Caching and Replication in Mobile Data Management

Evaggelia Pitoura
Computer Science Department,
University of Ioannina, Greece
pitoura@cs.uoi.gr

Panos K. Chrysanthis
Department of Computer Science,
University of Pittsburgh, USA
panos@cs.pitt.edu

Abstract

Mobile data management has been an active area of research for the past fifteen years. Besides dealing with mobility itself, issues central in data management for mobile computing include low bandwidth, intermittent network connectivity and scarcity of resources with emphasis on power management. In this article, we focus on how caching and replication in mobile data management address these challenges. We consider two antagonistic criteria, that of ensuring quality of data in terms of consistency and coherency and that of achieving quality of service in terms of response time and availability.

1 Introduction

Mobile computing refers to computing using devices that are not attached to a specific location, but instead their position (network or geographical) changes. Mobile computing can be traced back to file systems and the need for disconnected operations in the end of the 80s. With the rapid growth in mobile technologies and the cost effectiveness in deploying wireless networks in the 90s, the goal of mobile computing was to support of AAA (anytime, anywhere and any-form) access to data by users from their portable computers and mobile phones, devices with small displays and limited resources. This led to research in mobile data management including transaction processing, query processing and data dissemination [22]. A key characteristic of all these research efforts was the great emphasis on the mobile computing challenges, including:

- *Intermittent Connectivity* This refers to the fact that computation must proceed despite short or long periods of network unavailability.
- *Scarcity of Resources* Due to the small sizes of portable devices, there are implicit restrictions in the availability of storage and computation capacity and mostly of energy.
- *Mobility* The implications of mobility are varying. First, mobility introduces a number of technical challenges at the networking layer. It also offers a number of opportunities at the higher layers for explicitly using location information either at the semantic level (for instance, for providing personalization) or at the performance level (for instance, for data prefetching). Finally, it amplifies heterogeneity.

In general, one can distinguish between single-hop and multi-hop underlying infrastructures. In *single-hop* infrastructures, each mobile device communicates with a stationary host, which corresponds to its point

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

of attachment to the network. The rest of the routing is the responsibility of a stationary infrastructure, e.g., the Internet. On the other hand, in *multi-hop* wireless communication, an ad-hoc wireless network is formed in which, wireless hosts participate in routing messages among each other. In both infrastructures, the hosts between the source (or sources) that holds the data and the requester of data (or data sink) form a dissemination tree. The hosts (mobile or stationary) that form the dissemination tree may store data and take part in the computation towards achieving *in network* processing (e.g., aggregation). Locally caching or replicating data at the wireless host or at the intermediate nodes of the dissemination tree are important for improving system performance and availability.

Caching and replication generally attempt to guarantee that most data requests are for data that is being held in main memory or local storage, negating the need to perform I/O, or a remote data retrieval. Hence, the use of appropriate caching/replication schemes have been traditionally used to improve performance and reduce service time. In mobile environments the performance considerations go beyond simple speedups and data retrieval delays. In this article, we examine how caching and replication has been utilized in mobile data management and more specifically in the first infrastructure where data are cached at the mobile device in order to avoid excessive energy consumption and to cope with intermittent connectivity.

In this paper, our focus is on consistency, that is, how to ensure the correctness of operations on cached data. In Section 2, we provide a brief taxonomy of related correctness criteria. While traditionally, cached data are read-only, in mobile computing, some restricted form of cache updates is supported especially in case of disconnections. We call this form of caching that allows cache updates at the client, *two-tier caching* and discuss it in detail in Section 3. In Section 4, we present issues related to disseminating updates from the rest of the network to the mobile device. Finally, Section 5 concludes the paper.

2 Consistency Levels

We consider the case in which a mobile computing device (such as a portable computer or cellular phone) is connected to the rest of the network typically through a wireless link. Wireless communication has a double impact on the mobile device since the limited bandwidth of wireless links increases the response times for accessing remote data from a mobile host and transmitting as well as receiving of data are high energy consumption operations.

