Cooperative Caching in a Mobile Environment

Say Ying Lim
Monash University, Australia

INTRODUCTION

There has been a rapid recent development in the usage of mobile devices, such as personal digital assistant (PDA), mobile notebooks and other mobile electronic devices. This has been increasingly in demand due to new smaller designs and easy to carry around features. The mobile computing technology has pushed the uses of mobile device even more by providing the ability for mobile users to access information anytime, anywhere. However, mobile computing environments have certain limitations, such as short battery power and limited storage and involving communication cost and bandwidth limitations (Kossman et al., 2001). Hence, it is of great interest to provide efficient query processing with quick response rate and with low transfer cost. Due to the inherent factors like low bandwidth and low reliability of wireless channels in the mobile computing environment, it is therefore important for a mobile client to cache its frequently accessed database items into its local storage (Chan, Si, & Leong, 2001).

Caching is a key strategy in improving data retrieval performance of mobile clients (Barbara & Imielinkski, 1994; Chow, Leong, & Chan, 2005). In order to retain the frequently accessed database items in the client's local memory, a caching mechanism is needed. By having the caching mechanisms, it allows a client to serve database queries at least partially during connection, which is an inherent constraint of the mobile environment. Thus, by having an effective caching mechanism in keeping the frequently accessed items, the more queries could be served in case of disconnection (Chan, Si, & Leong, 2001). The aim of caching is beneficial to the mobile environment, because having the data cached into the local memory can help future queries to be answered more quickly or to access the data faster, with low latency time and reduced start up delays that may be caused on the client side (Kara & Edwards, 2003). In addition to improving the access latency, it also helps to save power due to the ability to allow lower data transmission, as well as improvement in terms of data availability in situations of disconnection (Wu, Yu, & Chen, 1996; Lee, Xu, & Zheng, 2002).

In this article, we concentrate particularly on cooperative caching, which is basically a type of caching strategy that not only allows mobile clients to retrieve database items from the servers, but also from the cache in their peers.

BACKGROUND

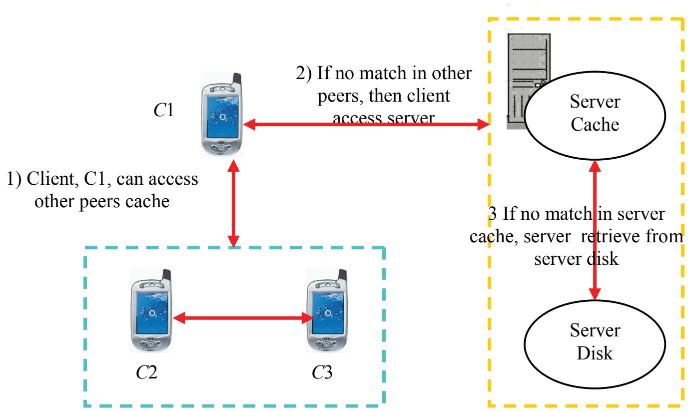
The effect of having the ability to cache data is of great importance, particularly in the mobile computing environment. This is due to the reason that contacting the remote servers for data is expensive in the wireless environment and the vulnerability to frequent disconnection can further increase the communication costs (Leong & Si, 1997). Also, caching the frequently accessed data in the mobile devices will help reduce tune-in time and power consumption because requested data can be fetched for the cache without tuning into the communication channel for retrieval (Lee & Lee, 1999). Caching has also been proven as a significant technique in improving the performance of data retrieval in peer to peer network in helping to save bandwidth for each data retrieval that are made by the mobile clients (Joseph et al., 2005). The cached data are meant to support disconnected or intermitted connected operations. There are many different types of caching strategies that serve the purpose to improve query response time and to reduce contention on narrow bandwidth (Zheng, Lee, & Lee, 2004).

Regardless of which caching strategy one chooses, they have their own advantages and disadvantages in some way. In principle, caching can improve system performance in two ways: (a) It can eliminate multiple requests for the same data and (b) improve performance by off-loading the work. The first way can be achieved by allowing mobile clients to share data among each other by allowing them to access each other's cache within a reasonably boundary between them. The second way, however, can be demonstrated as in an example of a mobile user who is interested in keeping stock prices and caches them into his mobile device. By having copies in his own mobile device, he can perform his own data analysis based on the cached data without communicating directly to the server over the wireless channel.

COOPERATIVE CACHING

The cooperative caching is a kind of information sharing that was developed by the heavy influence and emergence of the robust yet reliable peer-to-peer (P2P) technologies, which allows mobile clients (Kortuem et al., 2001). This type of information sharing among clients in a mobile

Figure 1. An overview of cooperative caching architecture



environment has generally allowed the clients to directly communicate among themselves by being able to share cached information through accessing data items from the cache in their neighboring peers rather than having to rely on their communication to the server for each query request (Chow, Leong, & Chan, 2004).

There are several distinctive and significant benefits that cooperative caching brings to a mobile computing environment. These include improving access latency, reducing servers' workloads and alleviating point–to-point channel congestion. Though the benefits outweigh the drawbacks, there is still a main concern that cooperative caching may produce. This refers to the possibility of the increase in the communication overhead among mobile clients (Ku, Zimmermann, & Wan, 2005)

Framework Design of Cooperative Caching in a Mobile Environment

Several clients and servers are connected together within a wireless channel in a mobile environment. Basically, the clients which denote the mobile hosts are connected to each other wirelessly within a certain boundary among each other, and they can exchange information by allowing each other to access the other peers' cache (Cao, Yin, & Das, 2004).

Figure 1 illustrates an example of the framework architecture design of the cooperative caching, which provides the ability for mobile clients to communicate among each other. If the client encounters a *local cache miss*, it will send a query to request, from its neighboring peers, to obtain a communication and the desired data from its peer's cache. Otherwise, it will be known as a *local cache hit* if the desired data exists in its local cache. As for trying to obtain data from its peers, if the desired data is available from its neighboring peers or if the other peers can turn in the requested data before it is broadcast on the channel, then it is known as a *global cache hit*; otherwise it is called a *global cache miss* and the client would have to wait for the desired data to arrive in the broadcast channel or access the server cache and, if that fails, then the server would retrieve the desired data from the disk (Sarkar & Hartman, 2001; Hara, 2002; Chow, Leong, & Chan, 2005).

As a summary, a mobile client can choose to either (a) retrieve data from the server directly by having a direct communication through issuing a query (Hu & Johnson, 2000) or (b) capture the data from a scalable broadcast channel (Su, Tassiulas, & Tsotras, 1999). These are known as a pull-base and push-base mechanism respectively. Further investigation on pull and push-based environments are made in the subsequent subsections.

Using Cooperative Caching in an On-Demand (Pull-Based) Environment

A pull-based environment refers to relating the use of traditional point–to-point scenario similar to client-server communication directly. It can also be known as an on demand query or server strategy whereby processing can be done on the server upon request sent by the mobile clients. Figure 2 illustrates an example of a pull-based architecture. It can be seen that the mobile client issues a direct query to the server over a dedicated channel to be processed. Processing takes