

An Adaptive AVI-based Cache Invalidation Scheme for Mobile Computing Systems*

Joe Chun-Hung Yuen, Edward Chan, Kam-Yiu Lam, H. W. Leung
Department of Computer Science,
City University of Hong Kong
{csjyuen, csedchan, cskylam}@cityu.edu.hk

Abstract

In a mobile computing system, caching data items at the mobile clients is important to reduce the data access delay in a unreliable and low bandwidth mobile network. However, efficient methods must be used to ensure the coherence between the cached items and the data items at the database server. In this paper, by exploring the real-time properties of the data items, we propose a cache invalidation scheme named as Invalidation by Absolute Validity Interval (IAVI). We define an absolute validity interval (AVI), for each data item based on its real-time property, e.g., update interval. A mobile client can verify the validity of a cached item by comparing the last update time and its AVI. A cached item is invalidated if the current time is greater than the last update time by its AVI. With this self-invalidation mechanism, the IAVI scheme uses the invalidation report to inform the mobile clients about the change of AVI rather than the update event of the data item. As a result, the size of invalidation report can be reduced significantly. Performance studies show that the IAVI scheme can significantly reduce the mean response time and invalidation report size under various system parameters.

Keywords: mobile computing, cache invalidation, real-time data and cache coherency

1. Introduction

As the importance of mobile computing becomes more apparent, there is an explosive growth of research in this area. Until recently much of the work has been at the infrastructure level such as transmission mechanisms and multiple access protocols. Lately, however, there is an increasing interest in the upper layer issues such as effective dissemination of data items from the database server using both push and pull technology over mobile networks [1, 6], as well as the role of data caching at the clients, which is the

theme of this paper. In particular, maintaining coherence between the cached data items in the client and the data items at the database server has emerged as a major issue in the design of practical mobile computing systems as this can greatly improve the system response and the usefulness of the information.

In this paper, we propose another cache invalidation scheme, called *Invalidation by Absolute Validity Interval (IAVI)*, which makes use of the fact that the values of most of the data items in mobile computing systems have real-time properties, which in many cases may change according to a pre-defined pattern which can be approximated. Based on this property of a data item, we can define an *Absolute Validity Interval (AVI)* to estimate the life-span of the data value for a data item. In IAVI, by exploiting this real-time property, it is possible to reduce the size and frequency of invalidation messages required to maintain the coherence of the cached data items.

This paper is organised as follows. Section 2 covers related work in cache invalidation for mobile systems. We formally define the notion of Absolute Validity Interval for cache invalidation in Section 3. In Section 4 the IAVI cache invalidation scheme is proposed, followed by simulation experiments for studying its performance in Section 5. The conclusion of the paper is in Section 6.

2. Related Work

Caching frequently accessed data items on the client side has been recognised as an important technique to reduce access delay and network traffic in a limited bandwidth mobile environment. Most of the studies on cache coherency schemes for mobile environments are based on periodic broadcast of invalidation reports. For instance, Barbara and Imielinski, in one of the earliest work in this area, proposed three different variants of this approach – Broadcasting Timestamp (TS), Amnesic Terminals (AT) and Signatures (SIG) – depending on the expected duration of network disconnection [2]. However, the algorithms are only effective if the clients have not been disconnected for a

* This research is supported in part by UGC grant # 9040431

period exceeding an algorithm specific parameter. Otherwise the entire cache has to be discarded even though some of the cached data items might still be valid.

Jing et. al. proposed a bit-sequence scheme (BS) in which the invalidation report consists of bit sequences associated with a set of timestamps [4]. This approach has the drawback of greater complexity and much larger invalidation reports than the TS or AT methods, particularly when the number of data items is larger. More recently, a family of adaptive cache invalidation algorithms is proposed [7]. The essence of these algorithms is that the type of invalidation report to be sent (i.e TS or BS) is determined dynamically based on system status such as disconnection frequency and duration as well as update and query pattern.

There is also a lot of research in Web-based cache coherence, but these studies are confined primarily to Web-specific protocols and in wired networks [3].

