Workload Models for Autonomic Database Management Systems

Pat Martin
School of Computing
Queen's University
Kingston, ON Canada K7L 3N6
martin@cs.queensu.ca

Said Elnaffar
College of IT
UAE University
Al-Ain, UAE
elnaffar@uaeu.ac.ae

Ted Wasserman
IBM Silicon Valley Laboratory
555 Bailey Avenue
San Jose, California USA 95141
tedwas@us.ibm.com

Abstract

Autonomic computing is a promising approach to the problem of effectively managing large complex software systems such as database management systems (DBMSs). In order to be self-managing, an autonomic DBMS (ADBMS) must understand key aspects of its workload, including composition, frequency patterns, intensity and resource requirements. It must therefore use and maintain different characterizations, or models, of the workload to support its various kinds of decision-making. Our research into various aspects of ADBMSs has led us to develop a number of different workload models. In this paper, we examine the importance of workload models to ADBMSs. We discuss the types of workload models needed by ADBMSs and describe examples from our research. We then outline the requirements for an infrastructure to develop and maintain the workload models needed by an ADBMS.

1. Introduction

As consumers demand more functionality and greater sophistication from Database Management Systems (DBMSs), vendors have been quick to deliver. However, the desire to include complex data types, the ability to store very large objects, and the emergence of diverse and varying workloads are factors that have led to unmanageable complexity. It is no longer feasible for database administrators to manually configure and tune these systems.

One approach to this management problem is an *autonomic DBMS* (ADBMS) that is capable of automatically managing its resources to maintain acceptable performance in the face of changing conditions [2][4]. An ADBMS must be able to perform typical configuration and tuning tasks including determining appropriate allocations for main memory areas such as the buffer pools and the sort heap, mapping database objects to buffer pools and adjusting the many DBMS configuration parameters to maintain acceptable performance.

In order to be able to manage its own performance, an ADBMS must be *self-aware*. A number of definitions of self-awareness exist in the literature [15]. For ADBMSs, self-awareness means that the system is able to formally model the consumption of its resources under the various demands caused by the workload components. Specifically, an ADBMS must be aware of several aspects of its workload, including composition, frequency patterns and intensity, and must maintain different models of the workload to support its various kinds of decision-making.

Our research in ADBMSs has led us to develop a number of different workload models. These models fall into two categories. The first category includes *exploratory* models, which provide a compact and summarized representation of the workload using unsupervised learning techniques. Discovering homogenous classes of database queries using clustering techniques is an example of exploratory model that can be used for tasks such as capacity planning. Exploratory models can also be useful for discovering the existence of periodic pattern in the workload. An autonomic DBMS can exploit such a pattern by predicting changes so that it can proactively tune its resources accordingly. The second category includes *confirmatory* models, which are used to ascertain that some significant property of the workload is occurring at some point in time using supervised learning techniques. Detecting interesting workload properties (e.g., identifying the dominant type of the workload) using confirmatory models (e.g., classification) help the ADBMS be reactively adaptive to on-line events that may threaten its performance or the security of its data.

In this paper we argue that there is a need for multiple workload models of different types in ADBMSs. We identify the types of models required in an ADBMS and then examine sample workload models of each type. We discuss the properties of these models and propose requirements for a framework to effectively create, maintain and use the models. The remainder of the paper is structured as follows. Section 2 reviews related work in the area of ADBMSs. Section 3 describes the workload models used in an ADBMS and outlines some example models.

0-7695-2653-5/06/$20.00 (c) 2006 IEEE

Section 4 discusses the problem of maintaining the workload models used by an ADBMS. Section 5 presents our conclusions.

2. Related work

An ADBMS system can be viewed as a feedback control loop as shown in Figure 1 [11], controlled by an Autonomic Manager. The autonomic manager oversees the monitoring of the DBMS (the Managed Element), and by analyzing the collected statistics in light of known policies and/or goals, it determines whether the performance is satisfactory. If necessary, a plan for reconfiguration is generated and executed.

