Lecture Notes in Computer Science 1182
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Advisory Board: W. Brauer D. Gries J. Stoer

Waqar Hasan

Optimization of SQL Queries for Parallel Machines

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany

Juris Hartmanis, Cornell University, NY, USA

Jan van Leeuwen, Utrecht University, The Netherlands

Author

Waqar Hasan
Stanford University, Department of Computer Science
Stanford, CA 94305, USA
E-mail: hasan@db.stanford.edu

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Hasan, Waqar:
Optimization of SQL queries for parallel machines / Waqar Hasan. - Berlin ; Heidelberg ; New York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ; Santa Clara ; Singapore ; Tokyo : Springer, 1996
(Lecture notes in computer science ; 1182)
Zugl.: Stanford, CA, Univ., Diss.
ISBN 3-540-62065-6
NE: GT

CR Subject Classification (1991): H.2, H.3, E.5

ISSN 0302-9743
ISBN 3-540-62065-6 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10550447 06/3142 - 5 4 3 2 1 0 Printed on acid-free paper

To my father

Dr. Amir Hasan

for showing me the paths that I follow.

Foreword

Performance in computing, and particularly in data access, is crucial as our dependence on computing becomes pervasive. Continued innovation is essential for increases in performance to keep up with our requirements: at any time in the past there has been a majority of tasks which can be performed in time, while there has been a remainder, the tail of the distribution, that provides a challenge. In addition, we are always facing some problems that are in the infeasible range.

As examples for the three classes of problems, namely satisfactory, challenging, and infeasible, we see in the arena of information processing, respectively, routine business processing, delivering information for decision-making from distributed large databases, and prediction of future events from models and historical data.

Performance in a computing system is improved in two dimensions: using higher speed in the individual modules and increasing the number of modules that can operate in parallel. These two dimensions are not independent, since the coordination and communication required in parallel execution increases the amount of work to be done by each module. Crucial in this balance is hence the granularity, the size of the modules. At the very fine grain the tradeoffs are well understood. Computer architectures, in various generations, have moved from say 8-bit, to 16-bit, 32-bit, and even wider paths. There is, however, a diminishing return here: if the natural data elements are small, and interact to inhibit parallel operation, then wider data paths do not provide an advantage.

Databases deal with mixed granularity. The primitive elemental values being stored are often small, but are aggregated into records of hundreds of elements, and then stored in files or tables containing thousands or millions of records. Relevant data for even a modest problem may be found on dozens of computers distributed anywhere over world-wide networks.

This monograph provides both insights and algorithms pertaining to parallel operation at a practical granularity relevant to database system operations. Tables or object classes provide sizable objects that can be treated in parallel while permitting serial, pipelined overlap. Modules can also be cloned by partitioning and replicating data objects. These approaches interact with each other as well, making the selection of effective processing schedules yet

more complex. Fortunately, the algebras over these objects are well-behaved, so that their operations can be completely and precisely defined.

This work provides an approach that balances the advantages and costs of parallel execution. Module granularity is determined by the actual operations being scheduled while respecting intrinsic limits on available parallelism such as timing and data-placement constraints and accounting for the trade-off between using parallel execution and incurring communication costs. The result is applicable to modern system configurations, where computation is performed on pipelining-capable workstations operating in parallel. Further research will have to focus on dynamic aspects of parallel computation, letting the scheduling itself overlap with the computation, since this work seems to be able to exploit all the information likely to be available prior to execution in practical systems.

Stanford, California, USA *Gio Wiederhold*
September 1996

Preface

This book is about optimization techniques to determine the best way of exploiting parallel execution for SQL queries against large databases. It is the published version of my PhD dissertation at Stanford University. The techniques in this book are useful in the construction of SQL compilers that can exploit parallel machines effectively.

SQL permits questions to be posed declaratively. Users are insulated from the physical hardware and the layout of the data and thus are able to avoid the complex procedural details of programming a parallel machine. A Data Base Management System (DBMS) answers a SQL query by first finding a procedural plan to execute the query and subsequently executing the plan to produce the query result. This book provides techniques for the problem of *parallel query optimization*: Given a SQL query, find the parallel plan that delivers the query result in minimal time.