3. The Absolute Invalidity Interval (AVI) of a Data Item

One important characteristic of the data items in the database of a mobile computing system is that they often represent the current status of the objects in the external environment, whose value may change quite rapidly. Examples are news updates, the latest market prices of stocks and the highway traffic conditions. Due to the real-time properties of the data items, updates, which are captured by some external devices or obtained from data vendors, are required to maintain the validity of the data values. It is assumed that stale (invalid) data values are much less useful. Normally, there are two types of update arrival patterns: periodic and sporadic. For example, the arrival of the most current price of a stock is sporadic. For some data items, it may not be necessary to track all changes in values. It might be sufficient to only sample the changes to the status of the actual object periodically (an example is road traffic conditions)

Based on the real-time properties (update arrival rates or update intervals) of a data item, we can assign a validity interval to the data item to characterize how rapidly the data value will change, e.g., the average life-span of a data value. We call this assigned validity interval the *Absolute Validity Interval (AVI)*¹. For certain types of data items, such as weather forecast, a larger AVI can be safely assigned, while for real-time stock quotes a shorter AVI on the order of seconds will be needed. Although the updates for some data items are sporadic, it is assumed that it is still possible to define an approximate update interval value for them. The AVI of a data item is not necessarily equal to the actual life-span of the values of the data item

(in fact, this is not possible for sporadic updates), it only needs to be a reasonably good approximation to the average life-span of the values of the data item. The AVI of a data item can be derived based on the previous update intervals of the data item. It is assumed that each update is associated with a time-stamp, which is its generation time. After the completion of the update, the time-stamp will be recorded with the updated data item. The time-stamp together with the AVI of the data item can be used to determine its validity.

The AVI of a data item can be used also as an estimator for the validity of a data value at client caches. For a data item cached at the clients, the data item has to be updated from time to time by the server in order to maintain consistency between the two. The update intervals can be used to define the validity periods of a cached item. A data item is updated at the beginning of each update interval. Hence, it is valid from the time it was cached until the time of the next update. In other words, as shown in Figure 1(a) the time from the n^{th} update on the data item to the time of the $(n+1)^{\text{th}}$ update is the validity period of the data item after n^{th} update. The value from the n^{th} update is stale after the arrival of the $(n+1)^{\text{th}}$ update.

According to the above model, the validity period of a data item is only known after the next update arrives, i.e. the validity period of a data item after the $(n-1)^{\text{th}}$ update is not defined until the n^{th} update has occurred. The differences between the validity period and AVI are shown in Figure 1(b). We define the *False Valid Period (FVP)* as the time period where the AVI overestimates the validity period of the data item and the *False Invalid Period (FIP)* as the time period where AVI underestimates the validity period. If FVP is greater than zero, the actual update interval of the data item is shorter than expected. The values of FVP should be kept small, since during FVP a client will consider an invalid data to be valid. On the contrary, if FIP is greater than zero, the actual update interval is longer than expected. A client will consider a cached item to be invalid during that period even though it is still valid. By suitably specifying the AVI based on the update intervals, the values of FVP and FIP can be kept to rather small values.

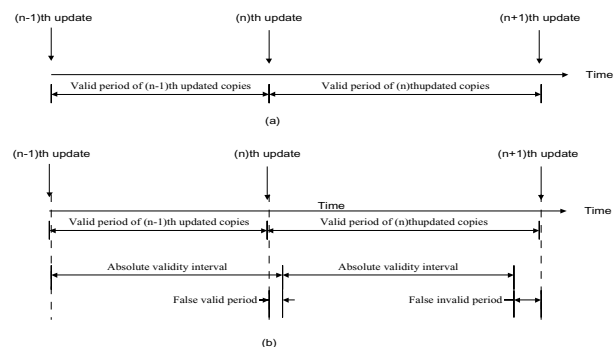


Figure 1 (a) Validity period, (b) AVI model

¹ We have previously used a simpler notion of AVI in improved broadcast scheduling and cache hit rate for mobile systems [5], without however explicitly exploring its usage in cache invalidation.

4. AVI in Cache Invalidation

We have defined AVI and validity period of a data item at the client cache in the previous section. Now we proceed to describe how they can be used to support an efficient cache invalidation scheme, which we call *Invalidation by Absolute Validity Interval* (IAVI).

In our model, since a cached item is assumed to be invalid if its AVI has expired, no explicit invalidation notification is needed to invalidate the cached items in the mobile clients. In the other words, a client can invalidate its cached items by calculating the items' last update times and the AVI of the data items.

As explained in the previous section, the arrival of some actual data items can be sporadic, and the optimal AVI value of a data item may vary with time. This change in the AVI of a data item will either shorten its FVP or enlarge its FIP. First, we consider the case that the new AVI of a data item is smaller than the previous one. In this case, the data item should be invalidated before its AVI expires. If the client uses the previous AVI, the FVP will be longer than the previous estimation. Changes in the AVI is typically caused by an update of the data item, and hence the cached item must be invalidated.

