has made the right decision by increasing the amount of memory available for utilities.

The final difference in memory related configuration parameters is the size of the buffer pools. In this case the wizard has chosen a smaller buffer pool by 147419 pages. This results in saving over 500 MB of physical memory. It is interesting to note that the wizard has chosen a buffer pool configuration that is just 63% the size of the hand tuned configuration with nearly the same result. Figure 6 shows the difference between the buffer pool and total memory consumption for the two runs.

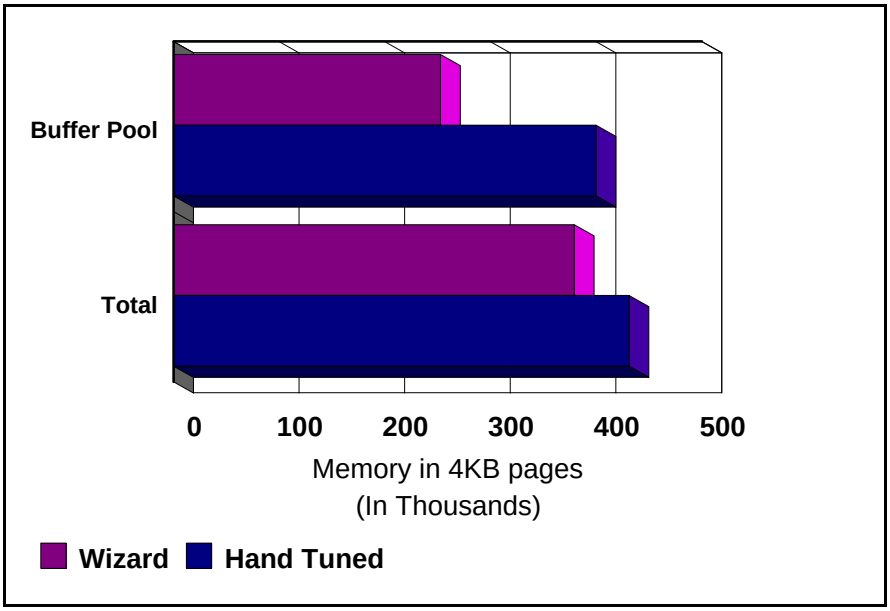

Figure 6. Buffer pool and total memory consumption in the two configurations

From the analysis of the configuration parameters selected by the wizard, it is apparent that the wizard has been designed to choose parameters that are suitable for production systems. While this is the case, it is remarkable that its selections perform so well in a benchmarking environment where hand tuning is crucial. This shows the power of the Performance Configuration Wizard as a tool for supplying the database administrator with a very good set of initial configuration parameters. If performance on the system is crucial, the administrator can then fine tune the system with additional knowledge of the workload characteristics.

Conclusions

The Performance Configuration Wizard was designed to help database vendors decrease the human costs involved in administering database management systems. With the increasingly large human component to the overall total cost of ownership of a database management system, the tool's value should not be understated.

The wizard has the ability to reduce and possibly eliminate the tedious and time consuming task of configuring a system for desired performance as is shown with the OLTP benchmarking results. If performance on the system is not crucial, the Performance Configuration Wizard eliminates the need for constant manual tuning of performance related configuration parameters. If performance is crucial, the wizard supplies a configuration that can serve as a springboard for further fine tuning based on specific workload characteristics. Best of all the tool takes just minutes to run and requires little knowledge of the many configuration parameters which it tunes. All these factors combined make the Performance Configuration Wizard a valuable tool for database administrators.

References

[1] E. Barucci, R. Pinzani, and R. Sprug-noli, "Optimal selection of secondary indexes"', IEEE Transactions on Software Engineering, January 1990, 16(1):32-38.

[2] P. Bernstein, M. Brodie, S. Ceri, D. DeWitt, M. Franklin, H. Garcia-Molina, J. Gray, J. Held, J. Hellerstein, H.V. Jagadish, M. Lesk, D. Maier, J. Naughton, H. Pirahesh, M. Stonebraker, and J.Ullman, "The Asilomar Report on Database Research", September, 1998B. A. Capara, M. Fischetti, D. Maio, "Exact and Approximate Algorithms for the Index Selection Problem in Physical Database Design", IEEE Transactions on Knowledge and Data Engineering, 7(6):955-967, December 1995.