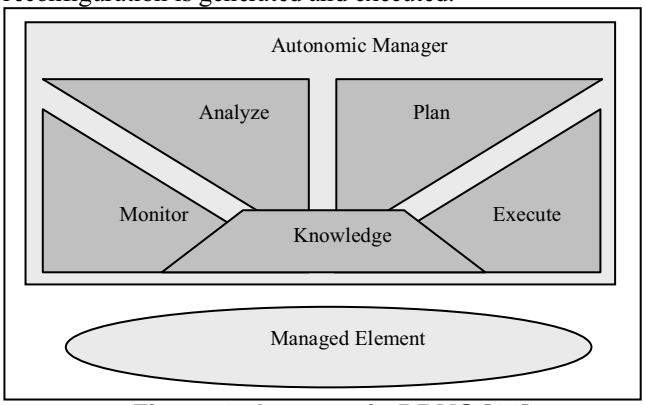

Figure 1. Autonomic DBMS [11]

ADBMSs, and autonomic systems in general, have received a great deal of attention both in the academic and the commercial worlds [2][4]. Self-tuning concepts have been applied to problems such as index selection [18], materialized view selection [1] and memory management [3][13]. Chaudhuri and Weikum [4] cite the need for self-tuning systems as an important reason to rethink current DBMS architectures.

3. Workload models and ADBMSs

Autonomic computing, as we explained earlier, is viewed as a promising approach to dealing with the increasing complexity of managing today's information systems. A system is considered to be autonomic if it possesses several capabilities, namely if it is self-configuring, self-optimizing, self-healing and self-protecting [11]. An understanding of the workload presented to a system is necessary to provide these capabilities.

A self-configuring ADBMS automatically adjusts to changes in its environment and to the set of components making up the system based on high-level policies. It must understand the composition and intensity of its workload in order to make the appropriate adjustments. For example, changes to the amount of resources available to an ADBMS from the underlying platform, or the removal of a component, may force the ADBMS to redistribute the load among its remaining components or to alter how individual requests are scheduled based on the requests' priorities and resource demands. A self-configuring ADBMS must have an exploratory model of the workload available to make its adjustments.

A self-optimizing ADBMS continually seeks to improve its performance or efficiency. It must understand the composition, intensity and priorities of the workload and be able to recognize shifts in the workload in order to effectively allocate resources. For example, to maintain acceptable levels of performance when the workload changes from mainly (On-Line Transaction Processing) OLTP requests to mainly Decision Support System (DSS) requests, an ADBMS must reallocate memory resources from the buffer area to the sort heap where joins are performed. A self-optimizing ADBMS would use confirmatory models to recognize workload shifts through value changes or trends in indicator properties. It may also use an exploratory model during the analysis and planning tasks (see Figure 1) to estimate the impact of possible changes on the system's performance.

A self-healing ADBMS automatically detects, diagnoses and repairs problems. It must understand the normal patterns and levels in the workload in order to detect possible sources of faults. For example, the system may tend to fail when the intensity of the workload exceeds certain levels. A confirmatory model of transaction arrival rates could be used to detect a high load that could cause failures. A self-healing ADBMS may also use an exploratory model during the analysis and planning tasks to predict the impact of potential repairs to the system.

A self-protecting ADBMS automatically defends itself against malicious attacks. It must therefore understand the normal patterns and levels in the workload in order to detect possible security threats from outside sources. The ADBMS could maintain confirmatory models for properties such as transaction arrival rates and connection requests that indicate possible denial of service attacks when the workload exceeds certain thresholds in the models.

3.1. Exploratory workload models

We find that exploratory workload models are required for the analysis and planning tasks of an ADBMS. The tasks typically incorporate a model of the system that is used to estimate the impact of possible plans. These system models use an exploratory model of the workload as input to the plan evaluation.

A sample or log of the actual workload of a system could be used as input to the system models but this is problematic for at least two reasons. First, in the case of a new system, the workload may not yet be known. Second, analysis and planning are performed on-line so using the real workload may make the analysis too complex to be practical.

Some types of exploratory models (e.g., time-series analysis) can reveal interesting patterns in the workload shifts of the ADBMS as explained by Elnaffar et al. [8]. Other exploratory models, such as clustering, can summarize the characteristics of resource demands in a small number of classes such as "light I/O queries" and "heavy CPU queries" [5][19][22].

Compact exploratory models can be constructed following a general methodology by Menascé et. al. [14] for constructing resource-oriented workload characterizations. The key steps in the process involve identifying appropriate characterization parameters that differentiate workload components and partitioning the workload into classes. Partitioning can be performed using clustering techniques from data mining.

In our work on sizing new database systems [20] we developed an exploratory model of DSS workloads. The ultimate goal of the sizing process is to arrive at a first estimate of a hardware configuration that will satisfy the performance demands, cost constraints, and functional requirements of an application. This can be a complicated task because of the wide variety of processor, disk, network and memory technologies that are available. Further, determining the quantity of each resource needed and predicting how the different components interact with each other under a specific workload are not trivial tasks.