The principal goal is to store appropriate pieces of data locally at the mobile device so that it can operate on its own data, thus reducing the need for communication that consumes both energy and bandwidth. At some point, operations performed at the mobile device must be synchronized with operations performed at other sites. The complexity of this synchronization depends greatly on whether updates are allowed at the mobile device. The main reason for allowing such updates is to sustain network disconnections. When there are no local updates, the important issue is disseminating updates from the rest of the network to the mobile device.

Synchronization depends on the level at which correctness is sought. This can be roughly categorized as replica-level correctness and transaction-level correctness. At the *replica level*, correctness or coherency requirements are expressed per item in terms of the allowable divergence among the values of the copies of each item. There are many ways to characterize the divergence among copies of an item. For example, with *quasi copies* [3], the coherency or freshness requirements between a cached copy of an item and its primary at the server are specified by limiting (a) the number of updates (versions) between them, (b) their distance in time, or (c) the difference between their values. At the *transaction level*, the strictest form of correctness is achieved through global serializability that requires the execution of all transactions running at mobile and stationary hosts to be equivalent to some serial execution of the same transactions. In case of replication, one copy serializability provides equivalence with a serial execution on a one-copy database. One-copy serializability does not allow any divergence among copies.

There is a large number of correctness criteria proposed besides serializability. A practical such criterion

is snapshot isolation [5]. With snapshot isolation, a transaction is allowed to read data from any database snapshot at a time earlier than its start time. Central are also criteria that treat read-only transactions, i.e., transactions with no update operations, differently. Consistency of read-only transactions is achieved by ensuring that transactions read a database instance that does not violate any integrity constraints (as for example with snapshot isolation), while freshness of read-only transactions refers to the freshness of the values read [11]. Finally, *relaxed-currency serializability* allows update transactions to read out-of-date values as long as they satisfy some freshness constraints specified by the users [6].

There are two basic ways of propagating updates. Eager replication synchronizes all copies of an item within a single transaction, whereas with lazy replication, transactions for keeping replica coherent run as separate, independent database transactions after the original transaction. One-copy serializability as well as other forms of correctness can be achieved either through eager or lazy update propagation.

3 Two-tier Caching

In this section, we assume that data can be updated at the mobile device. The main motivation is support for disconnected operation. *Disconnected operation* refers to the autonomous operation of a mobile client, when network connectivity becomes unavailable for instance, due to physical constraints, or undesirable, for example, for reducing power consumption. Preloading or prefetching data to sustain a forthcoming disconnection is often termed *hoarding*. The content of data to be prefetched may be determined (a) automatically by the system by utilizing implicit information, most often based on the past history of data references, or (b) by instructions given explicitly by the users, as in *profile-driven data prefetching* [7], where a simple profile language is introduced for specifying the items to be prefetched along with their relative importance. Additional information such as a set of allowable operations or a characterization of the required data quality may also be cached along with data. For example, in the *Pro-Motion infrastructure* [28, 15], the unit of caching and replication is a *compact*, an object that encapsulates the cached data, operations for accessing the cached data, state information (such as the number of accesses to the object), consistency rules that must be followed to guarantee consistency, and obligations (such as deadlines).

To allow concurrent operation at both the mobile client and other sites during disconnection, optimistic approaches to consistency control are typically deployed. Optimistic consistency maintenance protocols allow data to be accessed concurrently at multiple sites without a priori synchronization between the sites, potentially resulting in short term inconsistencies. Such protocols trade-off quality of data for improving quality of service. Optimistic replication has been extensively studied as a means for consistency maintenance in distributed systems (for example, see [23] for a thorough recent survey). In this paper, we present some representative examples of optimistic protocols in the context of mobile computing.

Consistent operation during disconnected operation has been also extensively addressed in the context of *network partitioning*. In this context, a network failure partitions the sites of a distributed database system into disconnected clusters. Various approaches have been proposed and are excellently reviewed in [8]. In general, protocols in network partition follow peer-to-peer models where transactions executed in any partition are of equal importance, whereas the related protocols in mobile computing most often consider transactions at the mobile host as second-class, for instance, by considering their updates as tentative. Furthermore, disconnections in mobile computing are common and some of them may be considered foreseeable.

Disconnections correspond to the extreme case of total lack of connectivity. Other connectivity constraints, such as weak or intermittent connectivity also affect the protocols for enforcing consistency. In general, weak connectivity is handled by appropriately revising those operations whose deployment involves the network. For instance, the frequency of propagation to the server of updates performed at the local data may depend on connectivity conditions.