I express my gratitude to the people and organizations that made my thesis possible. Gio Wiederhold was a constant source of intellectual support. He encouraged me to learn and use a variety of techniques from different areas of Computer Science. Rajeev Motwani helped enhance my understanding of theory and contributed significantly to the ideas in my thesis. Jeff Ullman was a source of useful discussions and I thank him for his helpful and incisive comments. Ravi Krishnamurthy served as a mentor and a source of interesting ideas and challenging questions. Hector Garcia-Molina provided helpful advice. Jim Gray helped me understand the realities of parallel query processing.

My thesis topic grew out of work at Hewlett-Packard Laboratories and was supported by a fellowship from Hewlett-Packard. I express my gratitude to Hewlett-Packard Company and thank my managers Umesh Dayal, Dan Fishman, Peter Lyngbaek, and Marie-Anne Neimat for management and intellectual and moral support.

I thank Tandem Computers for providing access to a parallel machine, the NonStop SQL/MP parallel DBMS, and for permitting publication of experimental results. I am grateful to Susanne Englert, Ray Glasstone, and Shyam Johari for making this possible and for helping me understand Tandem systems.

The following friends and colleagues were a source of invaluable discussions and diversions: Sang Cha, Surajit Chaudhuri, Philippe DeSmedt, Mike Heytens, Curt Kolovson, Stephanie Leichner, Sheralyn Listgarten, Arif Merchant, Inderpal Mumick, Pandu Nayak, Peter Rathmann, Donovan Schneider, Arun Swami, Kevin Wilkinson, Xiaolei Qian.

My thesis would not have been possible without the support and understanding of my family. I thank my father, Dr. Amir Hasan, for providing the inspiration to pursue a PhD. I thank my mother, Fatima Hasan, my brothers Safdar, Javed, and Zulfiquar, and sister Seemin for their love and encouragement. I owe a debt to my wife Shirin and son Arif for putting up with the long hours that made this work possible.

Stanford, California, USA *Waqar Hasan*
September 1996

Abstract

Parallel execution offers a solution to the problem of reducing the response time of SQL queries against large databases. As a declarative language, SQL allows users to avoid the complex procedural details of programming a parallel machine. A DBMS answers a SQL query by first finding a procedural plan to execute the query and subsequently executing the plan to produce the query result. We address the problem of *parallel query optimization*: Given a SQL query, find the parallel plan that delivers the query result in minimal time.

We develop optimization algorithms using models that incorporate the sources of parallelism as well as obstacles to achieving speedup. One obstacle is inherent limits on available parallelism due to parallel and precedence constraints between operators and due to data placement constraints that essentially pre-allocate some subset of operators. Another obstacle is that the overhead of exploiting parallelism may increase total work thus reducing or even offsetting the benefit of parallel execution. Our experiments with NonStop SQL, a commercial parallel DBMS, show communication of data across processors to be a significant source of increase in work.

We adopt a two-phase approach to parallel query optimization: *join ordering and query rewrite* (JOQR), followed by *parallelization*. The JOQR phase minimizes the total work to compute a query. The parallelization phase extracts parallelism and schedules resources to minimize response time. We make contributions to both phases. Our work is applicable to queries that include operations such as grouping, aggregation, foreign functions, and set intersection and difference, and joins.

We develop algorithms for the JOQR phase that minimize total cost while accounting for the communication cost of repartitioning data. Using a model that abstracts physical characteristics of data, such as partitioning, as colors, we devise tree coloring algorithms that are efficient and guarantee optimality.

We model the parallelization phase as scheduling a tree of inter-dependent operators with computation and communication costs represented as node and edge weights. Scheduling a weighted operator tree on a parallel machine poses a class of novel multi-processor scheduling problems that differ from the classical in several ways.

We develop and compare several efficient algorithms for the problem of scheduling a pipelined operator tree in which all operators run in parallel

using inter-operator parallelism. Given the NP-hardness of the problem, we assess the quality of our algorithms by measuring their performance ratio which is the ratio of the response time of the generated schedule to that of the optimal. We prove worst-case bounds on the performance ratios of our algorithms and measure the average cases using simulation.

We address the problem of scheduling a pipelined operator tree using both pipelined and partitioned parallelism. We characterize optimal schedules and investigate two classes of schedules that we term symmetric and balanced.

The results in this thesis enable the construction of SQL compilers that can exploit parallel machines effectively.

Table of Contents

List of Figures

List of Tables