In constructing our exploratory workload model for DSS workloads we used the 22 queries of the TPC-H benchmark [16], which was designed to simulate the type of resource activity commonly found in DSS applications, as our representative workload. The model represents the resource demands of different classes of queries typically found in DSS workloads. These demands are then used as input to determine an appropriate hardware configuration.

For our analysis, we monitored several performance-oriented parameters for each individual query, including response time, average CPU utilization, the average rate of random I/O, and the average rate of sequential I/O. The input data for our analysis were obtained from six pre-audited TPC-H benchmark runs on IBM's DB2 Universal Database (DB2) Version 8 [10].

Raw data were monitored and collected using standard operating system performance monitoring tools, which were configured to sample the desired parameters at five second intervals. We used a combination of Singular Value Decomposition (SVD) [9] and SemiDiscrete Decomposition (SDD) [12] to partition our workload. Both techniques are examples of unsupervised data mining, where the goal is to discover structured information in the dataset without knowing or providing hints as to what that structure might look like. SVD and SDD were jointly applied to our dataset by using both decompositions, truncating the SVD at $k = 3$, plotting the points corresponding to queries, and labeling each point according to its location in the top few levels of the SDD decomposition.

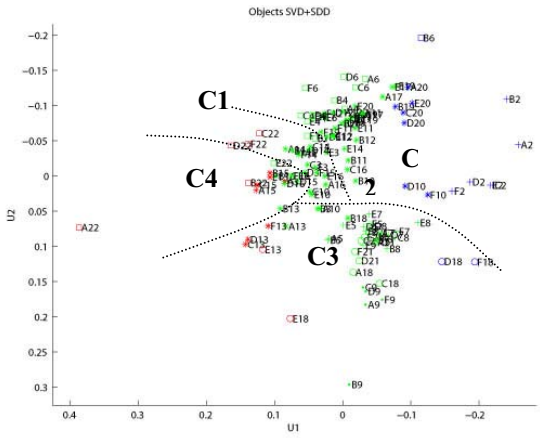

Figure 2. Clustering analysis of TPC-H data

The results of our analysis, which are summarized in Figure 2, identified four clusters. Cluster C2 represents simple queries that are generally IO-bound in nature and have a small number of tables being joined. Cluster C3 represents long-running, large and complex queries with a large number of tables (5+) being joined. Queries in this group also exhibit high sequential and random IO usage. Cluster C4 represents short-running, trivial complexity queries with a varying number of tables (3-8) being joined. Finally, Cluster C1 represents medium-complexity queries with a smaller number of tables (1-5) being joined and exhibiting high CPU utilization.

3.2. Confirmatory Workload Models

We can see from the previous discussion that confirmatory workload models are used in the analysis tasks performed by an ADBMS to detect the existence of specific conditions within the state of the ADBMS or its performance. Confirmatory models are often used in conjunction with one or more threshold values that trigger specific responses during the analysis tasks. Each confirmatory model focuses on either a single parameter or a small number of related parameters.

Our work on self-optimization has involved the development of several confirmatory workload models [7][21]. For example, the recognition of the workload type, specifically whether it is OLTP or DSS, is an important criterion for tuning because effective resource allocations to components such as the buffer area, sort heap and log buffer vary substantially depending on the type of the workload. A self-optimizing ADBMS could therefore detect shifts in the type of the workload and then reallocate resources on the fly in order to improve performance. The confirmatory model we built in this case is a decision-tree classifier that, for a given window of execution, assesses the degree to which the workload processed in the window is like an OLTP workload or like a DSS workload. Based on this assessment, appropriate tuning strategies can be adopted.