The second case is for a data item whose new AVI value is longer than the previous one. In this case no notification message is needed, since the cached item will be invalidated automatically when its AVI expires. The FIP in this case will be increased although this is still not desirable. Although it is possible to send explicit notifications to mobile clients on the new AVI value, this must be done before the AVI of the cached item expires. Mobile clients experiences frequent link disconnection, and it is hard to verify the new AVI to the current cached copy of a mobile client who has been disconnected.

Thus, we conclude that invalidation report is needed to notify the clients when the AVI value of a data item is reduced but not when it is increased. When the AVI of a data item is reduced, notification is needed to inform the mobile client in order to minimize the FVP. Otherwise the client might use invalid copy of the data in its cache. However, for the case where the AVI of a data item has increased, it is best to do nothing and let the next update of the data item refresh the cache. Suppose the server sends a report to inform the clients of the new AVI i.e. the cached item is still valid. Now if after a short time the data item is updated before the new AVI expires, an additional invalidation message will have to be sent. All this will result in substantial increase in overhead. Moreover, all the additional effort is wasted if the cached data item is not accessed after the original AVI has expired. It is preferable to accept a larger FIP than trying to update the AVI, and let the client make an explicit request when the data is actually accessed.

We will now discuss in details how the IAVI scheme is implemented at both the server and the client side.

4.1 Server Algorithm

The server algorithm consists of two parts, invalidation report generation and AVI adjustment. The former deals with the selection of the information to be included in the invalidation report. Since the broadcast bandwidth is a valuable resource that is shared among large number of mobile clients, it must be used efficiently. AVI adjustment is concerned with adapting the AVI values of the data items to achieve the desired level of the cache coherence for the system.

In order to notify the mobile clients about the changes in AVI values, invalidation reports are generated and disseminated periodically. The invalidation report contains a data ID and its update time, whose update interval must satisfy the following expression:

$$T_{\text{update}(i,n)} - T_{\text{update}(i,n-1)} < AVI_{(i)} \times (1 - F_i) \quad (1)$$

where $T_{\text{update}(i,n)}$ is the timestamp of n^{th} update on data item i ;

$AVI_{(i)}$ is the AVI of data item i and

F_i is the AVI tolerance for data item i

According to the above expression, if the update interval of data item i is longer than its AVI plus its AVI tolerance, it will be included in the invalidation report as this implies that the life-span of a data value is shorter than expected. The AVI tolerance is proposed to tackle the randomness in update interval. Since it is not possible to predict the occurrence of update events, the update interval and AVI will not be perfectly matched. In other words, the current AVI is valid if the difference between the update interval and the AVI of a data item is small. The tolerance limit is data dependent, and different kinds of data items may have different degrees of tolerance. However, for data items with the same AVI tolerance, the one with a shorter AVI will have a smaller tolerance limit according to the above expression. This is because a shorter AVI implies that the data item is updated frequently and the value is relatively dynamic, and hence a tighter tolerance is desirable to minimize the FVP.

4.2 Client Algorithm

Cache invalidation on client side is divided into two parts, implicit invalidation and explicit invalidation. In implicit invalidation, a cached item is invalidated by the expiration of its AVI. This occurs when the cached item is accessed and the AVI is found to have expired. In contrast,

explicit invalidation is caused by the receipt of an invalidation report from the server.

4.2.1 Implicit Invalidation. The validity of a cached data item is determined by the update time (the time stamp of data item) and the AVI value. When a cached item is referenced by a transaction from the mobile client, the transaction will examine the update time and AVI value of the item. If the sum of the update date time and the AVI value is smaller than the current time, the item is assumed to be invalid. (Note that it may be valid but the transaction does not know, which is the case during the FIP).

By using implicit invalidation, traffic cost for invalidation is minimized since a client can invalidate its cached items using the method described above without waiting for additional information from the server. Moreover, when the client reconnects to the mobile network after disconnection, it can invalidate its cached copy without waiting for the next invalidation report from mobile server. If the AVI has expired for a cached item, it can be invalidated without extra verification. Note, however, that if the cached copy seems to be valid based on its AVI, the client needs to reference the first invalidation after reconnection report to determine the validity of its cached items.

