Journal of Advances in Science and Technology
Vol. II, Issue II, November-2011, ISSN 2230-9659

Development Study of self Managing databases and its latest achievements

B.CHEENA KESHAVULU
Research Scholar, Manav Bharti University, H.P., India

Abstract: The problem of self-tuning databases has been very important for the last 10 years. Commercial database systems have been adding features for a long time and have now reached a level of complexity that makes them a difficult choice as a central platform for information services or e-commerce. It is critical that database systems be easy to manage, predictable in their performance characteristics and, ultimately, self-tuning. Self-tuning database technology is bound to build on mathematical models and to provide proper engineering into system components.Today's DBMSs should be knob-free and must be able to adapt as conditions change, without human intervention. An autonomic DBMS should be capable of managing and maintaining itself as well. This paper examines the characteristics that a DBMS should have in order to be considered autonomic.Optimization of self-tuning models strives to minimize a programmable goal function that reflects the system designer's preferences and the system behavior.

-------------------------♦-------------------------

1. INTRODUCTION

This research focuses on the characteristics of databases that do not have database administrators and are considered self-tuning.
To be self-tuning there can be no parameters set by humans, and the system must be able to adapt as conditions change. The main investigation is on the problem of how to make a database "knob-free," meaning it has no user-specifiable parameters.

1.1 Definition of Autonomic DBMS

The combination of increased data volumes, large systems and more functions drive the need for autonomic database management systems. These databases operate in the environment with limited intervention by personnel, thus reducing costs. In general, autonomic DBMS are capable of managing and maintaining themselves, adjusting to various circumstances and preparing their resources to provide efficient handling of workloads.

These systems are designed in such a way that users can concentrate on anticipation and accomplishment of their needs. In this paper, we examine the characteristics that a DBMS should have in order to be considered autonomic.
The self-managing features of autonomic technologies in DB2, Oracle and SQL Server illustrate how self-managing technology can reduce complexity, helping to reduce the total cost of ownership of DBMSs and improve system performance.

1.2 Self-tuning Applications

1.2.1 RISC-style Components

The advances made in self-tuning database technology during the last ten years, have been based on the paradigm of a feedback control loop. Further advances are bound to build on mathematical models and their proper engineering into system components. In addition, the transformation of information services into truly self-tuning, higher-level E-services may require a conversion of highly componentized software (RISC-style components) into simpler architectures with narrow interfaces.

It is critical that database systems should be easy to manage, predictable in their performance characteristics, and ultimately self-tuning. To reach this elusive goal, RISC-style simplification of server functionality and interfaces is absolutely crucial.

Database technology is packaged into small RISC-style data managers with lean, specialized APIs, and with built-in self assessment and auto-tuning capabilities.
This problem of simple models leads us to the tuning and administering of large installations.

1.2.2 Active Databases

Active databases systems detect events and trigger actions as a result. Active capabilities are provided by a set of rules, with each rule consisting of three components: event, condition, and action.
A major performance issue in active databases is the issue of relationships among rule components. Current implementations of triggers do not allow flexibility in the

selection of transaction policies (partition of rules to transactions), the inter-transaction timing policies of rules' components, the inter-transaction policies of commit and abort dependencies, and synchronization issues.
The self-tuning model consists of the optimization model, which strives to minimize a programmable goal function that reflects the system designer's preferences and the system behavior and the application's semantics through constraint definitions. The tuning model strives to optimize the mutual relationships among the system rules' components.

The rest of the paper is organized as follows:

Section 2 examines and discusses the characteristics that a DBMS should have to be considered autonomic. This section also outlines the main features of an autonomic database management system.

Section 3 is a set of case studies, which compare autonomic characteristics in Oracle, DB2 and MS SQL Server databases.

Section 4 discusses auto-tuning principles and the challenges of large databases. We also present RISC-style architecture, which involves detailed examination of the self-tuning, RISC-style components of a DBMS.
We evaluate the performance of active databases models. Two types of coupling modes or decision packages are shown to exemplify the self-tuning process.
We discuss attempts to adjust individual knobs in automated tuning and research a methodology that uses a probabilistic, graphical model, known as an influence diagram, as the foundation of an effective, automated approach to software tuning.

Section 5, we conclude with advances in different database areas, which are trying to simplify models.

2. CHARACTERISTICS OF ADBMS

Database management systems (DBMSs) are a vital component of many mission-critical information systems and, as such, must provide high performance, high availability, excellent reliability and strong security. The Data Base Administrators (DBAs) who manage these systems must be knowledgeable in areas such as capacity planning, physical database design, systems tuning and systems management.

Autonomic Database Management Systems (ADBMS) are a desirable long-term research goal.
What are the properties of an autonomic computing system? To answer that question, let us first discuss the autonomic characteristics of DBMS systems [1].

2.1 Self-optimizing

Self-optimization is one of the most challenging features in a DBMS. It allows a DBMS to perform any task and execute any service utility most efficiently, given the workload parameters, available resources, and environment settings. The most important task for optimization is the execution of a query and that is one of the most apparent autonomic features of today's DBMSs. Usually query optimization involves query translation, the generation of a cost-efficient execution plan, and dynamic, run-time optimizations.