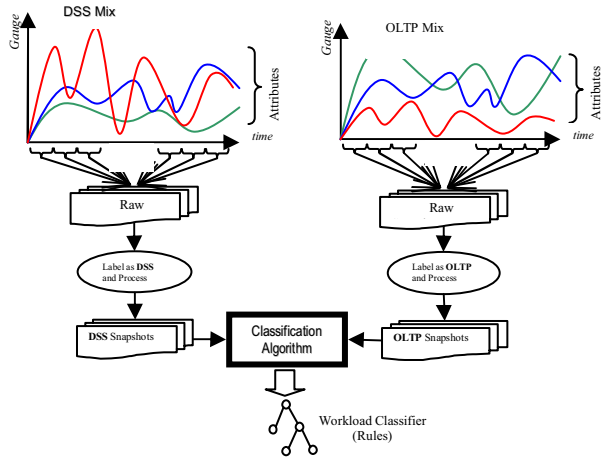

Figure 3. Classifying DBMS workloads

Our approach, which is shown in Figure 3, viewed the problem of classifying DBMS workloads as a machine-learning problem in which the DBMS must learn how to recognize the type of the workload mix. We chose to base our analysis on dynamic data collected by DBMS performance monitors because the dynamic data includes frequency information that is not available from static sources, it captures the variability of the workload over time, and it is easier to analyze than SQL statements or access plans. We first analyzed attributes available from performance snapshots with respect to their suitability for differentiating between DSS and OLTP and derived a set of attributes to be used in our classification process. We next built a model (or *classifier*) to describe a predetermined set of data classes. The model was constructed by analyzing a training set of data objects. Each object was described by attributes, including a *class label attribute* that identified the class of the object as either OLTP or DSS. The data objects needed to build the classifier are performance snapshots taken during the execution of the database workload. Each snapshot reflects the workload behavior at some time during the execution. The learned model was represented in the form of a decision tree embodying the rules that can be used to categorize future data objects.

We considered the following list of candidate attributes for the workload snapshots: *Ratio of Queries vs. Insert/Update/Delete statements, #Pages Read, #Rows Selected, Throughput, Number of Locks Held, Ratio of Using Indexes vs. Data Tables, Number of Sorts, Average Sort Time, Logging, Hit Ratio (%), and # Pages Scanned.*

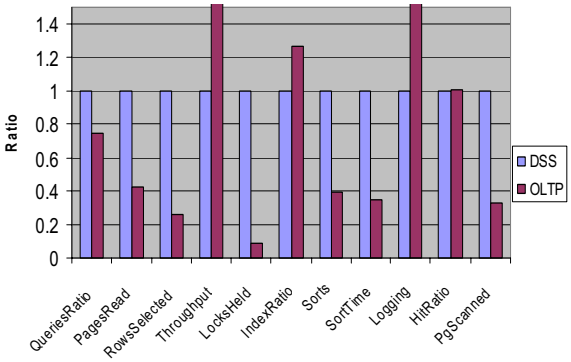

Figure 4. Candidate attributes for snapshots

We considered the *Browsing* and *Ordering* profiles defined in the TPC-W benchmark [17] as examples of DSS and OLTP workloads, respectively. Figure 4 shows the relative values, with the DSS values normalized to 1, for a set of candidate attributes. The values were derived from experiments with the TPC-W workloads on DB2 and all were easily obtained with the DB2 Snapshot Monitor. We eventually eliminated *Throughput* and *Hit Ratio* from the snapshot because *Throughput* was dependent on the current system utilization and *Hit Ratio* was strongly affected by the DBMS configuration, which can include buffer pool sizes and the assignment of database objects to these pools.

The remaining attributes were not equally system-independent. Therefore we grouped the attributes into three classes based on their degrees of system-dependence and assigned different weights to each class of attribute to reflect their significance to the classification process. *Queries Ratio, Pages Read, Rows Selected, Pages Scanned,* and *Logging* were the least sensitive to changes in the system configuration and so were the most significant to classification. *Number of Sorts* and *Ratio of Using Indexes* were somewhat vulnerable to configuration changes that were likely to occur infrequently, such as changing the current available set of indexes or views in the database schema. *Sort Time* and *Number of Locks Held* were the most sensitive to changes in the system configuration and hence were the least significant to classification. The classification model (decision tree) created for the TPC-W Browsing and Ordering profiles is shown in Figure 5.

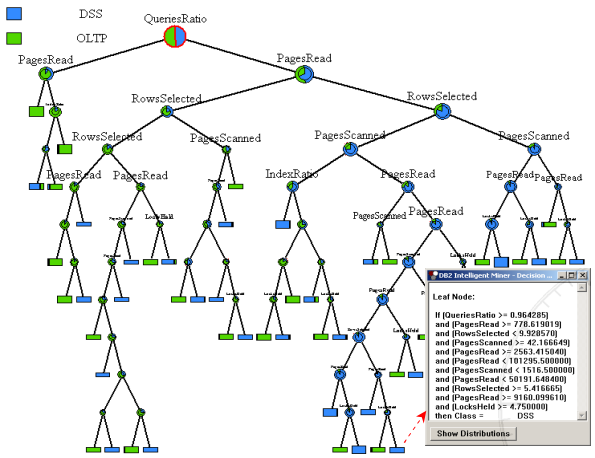

Figure 5: Workload classifier

4. Maintaining Workload Models

The effective creation and maintenance of workload models will be key to the viability of ADBMSs. Workload model maintenance involves the Knowledge Base, Monitoring and Analysis components of the autonomic manager (see Figure 1). The Knowledge Base stores descriptions of the models and performance data collected by the Monitoring component. The Analysis component includes statistical analysis, data mining and machine learning techniques, such as those described above, to create and update the workload models.