In early research in mobile computing, a general concern has been whether issues such connectivity or

mobility should be transparent or hidden from the users. In this respect, adapting the levels of transaction or replica correctness to the system conditions such as the availability of connectivity or the quality of the network connection and providing applications with less than strict notions of correctness can be seen as making such conditions visible to the users. This is also achieved by explicitly extending queries with quality of data specifications for example for constraining the divergence between copies.

Some common characteristics of protocols for consistency in two-tier caching are:

- the propagation of updates performed at the mobile site follow in general lazy protocols,
- reads are allowed at the local data, while updates of local data are tentative in the sense that they need to be further validated before commitment.
- for integrating operations at the mobile hosts with transactions at other sites, in the case of replica-level consistency, copies of an item are reconciled following some conflict resolution protocol. At the transaction-level, local transactions are validated against some application or system level criterion. If the criterion is met, the transaction is committed. Otherwise, the execution of the transaction is either aborted, reconciled or compensated. Such actions may have cascaded effects on other tentatively committed transactions that have seen the results of the transaction.

Next, we present a number of consistency protocols that have been proposed for mobile computing.

Isolation-Only transactions in Coda Coda [24] is one of the first file systems designed to support disconnections and weak connectivity. Coda introduced *isolation-only* transactions (IOTs) [14] in file systems. An IOT is a sequence of file access operations. A transaction T is called a *first-class transaction*, if it does not have any partitioned file access, i.e., the mobile host maintains a connection for every file it has accessed. Otherwise, T is called a *second-class transaction*. Whereas the result of a first-class transaction is immediately committed, a second-class transaction remains in the pending state till connectivity is restored. The result of a second-class transaction is held within the local cache and visible only to subsequent accesses on the same host. Second-class transactions are guaranteed to be locally serializable among themselves. A first-class transaction is guaranteed to be serializable with all transactions that were previously resolved or committed at the fixed host. Upon reconnection, a second-class transaction T is validated against one of the following two serialization constraints. The first is global serializability, which means that if a pending transaction's local result were written to the fixed host as is, it would be serializable with all previously committed or resolved transactions. The second is a stronger consistency criterion called global certifiability (GC) which requires a pending transaction be globally serializable not only with, but also after all previously committed or resolved transactions.

Two-tier Replication With *two-tier replication* [12], replicated data have two versions at mobile nodes: master and tentative versions. A master version records the most recent value received while the site was connected. A tentative version records local updates. There are two types of transactions analogous to second- and first-class IOTs: tentative and base transactions. A *tentative transaction* works on local tentative data and produces tentative data. A *base transaction* works only on master data and produce master data. Base transactions involve only connected sites. Upon reconnection, tentative transactions are reprocessed as base transactions. If they fail to meet some application-specific acceptance criteria, they are aborted.

Two-Layer Transactions With two-layer transactions [17], transactions that run solely at the mobile host are called *weak*, while the rest are called *strict*. A distinction is drawn between weak copies and strict copies. In contrast to strict copies, weak copies are only tentatively committed and hold possibly obsolete values. Weak transactions update weak copies, while strict transactions access strict copies located at any site. Weak copies are integrated with strict copies either when connectivity improves or when an application-defined freshness limit to the allowable deviation among weak and strict copies is passed. Before reconciliation, the results of weak transactions are visible only to weak transactions at the same site. Strict transactions are slower than weak

transactions, since they involve the wireless link but guarantee permanence of updates and currency of reads. During disconnection, applications can only use weak transactions. In this case, weak transactions have similar semantics with second-class IOTs [14] and tentative transactions [12]. Adaptability is achieved by restricting the number of strict transactions depending on the available connectivity and by adjusting the application-defined degree of divergence among copies.