4.2.2. Explicit Invalidation. Besides implicit invalidation, a cached item can also be invalidated explicitly by invalidation reports sent from the server. If the update interval of a data item violates the AVI assumption (Equation 1), its ID will be included in the invalidation report and broadcast to the mobile clients. Note that in the explicit invalidation scheme, the size of invalidation report is much smaller than other techniques such as Bit Sequence (BS) and timestamp (TS). Unlike BS and TS, which include every update event in invalidation reports within certain period or interval, the invalidation report of the AVI scheme only contains the entries for the data items whose AVIs have been modified and meet the condition defined in equation (1). Moreover, if the AVI and the update interval of the data items are reasonably well matched, many of the update events do not generate explicit invalidation reports. This major advantage will be verified in the next section by our simulation studies.

5 Performance Study

In this section, the performance of different cache invalidation schemes is compared. We compare the performance of IAVI with two efficient schemes, Bit-Sequence (BS) [4] and Timestamp (TS) [2]. The simulation program is developed using CSIM and the Generic Mobile System Simulation Package [8]. Besides the above three schemes, an idealized cache invalidation scheme called *Perfect Server* (PS) is also developed for

comparison purposes. In PS, it is assumed that the system has full knowledge of the content of all the mobile clients' caches. As a result, the invalidation reports generated by PS will only contain the update information of the data items cached in the mobile client's caches, thus releasing more broadcast bandwidth for data dissemination. Such a scheme would be too costly to implement in practice as it requires a constant flow of updates to the mobile server regarding the content of mobile client caches.

5.1 Simulation Parameters

The baseline setting of the system parameters used in the simulation are listed below.

Table 1 Database

Database size	100000 items
Items size	256 bits

Table 2 Update process

Update interval	3600 sec.
Update interval variance	-10%~+10% uniform
Grp. 1 Update range/ relative interval	0~4999 / 1.0*
Grp. 2 Update range/ relative interval	5000~14999 / 2.0
Grp. 3 Update range/ relative interval	15000~29999 / 3.0
Grp. 4 Update range/ relative interval	30000~ Database size / 4.0

* The relative update interval is a multiple of the parameter *update interval*.

Table 3 Communication network

Broadcast bandwidth	10000 bps
Up-link capacity	1000 bps

Table 4 Mobile server

Broadcast cycle	30 sec. per cycle
Invalidation report	1 per cycle
Invalidation report window size	10 cycle
AVI Tolerance	0.1

Table 5 Mobile client

Number of mobile clients	100
Disconnect probability	0.1
Mean disconnect interval	4000 sec
Access hot spot / access probability	100 items / 0.8
Mean think time	100 sec
Mean query length	10 data items
Cache size	50 items
Cache Replacement Scheme	Least recently used (LRU)

5.2 Performance Evaluation

The simulation experiments presented in this section compare the performance of IAVI with the other cache invalidation approaches under different server and client settings. We have run a large number of simulations. Due to the limitation of space we will only describe the most important cases below; the full results can be found in [9].

Fig. 2 shows the impact of database size on the performance of the different cache invalidation schemes. by varying the number of data items in the database. It is clear that the response times for the PS and IAVI schemes are much lower than TS and B and are not much affected by the changes in database size. The performance of BS is significantly affected by the size of the database. When the

database size is very large, it is even worse than TS whose performance is consistently worse than PS and IAVI. This is due to the fact that the size of invalidation report increases with the size of the database according to the formula $2N + \log 2N \times \text{Time_bit_size}$, where N is the size of the database and Time_bit_size is the number of bits needed to represent update time [4]. On the other hand, the invalidation report sizes of TS, IAVI and PS depend on the update volume (update rate \times number of records per update) and are less sensitive to database size.

