A Prediction-Based Query Processing Strategy in Mobile Commerce Systems

Chiang Lee and Chih-Horng Ke
National Cheng-Kung University, Taiwan (R.O.C.)

The rapid advances and popularity of the Internet encourages the wide dissemination of electronic commerce. Various applications of electronic commerce, e.g., querying supermarket catalogs, purchasing products, booking a hotel room of a trip, etc., are rapidly increasing. Moreover, the emergence of mobile computing makes it possible for users to access information anywhere and at anytime. With the merging of these two technologies, millions of users in the near future could carry a mobile device requesting services through electronic commerce applications. While serving such a mobile user, the server should consider how to minimize the response time of the request. However, conventional strategies for query processing are no longer adequate in the mobile environment. This is mainly because the user may not be at the same site as where he/she issued the request when the result of the request is ready to be delivered to the user. The issue becomes more complicated when the user’s request involves data residing in multiple data sources. This paper proposes some strategies for query processing in a mobile commerce environment. All the strategies are presented, along with a cost analysis as the theoretical basis for the support of our design. Through the analysis of each strategy, we find the weakness of the strategy and improve it by a new design. Finally, the performances of all strategies are studied and compared to find the performance difference of the strategies.

Electronic commerce has recently emerged as an appealing new application in the distributed computing field. The rapid growth of the Internet and the widespread popularity of the Web allow companies to overcome many of the physical constraints that often prevent them from doing business in distant markets, which means that an e-commerce market is fundamentally global (Chol & Whinston, 1999). More and more applications of electronic commerce are designed to conduct business between enterprises and customers. Additionally, recent advances in wireless related technologies have led to the emergence of mobile computing devices and wireless communication systems which allow the user to access online information without any time and space constraints (Imielinski & Badrinath, 1994). The merging of these two great waves, forming the paradigm of mobile commerce, is expected to create numerous business opportunities for various types of commercial companies. An interesting application is, for example, an electronic catalog or newsletter which is updated by retailers from anywhere and at anytime. In particular, it allows retailers to change product prices on the fly and add or remove products at any time. The data could be centralized, permitting single-copy updates which immediately apply to all customers. In this manner, maintenance is simplified, and materials can get online quickly and can originate from real-time sources. We refer to the systems managing such applications as the mobile commerce systems.

Various issues in such a mobile commerce environment need to be studied. One of them is related to the efficiency of processing a user’s request in this environment. Through a friendly interface of a mobile device, the user can access the needed information by issuing queries or transactions. In conventional distributed database systems, the site at which a query is issued is also the site to which the result is sent. However, in a mobile commerce system, these two sites may often be different due to the user’s mobility. Conventional query processing/optimization strategies become very inefficient or even invalid in this new environment because the mobile user may have already moved to a new location and will not receive the result if the result is sent to the previous location where the mobile user stayed. Redirecting the result to the user’s new location causes a waste of the precious communication bandwidth and the user will also have to tolerate the delay. Hence, new processing strategies consid-

* This research was supported by the National Science Council under Grant NSC89-2213-E-006-008.
erating the mobility of a user need to be designed to optimize the process.

A simple idea of resolving this problem is to process the query at the site which has the required data. After the result is obtained, the system finds the user’s new location through the user’s HLR (Home Location Register) and sends the result to the new location (EIA/TIA, 1991; Lyckessell, 1991). This however, may not always be the best solution. When the size of the query result is larger than its input (due to the join of two tables, for instance), it becomes costly to send a large result to the user. In this situation, it is better to send the input data of a query (which is smaller than the result) to one or more locations that the mobile user will most likely visit at the time the query processing is completed, and let the systems at these locations process the query. Then, the result can be transmitted to the user right after the query execution is completed. Of course, here we assumed that the systems at these locations have the capability to do the job. Based on this idea, we propose a prediction-based query processing strategy which takes a user’s mobility into account so that it avoids the problems in the first method. We also design a cost model for these strategies to analyze their costs and the performance study shows that a significant improvement is indeed obtained by using the proposed method. Finally, we stress that although query processing strategies in the mobile computing environment have been studied in the past (Alonso & Ganguly, 1992; Dunham, Helal & Balakrishnan, 1997; Ganguly & Alonso, 1993), they all focused on very different issues (for instance, under the constraint that a mobile device’s battery life is limited, or in the paradigm of transaction processing, etc.). To the best of our knowledge, this is the first work considering users’ mobility while processing a large amount of data.

This paper is organized as follows. First, we present the environment under which this research is conducted. The assumptions we made for this research are also given in this section. Then, we discuss two scenarios of query processing in a mobile commerce system: a query involving only one site and a query involving multiple sites. One section discusses the first scenario and presents algorithms for that issue, and then concentrates on the second scenario. Following each algorithm presented in these two sections, we immediately give the cost analysis of the algorithm. The performance evaluation results are depicted, and finally, conclusions and the future of this work is presented.

ENVIRONMENT AND ASSUMPTIONS

Figure 1 shows a simplified version of the mobile computing environment. Geographical regions are divided into cells, each being a radio port coverage area (a dash circle). The station serving a cell is called a base station (BS). If a mobile user (MU) equipped with a PDA in BS1, for instance, would communicate with another mobile user equipped with a laptop in BS2, the routing path of message is PDA —> BS1(DB1) —> BS2(DB2) —> Laptop. This mobile environment consists of wireline and wireless networks. Communication between base stations is normally through wireline network and that between a base station and its MUs through wireless channels. The wireline network can be in any topological configurations. To avoid getting involved in the details of network configurations, we assume that the wireline network is representable as a bus.

In practice, a database system may serve one or more base stations. To simplify our discussion but without loss of generality, we assume that one database is associated with each base station (Jain, Lin, Lo, & Mohan, 1994). Hence, if a few cells are served by one database, we can simply regard those cells as one big cell (served by one database). A database of a cell is assumed to contain not only MUs’ profile data and those required for MUs’ location management, but also the e-commerce data that are frequently accessed by the MUs. Each database system has the same capabilities (such as processing power) in processing these data. In this paper, we only consider queries with one database join operation. The reasons are that it is a frequently used operation in various types of database applications, and also it is a very time-consuming operation. We suppose that multiple join queries are less often seen in such mobile commerce systems because the interface for a user with a mobile device would not be too complicated. (Some mobile commerce users may submit complicated requests to the system. But that should be limited to a relatively small group of sophisticated users only.) The queries that are issued from such an interface would be mostly simple queries, such as single join queries. Even though we only study the single join case, our developed method can still be applicable to the case of multiple join queries.

While processing a query, the longer the query needs to be processed, the more likely the MU will have moved to a new location when the execution of the query is completed. In our research issue, the input data of this operation may be dispersed among multiple databases, a very normal situation. We assume that the MU’s movement history is collectable and analyzed periodically so that the MU’s new location at the time while the query processing is completed is predictable, but with a certain error rate (Brown & Mohan, 1997; Lee, Ke, & Chen, 2001; Markoulidakis & Anagnostou, 1995; Zoumozi & Dassanayake, 1997). In computing query execution cost, we ignore the CPU cost because it is much lower than the disk I/O cost and the communication cost. As communication is always expensive in a distributed environment, the data involved in a query should be processed either in the database system(s) that contains those data or in the one(s) that the MU will likely appear when the query execution is completed. By analyzing the collected movement history of a MU, we can get a list of cells (locations) in which the MU will most likely appear when the query processing is completed. Query execution