Bayou Bayou [27, 26, 16] is built on a peer-to-peer architecture with a number of replicated servers weakly connected to each other. Bayou does not support full-fledged transactions. A user application can read-any and write-any available copy. Writes are propagated to other servers during pair-wise contracts called *anti-entropy* sessions. When a write is accepted by a Bayou server, it is initially deemed tentative. As in two-tier replication [12], each server maintains two views of the database: a copy that only reflects committed data and another full copy that also reflects the tentative writes currently known to the server. Eventually, each write is committed using a primary-commit schema. That is, one server designated as the primary takes responsibility for committing updates. Because servers may receive writes from clients and other servers in different orders, servers may need to undo the effects of some previous tentative execution of a write operation and re-apply it. The Bayou system provides dependency checks for automatic conflict detection and merge procedures for resolution. Instead of transactions, Bayou supports *sessions*. A session is an abstraction for a sequence of read and write operations performed during the execution of an application. *Session guarantees* are enforced to avoid inconsistencies when accessing copies at different servers; for example, a session guarantee may be that read operations reflect previous writes or that writes are propagated after writes that logically precede them. Different degrees of connectivity are supported by individually selectable session guarantees, choices of committed or tentative data, and by placing an age parameter on reads. Arbitrary disconnections among Bayou's servers are also supported since Bayou relies only on pair-wise communication. Thus, groups of servers may be disconnected from the rest of the system yet remain connected to each other.

4 Update Dissemination

In this section, we consider data at the mobile device to be read-only, as in traditional client-server caching [9]. In this case, the main issue is developing efficient protocols for disseminating server updates to mobile clients. Most such cache invalidation protocols developed for mobile computing focus on the case in which a large number of clients is attached to a single server. Often, the server is equipped with an efficient broadcast facility that allows it to propagate updates to all of its clients. Different assumptions are made on whether the server maintains or not any information about which clients it is serving, what are the contents of their cache, and when their cache was last validated. Servers that hold such information are called *stateful*, while servers that do not are called *stateless*. Another issue pertinent to mobile computing is again handling disconnections, in particular, ensuring that cache invalidation are received by clients despite any temporary network unavailability.

Update propagation may be either synchronous or asynchronous. In *synchronous* methods, the server broadcasts an invalidation report periodically. A client must listen for the report first to decide whether its cache is valid or not. Thus, each client is confident for the validity of its cache only as of the last invalidation report. This adds some latency to query processing, since to answer a query, a client has to wait for the next invalidation report. In case of disconnections, synchronous methods surpass asynchronous in that clients need only periodically tune in to read the invalidation report instead of continuously listening to the broadcast. However, if the client remains inactive longer than the period of the broadcast, the entire cache must be discarded, unless special checking is deployed.

Invalidation protocols vary in the type of information they convey to the clients. In case of replica level correctness, it suffices that single read operations access current data. In this case, invalidation may include just a list of the updated items or in addition to this, their updated values. Including the updated values may be wasteful of bandwidth especially when the corresponding items are cached at only a few clients. On the other

hand, if the values are not included, the client must either discard the item from its cache or communicate with the server to receive the updated value. The reports can provide information for individual items or aggregate information for sets of items. In case of transaction level correctness, invalidation reports must include additional information regarding server transactions.

The efficiency of an update dissemination protocol depends on the connectivity behavior of the mobile clients. In [4], clients that are often connected are called workaholic, while clients that are often disconnected are the sleepers. Three synchronous strategies for stateless servers are considered. In the broadcasting *timestamps* strategy (TS), the invalidation report contains the timestamps of the latest change for items that have had updates in the last w seconds. In the *amnestic terminals* strategy (AT), the server only broadcasts the identifiers of the items that changed since the last invalidation report. In the *signatures* strategy, signatures are broadcast. A signature is a checksum computed over the value of a number of items by applying data compression techniques similar to those used for file comparison. Each of these strategies is shown to be effective for different types of clients. Signatures are best for long sleepers, that is, when the period of disconnection is long and hard to predict. The AT method is best for workaholic. Finally, TS is shown to be advantageous when the rate of queries is greater than the rate of updates provided that the clients are not workaholics.

Another model of operation in the content of mobile databases is that of a broadcast or push model [1]. In this model, the server broadcasts (periodically) data to a set of mobile clients. Clients monitor the broadcast and retrieve the data items they need as they arrive. Data of interest may also be cached locally at the client.