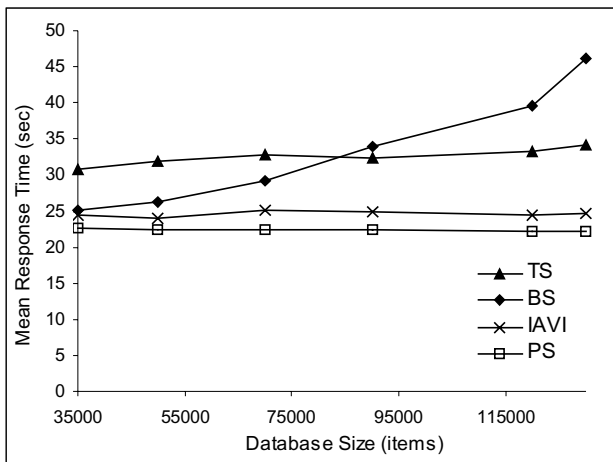


Fig. 2 Mean Response Times Vs Database Size

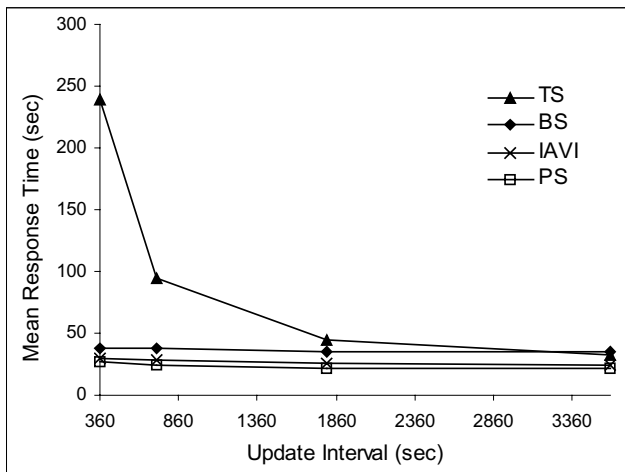


Fig. 3 Mean Response Time Vs Update Interval

Fig. 3 shows the mean response time and invalidation report size at different update intervals. The performance of TS decreases dramatically with an increase in update interval while other schemes are much less sensitive to the changes in update interval. This is because the invalidation report of TS includes all update records within the window frame, the invalidation reports size is greatly affected by update interval. However, IAVI only records the update records that violate their AVI value and hence is less affected by changes in update interval.

6 Conclusions

In this paper, we have proposed a scheme for the cache invalidation named as IAVI. We define an Absolute Validity Interval (AVI) for each data item and use this property to self-invalidate items in the client cache. When a mobile client accesses a cached item, the update timestamp and AVI of the data item can verify the validity of the item. The cached item is invalidated if the access time is greater than the last update time by its AVI. With this self-invalidation mechanism, IAVI uses the invalidation report to inform the mobile client about changes in AVI rather than the individual update events of the data item, and hence reduces the size of invalidation reports. Performance studies obtained by simulation shows that the IAVI scheme can significantly reduce the mean response time under various system parameters.

The current work does consider the characteristics of Web-based protocols. With rapid growth in mobile Internet, it will be useful to apply our models and techniques to mobile Web-based system and see if we can obtain similar performance improvements.

References

- [1] Acharya, S., Alonso, R., Franklin, M., Zdonik, S., "Broadcast Disk: Data Management for Asymmetric Communication Environments", *Proc. ACM SIGMOD*, May 1995.
- [2] Barbara, D. and Imielinski, T., "Sleepers and Workaholics: Caching Strategies in Mobile Environments", *Proceedings of ACM SIGMOD*, 1994.
- [3] Cao, P. and Chengjie, L., "Maintaining Strong Cache Consistency in the World Wide Web", *IEEE Trans. Computers*, Vol. 47 No. 4, April 1998.
- [4] Jing, J., Elmargamid, A., Helal, A. and Alonso, F., "Bit-Sequences: an Adaptive Cache Invalidation Method in Mobile Client/Server Environments", *Technical Report CSD-TR-94-074*, Purdue University, May 1995.
- [5] Lam, K. Y., Chan, E. and Yuen, C.H., "Broadcast Strategies to Maintain Cached Data for Mobile Computing Systems", *Proc. Mobile Data Access Workshop*, Singapore, Nov 1998.
- [6] Fernandez, J., and Ramamritham K., "Adaptive Dissemination of Data in Real-Time Asymmetric Communication Environments", Technical Report, Computer Science Dept., University of Massachusetts, 1998.
- [7] Hu, Q. and Lee, D.L., "Adaptive Cache Invalidation Methods in Mobile Environments", *Proc. 6th IEEE Intl. Symp. on High Performance Distributed Computing Environments*, 1997.
- [8] Yuen, C. H., "Data Dissemination Strategies for Mobile Computing Environments", M. Phil. Thesis, City University of Hong Kong, 1999.
- [9] Yuen, C. H., Chan, E., Lam, K.Y., "Performance Evaluation of an Adaptive AVI-based Cache Invalidation Scheme for Mobile Computing Systems", Technical Report, City University of Hong Kong, 2000.