[3] D. H. Brown Associates, "DB2 UDB vs. Oracle8i: Total Cost of Ownership", D. H. Brown Associates, Inc. December 2000.

[4] K. Brown, M. Mehta, M. Carey and M. Livny. "Towards Automated Performance Tuning For Complex Workloads", Proceedings of 20th International Conference on Very Large Databases, Santiago, Chile,1994.

[5] S. Chaudhuri, E. Christensen, G. Graefe, V. Narasayya and M. Zwilling. "Self-Tuning Technology in Microsoft SQL Server", IEEE Data Engineering Bulletin 22(2), June 1999, pp. 20-26.

[6] S. Chaudhuri and V. Narasayya, "AutoAdmin 'What-if' Index Analysis Utility", Procs. of the 1998 ACM SIGMOD Conference (Seattle, 1998), pp. 367-378.

[7] S. Chaudhuri and V. Narasayya, "Microsoft Index Tuning Wizard for SQL Server 7.0", Procs. of the 1998 ACM SIGMOD Conf. (Seattle, 1998), pp. 553-554.

[8] S. Choenni, H. M. Blanken, and T. Chang, "On the Selection of Secondary Indices in Relational Databases", Data & Knowledge Engineering, 11(3):207-233, 1993.

[9] J. Falkowski, "Comments on an Optimal Set of Indices for a Relational Database", IEEE Trans. on Software Engineering 18,2 (Feb. 1992), pp. 168-171.

[10] S. Finkelstein, M. Schkolnick, and P. Tiberio, "Physical Database Design for Relational Databases", ACM Transactions on Database Systems 13, 1 (March 1988), pp. 91-128.

[11] M. R. Frank, E. R. Omiecinski, and S. B. Navathe, "Adaptive and Automated Index Selection in RDBMS", International Conference on Extending Database Technology (EDBT), pages 277-292, Vienna, Austria, March 1992.

[12] H. Gupta, V. Harinarayan, A. Rajaraman, and J. D. Ullman, "Index Selection for OLAP", In Proceedings of the International Conference on Data Engineering, pp. 208-219, Birmingham, U.K., April 1997.

[13] IBM Server Group, “eLiza: Building an intelligent infrastructure for e-business”, online, http://www-1.ibm.com/servers/eserver/introducing/eliza/eliza_final.pdf
[14] M. Y. L. Ip, L. V. Saxton, and V. V. Raghavan, "On the Selection of an Optimal Set of Indexes", IEEE Transactions on Software Engineering, 9(2):135-143, March 1983.
[15] G. Lohman, G. Valentin, D. Zilio, M Zuliani, A Skelly, “DB2 Advisor: An optimizer smart enough to recommend its own indexes”, Proceedings of the 16th IEEE Conference on Data Engineering, February 2000.
[16] B. Schiefer and G. Valentin. “DB2 Universal Database Performance Tuning”, IEEE Data Engineering Bulletin 22(2), June 1999, pp. 12-19.
[17] G. Weikum, C. Hasse, A. Moenkeberg and P. Zabback. “The COMFORT Automatic Tuning Project”, Information Systems 19(5), 1994.
[18] K. Y. Whang, "Index Selection in Relational Databases", Proceedings of the International. Conference. on Foundations on Data Organization (FODO) (Kyoto, Japan), May 1985, pp. 369-378. Also reprinted in Foundations of Data Organization, Sakti P. Ghosh, Yahiko Kambayashi, and Katsumi Tanaka (eds.), Plenum Press (1987), pp. 487-500.

Special Notices

through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Trademarks

This article contains IBM and other company trademarks. IBM Trademarks information can be found at: http://www.ibm.com/legal/copytrade.shtml.

Microsoft, Windows, Windows NT, and Windows 2000 are trademarks of the Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other trademarks are the property of their respective owners.