When clients read data from the broadcast, a number of different replica-level correctness models are reasonable [2]. For example, if clients do not cache data, the server always broadcasts the most recent values, and there is no backchannel for on-demand data delivery, then the *latest value* model is a model that arise naturally. In this model, clients read the most recent value of a data item. Methods for enforcing transaction-level correctness are presented in [19]. With the *invalidation* method, the server broadcasts an invalidation report with the data items that have been updated since the broadcast of the previous report. Transactions that read obsolete items are aborted. With the *serialization graph testing* (SGT) method, the server broadcasts control information related to conflicting operations. Clients use this information to ensure that their read-only transactions are serializable with the server transactions. With *multiversion broadcast* [20, 18], multiple versions of each item are broadcast, so that client transactions always read a consistent database snapshot.

A general theory of correctness for broadcast databases as well as the fundamental properties underlying the techniques for enforcing it are given in [21]. Correctness characterizes the freshness of the values seen by the clients with regards to the values at the server as well as the temporal discrepancy among the values read by the same transaction.

More recently, the concept of materialized views was extended in the context of mobile databases to operate in a fashion similar to data caches supporting local query processing [13]. As in traditional databases, materialized views in mobile databases provide a means to present different portions of the databases based on users' perspectives and, similar to data warehouses, materialized views provide a mean to support personalized information gathering from multiple data sources. Personalization is expressed in the form of view maintenance options for recomputation and incremental maintenance. They offer a finer grain of control and balance between data availability and currency, the amount of wireless communication and the cost of maintaining consistency. In order to better characterize these personalizations, in [13] *recomputational consistency* was introduced and the *materialized view consistency* [30] was enhanced with new levels which correspond to specific view currency customizations.

5 Summary

In this short article, we presented issues related to cache consistency in mobile computing. The methods presented can be considered as extensions of traditional client-server caching, where the client is a mobile device.

The main motivation for this form of caching is improving availability especially in the case of network disconnections. Caching also improves performance through reducing the communication overhead in terms of both data access delays and energy consumption.

An interesting extension of the current methods is hierarchical caching for the emerging infrastructures of multi-hop wireless networks. Besides the challenges due to mobility, hierarchical caching introduces new complications such as the multiple levels of intervening caches can that create adverse workloads for the caching schemes used at different levels. A hierarchical caching scheme must have the ability to adapt itself, thereby acting synergistically and cooperatively with other caching schemes on mobile peers [10].

Finally, mobile computing is often related to wireless computing and to computation involving small devices, including sensors or RFID tags. In the case of sensors, their limited power restricts the amount of processing and communication that sensors can perform before they become inactive. Thereby, an interesting question related to hierarchical caching is how caching at different sensors can help in the conservation of their energy, thereby prolonging the lifetime of a sensor network and improving the quality of the data [25, 29].

References

[1] S. Acharya, M. J. Franklin, and A. B. Zdonik. Balancing push and pull for data broadcast. In *SIGMOD Conference*, pages 183–194, 1997.

[2] S. Acharya, M. J. Franklin, and S. B. Zdonik. Disseminating updates on broadcast disks. In *VLDB*, pages 354–365, 1996.

[3] R. Alonso, D. Barbara, and H. Garcia-Molina. Data Caching Issues in an Information Retrieval System. *ACM Transactions on Database Systems (TODS)*, 15(3):359–384, September 1990.

[4] D. Barbará and T. Imielinski. Sleepers and workaholics: Caching strategies in mobile environments. *VLDB J.*, 4(4):567–602, 1995.

[5] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil., and P. O'Neil. A Critique of ANSI SQL Isolation Levels. In *Proceedings of the ACM SIGMOD Conference*, pages 1–10, 1995.

[6] P. A. Bernstein, A. Fekete, H. Guo, R. Ramakrishnan, and P. Tamma. Relaxed-currency serializability for middle-tier caching and replication. In *SIGMOD Conference*, pages 599–610, 2006.

[7] M. Cherniack, E. F. Galvez, M. J. Franklin, and S. B. Zdonik. Profile-Driven Cache Management. In *Proceedings of the ICDE Conference*, pages 645–656, 2003.

[8] S. B. Davidson, H. Garcia-Molina, and D. Skeen. Consistency in Partitioned Networks. *ACM Computing Surveys*, 17(3):341–370, September 1985.