2.2 Self-configuring

The performance of a DBMS depends on the configuration of its hardware and software components. An autonomic DBMS should:

- Provide users with reasonable "out of the box" performance and dynamically adapt its configuration to provide acceptable, if not optimal, performance under constantly changing conditions.
- Recognize changes in its environment that warrant reconfiguration.
- Reconfigure itself without severely disrupting online operations.
- Provide support for determining the optimal set of indexes and materialized views to be used by the query optimizer.

The ADBMS configuration includes performance parameters, resource consumption thresholds, and the existence of auxiliary data structures, such as indexes and materialized views, in the database schema. DBMSs provide configuration wizards such as DB2's Configuration Advisor. Configuration advisors are tools to assist with initial configuration, but the settings are usually static.
The goal of an autonomic DBMS is to provide dynamic adjustment of these settings. Little support is currently provided for this type of self-configuration.

2.3 Self-healing

Self-healing, a fundamental requirement of a DBMS, means that the database remains in, or can be restored to, a consistent state at all times. A DBMS must reliably log all operations, periodically archive the database, and use the logs and backups to recover from failure.
Ideally, an ADBMS should recognize when a full or incremental backup is necessary and perform these operations with minimal system disruption. In case of catastrophic failure, an ADBMS should be able to retrieve the most recent backup, restore to the consistent point just before the failure, and then resume its halted operations after handling the exceptions.

2.4 Self-protecting

There are five features of self-protection: database security, privacy, analytical auditing mechanisms, data encryption, and admission control strategies. These features shield the DBMS from potential, errant requests that may deteriorate its performance or bring the DBMS down. All multi-user DBMSs provide authentication mechanisms that prevent unauthorized users from accessing the database. Database privacy ensures that users are granted access only to the portions of the database that are required.

2.5 Self-organizing

An ADBMS should be capable of dynamically reorganizing and re-structuring the layout of data stored in databases (e.g., tables), associated auxiliary data structures (e.g., indexes), and any system-related data (e.g., system catalog) to optimize performance.

2.6 Self-inspecting

An ADBMS should "know itself" in order to make intelligent decisions about all the autonomic features discussed. It must collect, store and analyze relevant information about its components, performance, and workload. This information should be used in optimizing performance, detecting any potential problems, updating statistics about the stored data, ensuring integrity of data read from disk, scheduling maintenance utilities, and identifying interesting trends in the workload.

3. CASE STUDIES

It is useful to evaluate current DBMSs in light of the properties of autonomic computing systems. Then we can judge what has been accomplished to date and what problems left to be solved. This section outlines where DBMSs are today in their autonomic capabilities. [1]

3.1 Oracle

Self-optimizing
Optimization models in Oracle automatically determine the appropriate amount of optimization on a per-query basis. During query execution, cost models will be able to benefit from the self validation of the cardinality model proposed by DB2's Learning Optimizer. Dynamic adjustment to the query execution strategy in Oracle consists of automatic memory allocation so that each query has the appropriate amount of memory. DB2 and Oracle both provide an automatic mechanism for query parallelism.

Self-configuring
Oracle provides automatic memory management. These systems allocate memory as needed by the database, limiting memory allocation when either a user-imposed limit is reached or the system's physical resources run low.

Self-healing
Oracle can resume operations (such as a batch load) following corrective action (such as the addition of more disk space). Oracle allows the DBA to set a recovery interval parameter that specifies a target for recovery time in seconds. All DBMSs support logging, backup and recovery mechanisms.

Self-organizing
Current DBMSs do not assist in the initial layout of data on disks and do not shift data from one disk to another to even out disk demands. However, Oracle does provide the ability to move tables while on-line.

Self-inspecting
Using the DB2 Health Center or the Oracle Manager Console, a DBA can examine the system for signs of unhealthiness and store performance data in a data warehouse.

3.2 DB2

Self-optimizing
The DB2 optimizer allows the user to adjust the amount of optimization. In addition to query optimization, a DBMS must also optimize utilities such as backup, restore, statistics collection and data load utilities. DB2's Load utility performs mass insertions of data into a target table by exploiting a series of parallel I/O sub-agents for pre-fetching, SMP parallelism degree, and the amount of memory available for buffering and sorting.

Self-healing
DB2 has a recovery tool, the Recovery Expert, which analyzes the recovery assets available and recommends a technique to be selected. DB2's Automatic Incremental Restore mechanism uses the backup history for automatically searching for the correct backup images.

Self-protecting
DB2 and SQL Server provide security on a per table basis, whereas Oracle provides row-level security. Admission and application control is essential for ADBMSs to protect the system from database requests that may deteriorate performance and/or undesirably consume system resources.

Self-organizing
To make efficient use of system resources, DB2 permits dynamic online index-reorganization to reclaim leaf-level storage.

