Data Distribution Strategies for Providing Database Scalability in E-commerce Applications[1]

Haiwei Ye
Université de Montréal
ye@iro.umontreal.ca

Brigitte Kerhervé
Université du Québec à Montréal
Kerherve.Brigitte@uqam.ca

Gregor v. Bochmann
University of Ottawa
bochmann@site.uottawa.ca

Don Bourne
IBM Canada Ltd
dbourne@ca.ibm.com

Abstract

The number of users of e-commerce applications is increasing, and users are becoming more and more sensitive to the quality of the offered services. This paper discusses performance and scalability issues for back-end parallel or distributed database servers used in e-commerce applications. We argue that database scalability cannot be achieved without considering efficient data placement. In particular data distribution strategies should consider the specifics of e-commerce applications and user expectations in terms of quality of service. We propose a generic data distribution strategy integrating user class information.

1. Introduction

Quality of Service (QoS) support in e-commerce systems means that the different components of the system all have to work toward providing security, high availability and fast response times [1]. In this paper we are interested in business-to-consumer (B2C) e-commerce and we focus on performance and scalability, which are the great challenges to be met for making e-commerce successful and largely adopted by customers.

In our previous work, we have studied QoS support in distributed database systems, and we have focused on integrating information on network and server performance for enhancing distributed query processing algorithms with adequate cost models [4, 5]. We are now interested in applying such approaches on how to wisely layout the data across the nodes in e-commerce systems. Because we believe the performance and scalability of a database system are contingent on the data distribution. In addition, a poor data distribution strategy can result in a non-uniform distribution of the load and the formation of bottlenecks.

In this paper, we concentrate on data distribution in e-commerce applications. Such applications are specific compared to those addressed in traditional data distribution strategies since (i) they are less dynamic in the sense that access patterns to the database are static and can be obtained from the application source code; (ii) they need more consideration of the user expectations in terms of quality of the provided service; and (iii) their capability to satisfy these expectations should adapt to the high variations in the number of connected customers.

Traditional data placement [2, 6] strategies are derived based on application characteristics (including data access pattern and data access frequency) and SQL complexity (referring to the number of tables participating in the query and number of joins involved). These factors are not sufficient to propose an optimal partitioning schema for the e-commerce application if user's concerns are ignored. In this paper we extend traditional way of data distribution by adding user class information. That is, our data distribution strategies should integrate distribution decisions based on user QoS expectations as well as on e-commerce application characteristics.

Defining classes of users is a way to differentiate users according to their QoS expectation in order to provide different levels of service based on priorities. This is an important factor to take into account to come up with an optimal data distribution strategy. Under the assumption that a higher priority user class should get better performance in terms of response time, we want to allocate the required data, for example, to more nodes, so that parallelism can be utilized for query execution. This also requires that the DBMS supports such priority awareness so that to route the query from different classes of users to different node groups.

[1] This work was supported by a grant from the Canadian Institute for Telecommunication Research (CITR), under the Network of Center for Excellence Program of the Canadian Government, a collaborative research and development grant from NSERC no CRD-226962-99 and by a student fellowship from IBM.

The remainder of the paper is organized as follows. Section 2 demonstrates our approach for the data distribution and proposes a generic methodology for data distribution in e-commerce applications. In Section 3, we present our conclusions and point out some future work.

2. Our approach

As pointed out in [3], "the customer's behavior while interacting with an e-commerce site has impacts on the IT resources of the site and on the revenue of the e-store". Eventually, this customer's behavior will affect the database access pattern. We believe it is also important to factor in user's information while organizing the e-store data for the back-end database system. For this study, we take different classes of users into account during the design of data distribution strategy.