[9] M. J. Franklin, M. J. Carey, and M. Livny. Transactional client-server cache consistency: Alternatives and performance. *ACM Trans. Database Syst.*, 22(3):315–363, 1997.

[10] P. K. Chrysanthis G. Santhanakrishnan, A. Amer. Towards universal mobile caching. In *Proc. of the 4th ACM Int'l Workshop on Data Engineering for Wireless and Mobile Access*, June 2005.

[11] H. Garcia-Molina and G. Wiederhold. Read-Only Transactions in a Distributed Database. *ACM Transactions on Database Systems*, 7(2):209–234, June 1982.

[12] J. Gray, P. Helland, P. O' Neil, and D. Shasha. The Dangers of Replication and a Solution. In *Proceedings of the ACM SIGMOD Conference*, pages 173–182, Montreal, Canada, 1996.

[13] S. Weissman Lauzac and P. K. Chrysanthis. Personalizing Information Gathering for Mobile Database Clients. In *Proceedings of the ACM Annual Symposium on Applied Computing*, pages 49–56, 2002.

[14] Q. Lu and M. Satyanarayanan. Improving Data Consistency in Mobile Computing Using Isolation-Only Transactions. In *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems*, Orcas Island, Washington, May 1995.

[15] S. Mazumdar and P. K. Chrysanthis. Localization of Integrity Constraints in Mobile Databases and Specification in PRO-MOTION. *ACM Mobile Networks*, 9(5):481–490, October 2004.

[16] K. Petersen, M. Spreitzer, D. B. Terry, M. Theimer, and A. J. Demers. Flexible update propagation for weakly consistent replication. In *SOSP*, pages 288–301, 1997.

[17] E. Pitoura and B. Bhargava. Data Consistency in Intermittently Connected Distributed Systems. *IEEE Transactions on Knowledge and Data Engineering*, 11(6):896–915, November 1999.

[18] E. Pitoura and P. K. Chrysanthis. Exploiting versions for handling updates in broadcast disks. In *VLDB*, pages 114–125, 1999.

[19] E. Pitoura and P. K. Chrysanthis. Scalable processing of read-only transactions in broadcast push. In *ICDCS*, pages 432–439, 1999.

[20] E. Pitoura and P. K. Chrysanthis. Multiversion data broadcast. *IEEE Trans. Computers*, 51(10):1224–1230, 2002.

[21] E. Pitoura, P. K. Chrysanthis, and K. Ramamritham. Characterizing the temporal and semantic coherency of broadcast-based data dissemination. In *ICDT*, pages 410–424, 2003.

[22] E. Pitoura and G. Samaras. *Data Management for Mobile Computing*. Kluwer Academic Publishers, 1998.

[23] Y. Saito and M. Shapiro. Optimistic Replication. *ACM Computing Surveys*, 37(1):42–81, March 2005.

[24] M. Satyanarayanan. The evolution of coda. *ACM Trans. Comput. Syst.*, 20(2):85–124, 2002.

[25] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. Balancing energy efficiency and quality of aggregate data in sensor networks. pages 13(4): 384 – 403, December 2004.

[26] D. Terry, A. Demers, K. Petersen, M. Spreitzer, M. Theimer, and B. Welch. Session Guarantees for Weakly Consistent Replicated Data. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, pages 140–149, September 1994.

[27] D. B. Terry, M. M Theimer, K. Petersen, A. J. Demers, M. J Spreitzer, and C. H. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, December 1995.

[28] G. Walborn and P. K. Chrysanthis. PRO-MOTION: Support for Mobile Database Access. *Personal and Ubiquitous Computing*, 1(3), September 1997.

[29] D. Zeinalipour-Yazti, P. Andreou, P. K. Chrysanthis, and G. Samaras. Mint views: Materialized in-network top-k views in sensor networks. In *Proceedings of the 7th International Conference in Mobile Data Management*, pages 182–189, May 2007.

[30] Y. Zhuge, H. Garcia-Molina, and J. Wiene. Consistency Algorithms for Multi-Source Warehouse View Maintenance. *Distributed and Parallel Databases*, 4(4), 1997.