Self-inspecting
Analysis tools such as The DB2 Performance Expert can analyze performance data. The Maintenance Advisor

examines DB2 statistics and makes recommendations on what maintenance utilities should be run. Sector Consistency Checking for page I/Os ensures the integrity of read data by detecting any corruptions caused, for example, by incomplete I/Os.

4. CONCLUSIONS

Despite the many advances that have been made towards autonomic database management systems, much work remains to reduce the amount of human intervention required by these systems.

There is still plenty of space in the field of transition from manual to automatic managing of the systems. Even though current DBMS provide many tools for initial configuration, system monitoring and problem analysis, in most cases, these tasks still require a significant amount of input, intelligence and decision making from the DBA.

The settings, in most cases, do not automatically adapt to changes in the system environment or workload, so dynamic adaptation is needed.

The lack of ability to reset DBMS parameters online will motivate greater research unless intelligent strategies will enable autonomic features. Rules of thumb are not adequate for difficult configuration tasks. More robust analytical models and prediction mechanisms are required.

4.1 Focus for Further Research

ADBMS research should focus in four main areas:

1. Developing a proper infrastructure to allow the clean introduction of autonomic computing system features.
2. Developing intelligent decision-making and prediction tools based on feedback control loops and tractable mathematical models.
3. Exploring and exploiting significant characteristics and trends in the DBMSs workload using statistical and data mining techniques.
4. Developing a useful model of the system that explains the relationships among the numerous components of a DBMS.

4.2 The Future of DBA’s

Progressing towards ADBMSs will not mean the demise of DBAs. It will mean the end of repetitive administrative tasks, freeing DBAs to spend more time on new applications and on business policies and strategies. Furthermore, DBAs will be needed to evaluate and select recommendations before they are implemented.

Autonomic DBMS must have the ability to predict when is the best time to schedule the execution of utilities that can be provided by smart maintenance strategies.

ADBMSs should provide auditing mechanisms where logs are used to track all DBMS activity. Researchers should be encouraged to make the system architecture of database technology and the simplification of component interfaces a top-priority item.

Lean APIs need to be worked out for each of the most important RISC-style components on different kinds of query processors and storage managers.

To make a real challenge, the rules should limit the code size of each component, limit its footprint, and disallow any kinds of tuning knobs other than what the component does internally based on its own self-assessment.

In an ideal, simple and cost-efficient system, the applications would coordinate internally and with each other. System administrators would not need to set knobs to control memory usage on either the high-end or the low-end. Again, this broader goal comes from the fact that database software will be interacting with other end-user and system software.

4.3 Lessons Learned

Even if some tuning decisions, such as re-configuring an entire system are not completely automated, it is of great value to have observation and prediction components in the spirit of self-tuning. Alerting a system administrator about increasing load or significant changes in workload patterns as early as possible can often save days of unacceptable performance degradation.

In addition, automatically narrowing down the possible bottlenecks and recommending appropriate remedies can drastically simplify the administrator’s job.

Universal database systems grew out of the belief that database is the center of the universe and therefore the framework for integration. This is far from true in today’s world. Once we are liberated and can accept the fact that the database is a very important, component in the IT world, programmability and integration of database components with applications becomes a priority. In such a world, we need to build RISC-style data management components that have predictable performance and can be auto-tuned

5. REFERENCES

[1] Elnaffar, S.; Powley, W.; Benoit, D.; Martin, P. Today's DBMSs: how autonomic are they. Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA’03), 1-5 Sept. 2003 IEEE, Pages: 651 – 655

[2] Weikum, G.; Moenkeberg, A.; Hasse, C. Zabback. Selftuning database technology and information services:

from wishful thinking to viable engineering. Proceedings of the 28th International Conference on Very Large Data Bases, 20-23 Aug. 2002, Hong Kong, China.

[3] Chaudhuri, S.; Weikum, G. Rethinking Database System Architecture: Towards a Self-tuning RISC-style Database System. Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 2000

[4] Peter Spiro. Ubiquitous, Self-tuning, Scalable Servers. SIGMOD '98 Seattle, WA, USA 1996 ACM

[5] Yingping, H.; Xiaorong, X.; Madey, G. A self-manageable infrastructure for supporting web-based simulations. Proceedings of the 37th Annual Simulation Symposium (ANSS'04), IEEE 18-22 April 2004

[6] Lightstone, S.; Lohman, G.; Zilio, D. Toward Autonomic Computing with DB2 Universal Database. ACM SIGMOD Record, Volume 31 Issue 3, September 2002.

[7] Sullivan, D.; Seltzer, M.; Pfeffer, A. Using Probabilistic Reasoning to Automate Software Tuning. SIGMETRICS/Performance'04, June 12-16, 2004, New York, NY, USA. ACM 1-58113-873-3/04/0006.

[8] David Botzer and Opher Etzion, IEEE Computer. Society. Self-Tuning of the Relationships among Rules' Components in Active Databases Systems. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 16, NO. 3, MARCH 2004.

[9] Scheuermann, P.; Weikum, G.; Zabback, P. Data partitioning and load balancing in parallel disk systems. The VLDB Journal (1998) 7: 48–66