The COMFORT Prototype: A Step Towards Automated Database Performance Tuning

Axel Moenkeberg, Peter Zabback, Christof Hasse, Gerhard Weikum
ETH Zurich
Department of Computer Science
CH–8092 Zurich, Switzerland
E–mail: weikum@inf.ethz.ch

1 Project Goals

COMFORT stands for "Comfortable Performance Tuning". The long–term goal that we pursue in the COMFORT project is to automate, to the largest possible extent, the performance tuning of database systems. Tuning of database systems depends critically on the expertise and experience of system administrators and other human tuning experts which are responsible for the setting of system parameters. The purpose of such system parameters, or "tuning knobs", is to adapt the system to the specific characteristics of a given workload. With a wider use of OLTP and other multi–user database applications, on the one hand, and a lack of qualified tuning experts, on the other hand, there is a strong need for simplifying the tricky job of human administrators and ideally automating at least some critical tuning decisions.

Our approach is to derive appropriate tuning heuristics, or "rules of thumb", from quantitative performance models for individual tuning problems, and to incorporate such heuristics in an adaptive or "self–tuning" system architecture. Thus, the goal of automatic performance tuning entails two lines of research. On the one hand, we are investigating specific tuning problems, with emphasis on the challenging problems that are posed by multi–user parallel database systems. On the other hand, we are aiming at architectural principles of a database system that can automatically adapt itself to the workload. Such a system should ideally be

- self–tuning in that it does not require human intervention,
- responsive in that it can quickly react to significant changes of the workload,
- robust in that it guarantees acceptable performance even under "stress" conditions, and
- low–cost in that it can make decisions on–line at acceptable costs of bookkeeping, gathering of statistics, evaluation of analytic models, etc.

Within this framework of a self–tuning database system architecture, we have been addressing the tuning issues of data placement in multi–disk architectures, load control, parallelization and optimization of complex queries, and processor allocation for multi–user parallel database systems [1].

2 Prototype Architecture

The overall architecture of COMFORT can be viewed, in an idealized manner, as a feedback loop. The database system monitors a number of load and performance metrics, and gathers additional long–term statistics on data and workload distributions. This information is managed in a special sort of catalog, the Tuning Information Pool (TIP). The database system varies a number of control parameters based upon the observed performance/load behavior. This control process is driven by heuristic but quantitative rules, derived from simple performance models. The tuning decisions affect the policies employed by the various system components and the resource limits and other thresholds of system components. It is also possible, at least as a last resort, that a human tuning expert can prohibit or overwrite certain types of tuning decisions.

We have built a self–tuning storage system, which runs on a Sequent shared–memory multiprocessor and on Sun computers. This storage system currently consists of a parallel file system, a buffer manager, a transaction manager, and a load control component. The system has been implemented such that it can either manage real data on a real hardware platform, or it can simulate the performance impact of resources (processors, disks) that are not available to us.

3 Coping with Load Dynamics

In the COMFORT project, particular emphasis has been given to the problem of coping with load dynamics. Real–life workloads often exhibit significant fluctuations in the arrival rate of transactions; moreover, the transaction mix and hence the access characteristics of the data evolve over time. This dynamics of the workload poses additional challenges to the tuning expert. However, tuning by a human system administrator is inherently static and cannot react to load fluctuations. Our automatic tuning approach, on the other hand, is able to dynamically adapt the system parameters to the evolving workload. In the following two subsections, the methods for achieving this adaptation are discussed for the problems of dynamic disk load balancing and load control for locking.

3.1 Dynamic Disk Load Balancing

We have developed a parallel file system that includes a set of heuristics for data placement on multi–disk architectures (e.g., disk arrays) [1][3]. These heuristics provide automatic tuning capabilities for data partitioning, especially the tuning of striping units, data allocation, and data reorganization. An underlying assumption is that the system has already collected sufficient statistical information about data access characteristics such as average request size and access frequency of files. However, since access characteristics evolve over time, we also provide dynamic repartitioning and reallocation strategies.

A typical problem in practice is that cold (i.e., infrequently accessed) data may become hot (i.e., frequently accessed) and vice versa. This may incur significant imbalances in the disk load distribution of a multi–disk system. To avoid that the hottest disk becomes a bottleneck already at a relatively low arrival rate of data requests, we employ a dynamic load balancing heuristics that carefully migrates data from hot

SIGMOD /5/93/Washington, DC,USA
© 1993 ACM 0-89791-592-5/93/0005/0542...$1.50

disks onto colder disks. We coined this method "disk cooling" [3]. One of the crucial issues of this method is to take into account the cost of the cooling itself. Since data migration incurs additional load and could actually aggravate the queueing at the hottest disks, the cooling should not be initiated during load peaks. We use various heuristics to ensure that cooling steps are deferred until a point when the impact on the regular load is acceptable.

Figure 1 illustrates the dynamic behavior of the disk cooling method in a trace–driven performance experiment, based on an OLTP page reference trace. This workload exhibited heavily skewed access frequencies both across files and within the hot files, and, in addition, dynamic changes of these access frequencies and the overall arrival rate of requests. The charts show the average response time of page requests without cooling and with cooling as well as the cooling frequency for the latter case, evolving over time. In both methods, all files were striped across 24 disks, using track–sized striping units and round–robin allocation.