2.1. User classes

A user class is a generalization of a number of users sharing common characteristics. Classification of users can be based on different policies and criteria. For example, different users may exhibit various patterns of navigation through an e-commerce site, therefore based on their navigation behavior, we can segregate users into two classes: *buyer* and *browser*. Another example of segregating users can be as simple as priority based. That is we differentiate users into different classes according to, for example, their profit brought to an e-store. Thus an e-commerce site tries to provide better service to *higher priority* user than to *lower priority* user. A user perceived quality of service in e-commerce application could be the response time for each request. Or in a hybrid mode, these criteria can be combined together. For example, in the high priority user class, we have the buyer class and browser class. Similarly, the low priority class can also be divided into buyer and browser categories.

How to classify users and what criteria should be applied deserve a careful study. However, this is not the main concern of this paper. The main focus of the user class aspect is how to provide the possibility to consider such a factor. Therefore, what is important to our study is that we should realize the existence of different classes of users exist which may generate different workload to the database system. Accordingly, by collecting information relevant to user class, we can determine what communication modes, or database resources are being used by a particular class. This may allow for improvements to the system performance based on usage.

The basic idea of user class consideration in our data distribution strategy is that by allocating different resources, e.g. number of database nodes, to different classes of users, we can route the database requests to different resources. Partitioning requests not only improves the performance, but also provides different QoS levels. If the node cluster is our main concern, we could apply different distribution schema for each set of nodes. For example, the data distribution schema for *browser* user should be different from the one for *buyer* user. Because most activities of users in the *browser* class are searching and browsing, those data related to payment and shopping cart might not utilize the same resources as the data storing product information. On the other hand, for users in the *buyer* class, information related to payment has to be accessed with very good performance. Based on this heuristic, if, for example, 5 database nodes are available for each class, the following distribution schema might be reasonable: for *browser* class, 4 nodes could be allocated to product related data and only 1 node to store the data relevant to payment activities; for *buyer* class, 4 nodes dedicating for payment information will improve the performance and 1 node for product data.

Differentiating users can also be achieved by providing different data representation and data quality. For example, the data stored in the database for high priority user class could include multimedia format. In addition, data quality for different users could be different too. For example, one metric defined in data quality is timeliness, i.e. outdate or up-to-date. For high priority user, we may set the update frequency higher than that for low priority user to provide a better data quality (in term of timeliness in this example). This paper focuses on the performance category of QoS.

2.2. Data distribution strategies

The general method in our study is to allocate different database resources to different classes of user first and then apply the distribution strategy to each user class. Such procedure is expressed in the following steps:

1) Use resource allocation strategy to decide the number of database nodes of each user class.
2) For each user class, executing following steps:
   a) Deriving database access pattern from the characteristics for this class
   b) Collecting and analyzing the related statistics information
   c) Applying data distribution algorithm

Resource allocation. The first step is a resource allocation issue, which is a question of how to divide up the available resources among available user classes. To simplify the discussion, we consider, in this paper, one database node as one unit of resource. In the implementation, this abstract resource could be mapped to different hardware and software, such as CPU, memory, disk and network bandwidth.

Different QoS levels can be provided by either *shared* resource or *segregated* resource. In the case of shared resource, all the user classes access the same

resource. When there is plenty of resource, all the QoS requirements can be satisfied. When the resource utilization reaches to a point (or *threshold*, for example 80%) such that the system cannot guarantee all the QoS requirements, some admission control policies may be triggered to reject the requests from low priority user. In this method, an important step is how to decide the reasonable threshold. A high threshold may lead to service degradation for high priority user. In contrast, a low threshold may reject too many low priority users and lead to under utilization of system resource. This value could be derived from the simulation and should be later tuned for the purpose of changing policy or workload.