Workload models, like the workloads they represent, are not static and will have to change over time. The maintenance approach can be either on-line or off-line. An off-line approach will periodically regenerate the model off-line and then replace the current version of the model with the new version. The disadvantage of an off-line approach is that data collected from the Monitoring component to maintain the model will have to be stored by the ADBMS until it is needed. An on-line approach to maintaining the workload models will, on the other hand, perform incremental maintenance. It will therefore have smaller storage requirements than an off-line approach but runs the risk of more interference with the applications running on the ADBMS. On-line versions of popular clustering and classification algorithms are being studied in the field of data stream mining [6] and will be of great use in ADBMSs.

Monitoring is a double-edged sword for ADBMSs. On the one hand, maintaining the workload models can require continuous and comprehensive monitoring to provide up-to-date data. On the other hand, monitoring may have a negative impact on applications' performance and should be restricted. We therefore need more unobtrusive monitoring and selective techniques that make effective use of limited monitoring. Piggybacking is an example of an unobtrusive technique that provides monitoring on top of execution of query plans [23].

In our own research we have observed two general properties that have an impact on selective monitoring. First, we can identify specific patterns or cycles within a workload that can allow us to predict changes in a workload. Second, an autonomic manager typically makes management decisions in response to changes experienced by its managed element. We have found that changes in the workload are predictable and can be discovered by analyzing historical data [8]. We can therefore build a prediction model of the workload behaviour and perform selective monitoring guided by the prediction model.

5. Conclusions

Autonomic computing is a promising approach to managing complex systems like DBMSs. An autonomic DBMS, however, must understand several key aspects of its workload, including composition, frequency patterns and intensity, and must maintain different characterizations, or models, of the workload to support its various kinds of decision-making.

In this paper, we examine the different uses of workload models in an ADBMS and classify workload models as either exploratory or confirmatory. Exploratory models provide a compact representation of the workload as a whole. They are primarily used as input to prediction models in the analysis and planning tasks carried out by an ADBMS. Confirmatory models are typically used to detect certain conditions and so trigger specific actions in the analysis task. We describe examples of both types of models from our own research in ADBMSs.

We argue that techniques to effectively store and maintain workload models are critical to the viability of ADBMSs. We envision an infrastructure that builds upon the notion of an autonomic element (e.g., ADBMS) so that the workload models are stored and maintained by the autonomic manager. Several issues must be addressed in providing this infrastructure. The first issue is how to describe and store the workload models. We have developed particular methods for each model but a general solution is required. The second issue is how to provide effective and efficient monitoring of the managed element. An approach we are studying is to develop prediction models that highlight important events in the behaviour of the workload and so focus the use of monitoring around these events. The third issue is how to perform effective incremental maintenance of the models. We need methods by which recent data can be used to incrementally improve our models and thus avoid the need for storing potentially large amounts of data in order to periodically recreate the models from scratch. We believe that data stream mining is a promising source for these methods.

Acknowledgements

The authors thank other members of the Database Systems Research Lab, specifically Wendy Powley, Xilin Cui, Xiaoyi Xu, and Hamzeh Zawawy. They also thank David Skillicorn and their collaborators at IBM, specifically Berni Schiefer, Sam Lightstone, Haider Rizvi and Randy Horman. Finally, the authors gratefully acknowledge the support of IBM Canada Ltd., the National Science and Engineering Council of Canada (NSERC) and Communication and Information Technology Ontario (CITO).

References

[1] S. Agrawal, S. Chaudhuri and V. Narasayya. "Automated Selection of Materialized Views and Indexes," *Proc. of 26th Int. Conf. on Very Large Databases,* Cairo, Egypt, September 2000.

[2] P. Bernstein, M. Brodie and S.Ceri, et al., "The Asilomar Report on Database Research," *ACM SIGMOD Record,* Vol 27, No. 4, pp. 74-80*,* Dec. 1998.

