Multiversion Data Broadcast Organizations

Oleg Shigiltchoff[1], Panos K. Chrysanthis[1], and Evaggelia Pitoura[2]

[1] Department of Computer Science
University of Pittsburgh. Pittsburgh, PA 15260, USA
{oleg,panos}@cs.pitt.edu
[2] Department of Computer Science
University of Ioannina, GR 45110 Ioannina, Greece
pitoura@cs.uoi.gr

Abstract. In recent years broadcasting attracted considerable attention as a promising technique of disseminating information to large number of clients in wireless environment as well as in the web. In this paper, we study different schemes of multiversion broadcast and show that the way broadcast is organized has an impact on performance, as different kind of clients needs different types of data. We identify two basic multiversion organizations, namely Vertical and Horizontal broadcasts, and propose an efficient compression scheme applicable to both. The compression can significantly reduce the size of the broadcast and consequently, the average access time, while it does not require costly decompression. Both organizations and the compression scheme were evaluated using simulation.

1 Introduction and Motivation

The recent advances in wireless and computer technologies create expectation that data will be "instantly" available according to client needs at any given situation. Modern client devices often are small and portable, therefore they are limited in power consumption. As a result, the significant problem arises: how to transfer data effectively taking into consideration this limitation.

One of the schemes which can solve this problem is *broadcast push* [1]. It exploits the asymmetry in wireless communication and the reduced energy consumption in the receiving mode. Servers have both much larger bandwidth available than client devices and more power to transmit large amounts of data.

In broadcast push the server repeatedly sends information to a client population without explicit client requests. Clients monitor the broadcast channel and retrieve the data items they need as they arrive on the broadcast channel. Such applications typically involve a small number of servers and a much larger number of clients with similar interests. Examples include stock trading, electronic commerce applications, such as auction and electronic tendering, and traffic control information systems. Any number of clients can monitor the broadcast channel. If data is properly organized to cater to the needs of the client, such a scheme makes an effective use of the low wireless bandwidth. It is also ideal to achieve maximal scalability in regular web environment.

Y. Manolopoulos and P. Návrat (Eds.): ADBIS 2002, LNCS 2435, pp. 135–148, 2002.

There exist different strategies which can lead to performance improvement of broadcast push [6,8]. The data are not always homogeneous and clients sometime are more interested in particular data elements. Therefore some data, more frequently accessed, are called "hot" and the other data, less frequently accessed, are called "cold". To deal with this kind of data the idea of broadcast disks was introduced [3,4,2]. Here the broadcast organized as a set of disks with different speeds. "Hot" data are placed on the "hot" (or "fast") disk and the "cold" (or "slow") data are placed on the "cold" disk. Hence if most of the data that client needs are "hot" it reduces the response time.

Another strategy capable to reduce the access time is client caching. However when data are being changed, there arises a problem how to keep the data cached in a client consistent with the updated data on the server [10,12,5]. Clearly, any invalidation method is prone to starvation of queries by update transactions. This same problem also exists in the context of broadcast push, even without client caching. Broadcasting is a form of a cache "on the air." In our previous work, we effectively addressed this problem by maintaining multiple versions of data items on the broadcast as well as in the client cache [9]. With multiple versions, more read-only transactions are successfully processed and commit in a similar manner as in traditional multiversion schemes, where older copies of items are kept for concurrency control purposes (e.g., [7]). The time overhead created by the multiple versions is smaller than the overall time lost for aborts and subsequent recoveries.

The performance (determined by the access time and power consumption) of multiversion broadcast is directly related to the issue of the size of the broadcast. Towards this we try to find ways to keep the size of broadcast as small as possible. There is no need to assume that all data have to be changed every time interval such that data values of adjacent versions are always different. Hence, we can reduce the communication traffic by not explicitly sending unchanged part of the older versions [11]. Consequently the client can retrieve the needed version of data sooner if the data do not change very often, which reduces the time during which the client stays on. We exploit this idea in the compression scheme we are proposing in this paper.

The main contributions of this paper are:

1. Identification of two different broadcast organizations for multiversion broadcast, namely *Vertical* and *Horizontal*.
2. Development of a compression scheme along the lines of *Run Length Encoding* (RLE) [11], applicable to both of the proposed broadcast organizations and which incurs no decompression overhead at the client.
3. Evaluation of circumstances under which each of our proposed broadcast organizations performs better.

The rest of the paper is structured as follows. In Section 2,we present the system model. Section 3 and 4 describe server side broadcast organization and client access behavior, respectively. Sections 5 presents our experimental platform whereas our experimental results are discussed in Section 6.