Applying Probabilistic Adaptation to Improve the Efficiency of Intra-Query Load Balancing

Daniel M. Yellin, IBM Israel Software Lab, Jerusalem, Israel
Jorge Buenabad-Chávez, Departamento de Computación, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Mexico City, Mexico

ABSTRACT

In the context of adaptive query processing (AQP), several techniques have been proposed for dynamically adapting/redistributing processor load assignments throughout a computation to take account of varying resource capabilities. The effectiveness of these techniques depends heavily on when and to what they adapt processor load assignments, particularly in the presence of varying load imbalance. Most existing approaches to this problem use heuristics based only upon the current machine load levels. The authors provide an algorithm, prAdapt that probabilistically predicts the future load on processors, based upon the recent history. It uses this prediction to evaluate the expected performance of different alternative solutions, taking into account the cost of the adaptation itself. If it finds a better solution than the current load distribution policy, it adapts to that distribution. Using a simulation based evaluation; they compare prAdapt to other approaches for AQP reported in the literature. The authors’ simulation results indicate that prAdapt often outperforms these other approaches.

Keywords: Adaptive Querying Processing (AQP), Algorithm, Autonomic Systems, Load Assignments, Predictive Load Balancing, Probabilistic Adaptation

INTRODUCTION

The goal of an autonomic load balancing system (Ganek & Corbi, 2003; IBM, 2006; Huebscher & McCann, 2008; Sterritt, 2005) is to distribute load across multiple processors in such a way as to optimize system performance. An autonomic load balancing system will continuously monitor the performance of the system across all processors, and decide if a new policy for distributing work is warranted. For instance, one processor may be highly overloaded, and work will need to be redirected from that processor to other processors. It may require that even some of the existing load on that processor be moved to another processor.

In this paper we focus on a specialized domain of computation called Adaptive Query
Processing (AQP), which uses a form of partitioned parallel processing to provide scalable query processing. Since this technique performs at the speed of the slowest participant, it has prompted several proposals for adaptivity specifically aimed at load balance (Gounaris et al., 2005; Raman, Han, & Narang, 2005; Shah et al., 2003). These proposals differ from most work on dynamic load balancing for parallel databases (DeWitt et al., 1992; Rahm & Marek, 1993) in changing load balance within, rather than between, queries. An evaluation and comparison of these approaches can be found in Paton et al. (2006, 2009).

Most AQP approaches for deciding when to adapt use only current machine load levels to make their decision. In this paper we explore a probabilistic approach that uses past performance (machine load levels) over a period of time to predict future performance. This prediction is then used to decide when to adapt. We provide empirical evidence for the effectiveness of our approach by comparing it to other algorithms in the literature. An initial version of this algorithm was presented in Yellin, Buenabad-Chávez, and Paton (2008). This paper expands upon that work providing new theoretical results, new variations on the algorithm, and more focused experiments to validate the efficacy of the approach. The experimental approach used to evaluate our approach is based upon the approach in Paton et al. (2009). We chose to keep the same experimental setting in order to better evaluate our approach directly against previous benchmarks. Part of the contribution of this paper is to show that using some fairly straightforward adaptive techniques to decide when to adapt, performance can be significantly improved.

The rest of this paper is organized as follows. In the next section we present the AQP problem and prior work on this problem. We focus on two existing algorithms, Flux (Shah et al., 2003) and RevisedDelta (Paton et al., 2009). The former is the seminal work in AQP upon which almost all other research is based, while the latter is the work that inspired the prAdapt algorithm in this paper. We also provide a theoretical framework to analyze the effectiveness of different approaches. In Appendix A and Appendix B we show that under the assumption that the immediate future will imitate the recent past, and under some additional assumptions, RevisedDelta (or a slight variant) is the optimal algorithm. However, the assumption that the future will imitate the past is not always valid, and for that reason in the section following the next one we describe the prAdapt algorithm, which attempts to probabilistically predict the future given the past. In the Experimental Methodology section we describe our methodology and in the Results Section we state the results of our experiments comparing prAdapt to Flux and to RevisedDelta. prAdapt has several parameters, and we also provide experimental evidence as to the effect of these parameters on the performance of the algorithm. The last section presents our conclusion, additional related work, and directions for future research.

PREVIOUS WORK ON ADAPTIVE QUERY PROCESSING

In this section we first describe the AQP problem, and then two algorithms, Flux and RevisedDelta, for adaptive load balancing in the context of AQP. This provides background for the prAdapt algorithm described in the next section. Both Flux and RevisedDelta will be used as the basis for experimental comparison with prAdapt.

Adaptive Query Processing For Load Balancing

In Adaptive Query Processing (AQP) with load balancing, the problem to be solved is as follows, illustrated for the case of a query involving a single join. The result of a query $A \bowtie B$ is represented as the union of the results of a collection of plan fragments $F_i = A_i \bowtie B_i$, for $i = 1 \ldots P$, where $P$ is the level of parallelism. Each of the fragments $F_i$ is executed on a different computational node. The tuples in each $A_i$ and $B_i$ are identified by applying a hash
Related Content

Synthesis of Analog Circuits by Genetic Algorithms and their Optimization by Particle Swarm Optimization
www.igi-global.com/chapter/synthesis-analog-circuits-genetic-algorithms/38455?camid=4v1a

Mixing Workflows and Components to Support Evolving Services
www.igi-global.com/article/mixing-workflows-components-support-evolving/48263?camid=4v1a

Application of Machine Learning Techniques to Predict Software Reliability
www.igi-global.com/article/application-machine-learning-techniques-predict/47067?camid=4v1a
Fault Detection and Isolation for Switching Systems using a Parameter-Free Method