As the response time chart shows, the cooling method could not improve response time in the initial light–load period, since the load imbalance of the vanilla method did not yet incur any severe queueing. However, the cooling method did collect statistics about the access frequencies of pages during this period. This enabled the cooling method to rebalance the disk load by migrating fractions of files. Then, during the load peak, the cooling method achieved a response time improvement by a factor of 1.7, due to better load balance and, hence, reduced queueing. Note that the cooling method was careful enough to suspend the data migration steps during the load peak, as shown in the cooling frequency chart.

3.2 Adaptive Load Control for Locking

Load control is necessary to prevent a database system from data–contention or memory–contention thrashing, caused by excessive lock conflicts or excessive buffer replacements that may occur due to temporary load peaks. The load control method that is adopted by virtually all commercial database systems is to limit the degree of multiprogramming (DMP), that is, the maximum number of transactions that are allowed to execute concurrently.

We have developed an adaptive load control method for the avoidance of data–contention thrashing [2]. Unlike the DMP method, our adaptive method does not depend on any manu-

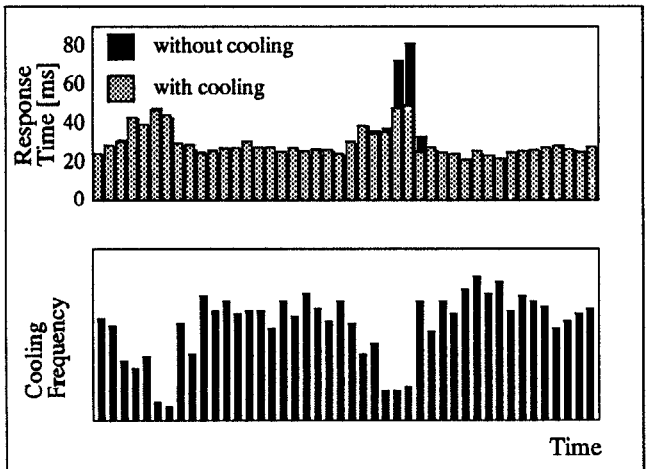

Figure 1: Performance of the dynamic load balancing

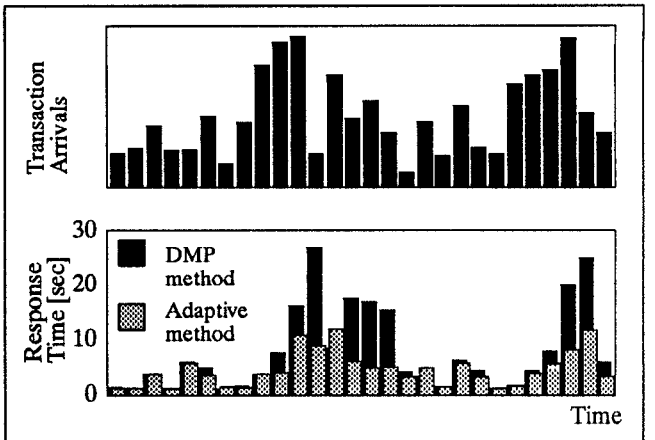

Figure 2: Performance of the adaptive load control

al tuning directives; it is completely self–tuning in the sense that it adapts the DMP to the current workload dynamically and automatically. The basic principle of this method is to monitor a data–contention performance metric called the conflict ratio, and to react to critical changes of the conflict ratio by temporarily suspending the admission of newly arriving transactions or by cancelling blocked transactions that block other transactions. The adaptive load control method copes well with dynamic load fluctuations, since it adapts the system to the evolving transaction mix, whereas the DMP method uses a fixed DMP as a (bad) compromise.

Figure 2 illustrates the behavior of the adaptive load control method. It shows the transaction arrivals and the average response time for the adaptive method and the DMP method, evolving over time. These figures were obtained from a trace–driven performance experiment, based on a one–hour page reference trace from a large OLTP system. This workload exhibited significant data–contention problems during the two load peaks shown in Figure 2. The DMP value of the DMP method was tuned optimally, by exhaustive experimentation for the given workload.

During the two load peaks, our adaptive method outperformed the DMP method by a factor of 3. In addition, our method performed significantly better also in the interval between the two load peaks. The problem with the DMP method was that a backlog of transactions was created during the first load peak. Our adaptive method, on the other hand, smoothed out the fluctuations in the transaction arrivals and the transaction mix, by dynamically varying the DMP depending on the current mix. Thus, the adaptive method could achieve acceptable response time even during the load peaks.

References

[1] G. Weikum, C. Hasse, A. Moenkeberg, M. Rys, P. Zabback, The COMFORT Project (Project Synopsis), Int. Conf. on Parallel and Distributed Information Systems, San Diego, 1993

[2] A. Moenkeberg, G. Weikum, Performance Evaluation of an Adaptive and Robust Load Control Method for the Avoidance of Data–Contention Thrashing, Int. Conf. on Very Large Data Bases, Vancouver, 1992

[3] G. Weikum, P. Zabback, P. Scheuermann, Dynamic File Allocation in Disk Arrays, ACM SIGMOD Int. Conf. on Management of Data, Denver, 1991