In the case of segregated resource, one feasible solution is to keep several copies of an identical database, with each copy for a particular user class, and then allocate the resources to different user classes. If the resource is the number of database nodes, then the allocation is to decide how many disjoint sets of nodes needed for different classes. The example we showed at the end of Section 2.1. belongs to this case. In this method, we have to address the problem of how to decide how many nodes to choose for each class. The decision is made depending on several factors such as the workload type, the navigation pattern for each class as well as the total available resources. This is also a policy issue. To do this, we can analyze the HTTP log and monitor resource utilization for each node. For example, from the HTTP log and monitor information, we observe that 80% of the total requests are searching product and only 20% involve payment activity, we could allocate 4/5 of the total database nodes to *browser* class and 1/5 to *buyer* class.

No matter which resource allocation strategy is applied, the data distribution strategy should allow for the use of flexible mechanisms that can adapt to workload change. For example, the resource allocation method shown in the above example will not be suitable if the observation of the workload shows that the payment activity increase to 40%. Therefore, with the time goes on, the number of users in each class will change and thus the number of nodes reserved for each class should also be adjusted to the new circumstance.

Distribution strategy. Ideally, we attempt to duplicate all the tables among all the available nodes for one user class. Because this will provide the maximum flexibility of utilizing the parallel techniques and reducing data transmission. However, this kind of replication strategy will greatly degrade the system's performance if the environment is updated intensively (such as insert, update or delete SQL), as required by the application. Therefore, data partitioning is useful in scenarios where there are frequent updates.

Accordingly, our heuristic is to pick up those tables that can be replicated first and then, for the rest of the tables, decide on the partitioning key. The data distribution algorithm can be summarized in two steps: (1) Grouping tables into two sets for replication (denoted as $\mathbf{S_{rep}}$) and partitioning (denoted as $\mathbf{S_{part}}$), respectively; and (2) Selecting the partitioning key for tables to be partitioned

The replication strategy can be chosen to duplicate all the tables in $\mathbf{S_{rep}}$ across all the available nodes for that particular class in the database server. For selecting partition key, we can use the frequency information for join or update attributes as our selection criterion. In addition, the selection of the partitioning key should also obey the constraints imposed by different implementations of the DBMS.

3. Conclusion and Future Work

This paper has discussed issues related to the scalability and performance of back-end database servers used in e-commerce applications. We proposed strategies for data distribution considering e-commerce application characteristics as well as user classes. We proposed to differentiate users according to their access patterns to the database. Our strategies include decisions on the configuration of the database in terms of the number of nodes as well as distribution strategy associated to each user class. The distribution strategy consists both partitioning method and replication method.

This work constitutes a first step for providing data distribution strategies allowing database scalability in e-commerce applications. Future work includes improvement of our data distribution strategy to provide dynamic data reorganization and additional studies on user classes for a wider definition of the access patterns and resource needs as well. We also plan to integrate our approach into the algorithm for QoS based query processing.

References

[1] G. v. Bochmann, B. Kerhervé and M. Mohamed-Salem, published in a chapter in book *Quality of service management issues in electronic commerce applications*, Electronic Commerce Technology Trends: Challenges and Opportunities, IBM Press, 2000.

[2] M.L. Lee, M. Kitsuregawa, B.C. Ooi, K. Tan. A. Mondal, *Towards Self-tuning data placement in parallel database systems*, SIGMOD 2000, 225-236.

[3] D. A. Menasce, V. A.F. Almeida, *Scaling for E-Business Technologies, Models, Performance, and Capacity Planning*, ISBN:0-13-086328-9, Prentice Hall Canada, 2000

[4] H. Ye, B. Kerhervé and G.. v. Bochmann, *Quality of service aware distributed query processing*, 10th DEXA Workshop on Query Processing in Multimedia Information Systems (QPMIDS), Florence, Italy, 1-3 Sept. 1999

[5] H. Ye, G. v. Bochmann, B. Kerheré, *An adaptive cost model for distributed query processing*, UQAM Technical Report 2000-06, May 2000.

[6] D. C. Zilio, A. Jhingran, S. Padmanabhan, *Partitioning Key Selection for a Shared-Nothing Parallel Database System* IBM Research Report RC 19820 (87739) 11/10/94