[3] K. Brown, M. Carey and M. Livny, "Goal Oriented Buffer Management Revisited," *ACM SIGMOD Record,* Vol 25 No. 2, pp. 353 – 364, June 1996.

[4] S. Chaudhuri, G. Weikum, "Rethinking Database System Architecture: Towards a Self-Tuning RISC-Style Database System," *Proc. of 26th Int. Conf. on Very Large Databases*, Cairo, Egypt, pp 1-10, Sept. 2000.

[5] X. Cui, P. Martin and W. Powley, "A Study of Capacity Planning for DBMS and OLAP Workloads", *Proc. of the Computer Measurement Group's 2003 International Conference*, Dallas TX, December 2003.

[6] P. Domingos and G. Hulten. "Catching Up with the Data: Research Issues in mining Stream Data", *Proc. Of 2001 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Santa Barbara CA, May 2001.

[7] S. Elanffar, P. Martin and R. Horman, "Automatically Classifying DBMS Workloads" *Proc of 11th International Conference on Information and Knowledge Management (CIKM),* November 2002.

[8] S. Elnaffar and P. Martin, "An Intelligent Framework for Predicting Shifts in the Workloads of Autonomic Database Management Systems", *Proc of 2004 IEEE International Conference on Advances in Intelligent Systems – Theory and Applications*, Luxembourg, November 2004.

[9] G.H. Golub and C.F. van Loan, *Matrix Computations*, Johns Hopkins University Press, 3rd edition, 1996.

[10] IBM, *DB2 Universal Database*, http://www.software.ibm.com/data/db2/udb.

[11] J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing" *IEEE Computer*, Vol. 36, No. 1, pp. 41-50, Jan. 2003.

[12] T.G. Kolda and D.P. O'Leary. "Computation and uses of the semidiscrete matrix decomposition", *ACM Transactions on Information Processing,* 1999.

[13] P. Martin, W. Powley, M. Zheng and K. Romanufa. Experimental Study of a Self-Tuning Algorithm for DBMS Buffer Pools. *Journal of Database Management 16(2)*, pp. 1 – 20, 2005.

[14] D. Menascé, V.A.F. Almeida, and L. Dowdy. *Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems*, Prentice Hall, 1994

[15] M. Mowbray and A. Bronstein. "What kind of self-aware systems does the Grid need?", HP Laboratories Bristol, HPL-2002-266 (R.1), February, 2005.

[16] Transaction Processing Performance Council, *TPC Benchmark H Standard Specification,* Revision 2.1.0 http://www.tpc.org/tpch/spec/tpch2.1.0.pdf.

[17] Transaction Processing Performance Council*, TPC Benchmark W (Web Commerce) Standard Specification Revision 1.7*, October 2001.

[18] G. Valentin, M. Zuliani, D. Zilio, G. Lohman and A. Skelly. "DB2 Advisor: An Optimizer Smart Enough to Recommend Its Own Indexes," Proceedings of Int. Conf. on Data Engineering, San Diego, California, pp. 101-110, February 2000.

[19] T. Wasserman, P. Martin and H. Rizvi, "Sizing DB2 UDB Servers for Business Intelligence Workloads", *Proc. of IBM Centre for Advanced Studies Conference* (*CASCON 2004),* Toronto ON, pp. 135 – 149, October 2004.

[20] T. Wasserman, P. Martin, D. Skillicorn and H. Rizvi, "Using a Characterization of Business Intelligence Workloads for Sizing New Database Systems", *Proc of ACM Seventh International Workshop on Data Warehousing and OLAP (DOLAP 2004)*, Washington D.C., pp. 7 – 13, November 2004.

[21] X. Xu, P. Martin and W. Powley, "Configuring Buffer Pools DB2 UDB", *Proc. of IBM Centre for Advanced Studies Conference* (*CASCON 2002),* Toronto ON, pp. 171 – 182, October 2002.

[22] H. Zawawy, P. Martin and H. Hassanein, "Capacity Planning for DB2 UDB", *Proc. of IBM Centre for Advanced Studies Conference* (*CASCON 2002),* Toronto ON, pp. 89 – 97, October 2002.

[23] Q. Zhu, B. Dunkel, W. Lau, S. Chen and B. Schiefer, "Piggyback Statistics Collection for Query Optimization: Towards a Self-Maintaining Database Management System", *The Computer Journal*, Vol.47, No.2, pp. 221 – 243, 2004.