Experimental Study of a Self-Tuning Algorithm for DBMS Buffer Pools

Patrick Martin, Queen’s University, Canada
Wendy Powley, Queen’s University, Canada
Min Zheng, Queen’s University, Canada
Keri Romanufa, IBM Toronto Laboratory, Canada

ABSTRACT

The tasks of configuring and tuning large database management systems (DBMSs) have always been both complex and time-consuming. They require knowledge of the characteristics of the system, the data, and the workload, and of the interrelationships between them. The increasing diversity of the data and the workloads handled by today’s systems is making manual tuning by database administrators almost impossible. Self-tuning DBMSs, which dynamically reallocate resources in response to changes in their workload in order to maintain predefined levels of performance, are one approach to handling the tuning problem. In this paper, we apply self-tuning technology to managing the buffer pools, which are a key resource in a DBMS. Tuning the size of the buffer pools to a workload is crucial to achieving good performance. We describe a Buffer Pool Tuning Wizard that can be used by database administrators to determine effective buffer pool sizes. The wizard is based on a self-tuning algorithm called the Dynamic Reconfiguration algorithm (DRF), which uses the principle of goal-oriented resource management. It is an iterative algorithm that uses greedy heuristics to find a reallocation that benefits a target transaction class. We define and motivate the cost estimate equations used in the algorithm. We present the results of a set of experiments to investigate the performance of the algorithm.

Keywords: buffer management; database systems performance; resource management; self-tuning database management systems

INTRODUCTION

As database management systems (DBMSs) continue to expand into new application areas, the complexity of the systems and the diversity of database workloads are increasing. In addition to the “classic” relational DBMS workload consisting of short transactions running concurrently, we can now see workloads containing nontraditional data types, such as image, audio, and video, and more complex query processing requirements, such as rules, recursion, and user-defined functions. The workloads may also exhibit very bursty resource demands; for example, a multimedia application could issue a series of conventional SQL queries followed by a
request to retrieve a video clip from the database. As a result of the increasing complexity and workload diversity, DBMSs will have to deal with a wider range of resource demands and execution times in the future.

The tasks of configuring and tuning a DBMS, which are primarily manual exercises, are necessary to ensure acceptable performance. They require knowledge of the characteristics of the system, the data, the workload, and of the interrelationships between them. The increasing complexity of DBMSs and their workloads means that manually managing the performance of a DBMS via direct adjustment of low-level system parameters is becoming impractical.

A self-tuning DBMS is one approach to solving this management problem (Weikum, Möenkeberg, Hasse & Zabback, 2002). A self-tuning system is able to automatically reallocate its resources to maintain acceptable performance in the face of changing conditions and so takes some, or all, of the responsibility for tuning away from the database administrator (DBA). Self-tuning and adaptive techniques have been applied to several aspects of the management problem, including index selection (Chaudhuri et al., 1999; Schiefer & Valentin, 1999), materialized view selection (Agrawal, Chaudhuri & Narasayya, 2000), distributed join optimization (Arcangeli, Hameurlain, Migeon & Morvan, 2004; Khan, McLeod & Shahabi, 2001), and memory management (Brown, Carey & Livny, 1993, 1996; Chung et al., 1995; Sinnwell & König, 1999). Chaudhuri and Weikum (2000) cite the need for self-tuning systems as an important reason to rethink current DBMS architectures.

The buffer area used by a DBMS is particularly important to system performance because effective use of the buffers can reduce the number of disk accesses performed by a transaction. Current DBMSs, such as DB2 Universal Database (DB2) (IBM, 2004), divide the buffer area into a number of independent buffer pools, and database objects (tables and indices) are assigned to a specific buffer pool. The size of each buffer pool is set by configuration parameters, and page replacement is local to each buffer pool. Tuning the size of the buffer pools to a workload is therefore crucial to achieving good performance.

For example, suppose we initially configure a database to have four buffer pools of equal size where indices are assigned to one buffer pool, and tables are assigned to the other three buffer pools. If the index buffer pool is too small to hold at least all the nonleaf pages of the active indices, then there will be excessive swapping on the index buffer pool, and performance will be poor for many of the queries. A more appropriate use of buffer pages is to give more pages to the index buffer pool so that index pages can remain in memory and to take away pages from the table buffer pools where pages are reused less often.

In this paper, we apply the self-tuning concept to managing the buffer pools. We present a self-tuning algorithm called the Dynamic Reconfiguration algorithm (DRF). It is based on the concept of goal-oriented resource management, which allows administrators to specify their expectations, or goals, for performance while leaving it up to the system to decide how to achieve those goals (Nikolaou, Ferguson & Constantopoulos, 1992). The DBA specifies average response time goals for each transaction class in the workload. If one or more classes are not meeting their goals, then DRF chooses a reallocation of buffer pages to buffer pools that will improve the performance of the classes so that their goals can be met.
Related Content

Concurrency Control for Replicated Data in Distributed Real-Time Systems
www.irma-international.org/article/concurrency-control-replicated-data-distributed/51162/

A Rigorous Approach for Mapping Workflows to Object-Oriented IS Models
www.irma-international.org/article/rigorous-approach-mapping-workflows-object/3284/

FOOM - Functional and Object-Oriented Analysis and Design of Information Systems: An Integrated Methodology
www.irma-international.org/article/foom-functional-object-oriented-analysis/3258/

Extended Action Rule Discovery Based on Single Classification Rules and Reducts
www.irma-international.org/chapter/extended-action-rule-discovery-based/8039/

Comparing Metamodels for ER, ORM and UML Data Models
Terry Halpin (2004). Advanced Topics in Database Research, Volume 3 (pp. 23-44).
www.irma-international.org/chapter/comparing-metamodels-orm-uml-data/4352/