Cache Invalidation Scheme for Mobile Computing Systems with Real-time Data*

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

Abstract

In this paper, we propose a cache invalidation scheme called Invalidation by Absolute Validity Interval (IAVI) for mobile computing systems. In IAVI, we define an absolute validity interval (AVI), for each data item based on its dynamic property such as the 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 plus its AVI. With this self-invalidation mechanism, the IAVI scheme uses the invalidation report to inform the mobile clients about changes in AVIs rather than the update event of the data items. As a result, the size of the invalidation report can be reduced significantly. Through extensive simulation experiments, we have found that the performance of the IVAI scheme is significantly better than other methods such as bit sequence and timestamp.

1 Introduction

Recent advances in mobile communication technology have greatly increased the functionality of mobile information services. An important application of mobile computing systems is to provide various types of real-time information such as stock quotes, weather conditions and traffic information, to mobile clients [SLR99]. A number of efficient data dissemination strategies have been proposed in recent years. Amongst the proposed methods, most are based on data broadcast as it is very cost effective in disseminating a substantial amount of information to a large number of mobile clients [AAFZ95, DCKV97].

In a mobile computing system, in order to reduce the data access delay, some data items are cached at the client machines. Thus, efficient cache invalidation methods are critical to the whole system performance [BI94, HL97, JEHL95, WYC96]. One of the most important considerations in the design of cache invalidation scheme is to minimise the required bandwidth for broadcasting the invalidation reports as well as the size of the invalidation reports. Various efficient methods have been proposed for preparing the cache invalidation reports such as the bit sequence and time-stamp methods [BI94, JEHL95]. In this paper, by exploring the fact that the values of many data items in mobile computing systems have real-time properties, we propose a cache invalidation scheme, called *Invalidation by Absolute Validity Interval (IVAI)*. In the scheme, we define an *Absolute Validity Interval (AVI)* to estimate the life-span of the data value for a data item. By estimating the AVI values of the data items, it is possible to reduce the size and frequency of invalidation messages required to maintaining the coherence of the cached data items. Extensive simulation experiments have been performed to compare IVAI with other cache invalidation schemes, and the results show that IAVI gives the best performance in most scenarios.

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. 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 [BI94]. However, the algorithms are only effective if the clients have not been disconnected for a period exceeding an algorithm specific parameter. Otherwise the entire cache has be 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 [JEHL95]. Using the information embedded in the bit sequences, a client needs only to invalidate its entire cache if more than half of the data items have been updated in the server since its last invalidation time. This ingenious approach has the drawback of greater complexity and much larger

* The work described in this paper was partially supported by a grant from the Research Grants Council of Hong Knog SAR, China [Project No. CityU1097/99E] and a grant from CityU (Project No. 7001069).

invalidation reports than the TS and AT methods, particularly when the number of data items is large.

Wu et. al. [WYC96] have modified the TS and AT algorithms to include cache validity check after reconnection. The proposed algorithms however do not solve the basic problem of TS in that the entire cache will have to be dumped if the disconnection time is greater than its update history window [WYC96]. More recently, a family of adaptive cache invalidation algorithms is proposed [HL97]. The essence of these algorithms is that the type of invalidation report to be sent is determined dynamically based on system status such as disconnection frequency and duration as well as update and query pattern.

3 Absolute Validity Interval (AVI)

An important characteristic of the data items in a mobile computing system is that they possess different degrees of real-time properties, e.g., they often represent the current status of the objects in the external environment, whose value may change quite rapidly [SLR99]. Examples include news updates and the latest market prices of stocks. 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. Usually, stale (invalid) data values are much less useful.

Basically, 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 the changes in values. It might be sufficient to only sample the changes to the status of the actual objects periodically (an example is road traffic conditions)

Based on the real-time properties 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). For some 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 assigned AVI value 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. The AVI value in this case only needs to be a reasonably good approximation to the average life-span of the values of the data item. In practice, 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 also be used as an estimator for the validity of a data value at a client cache. A data item cached at a client has to be updated from time to time by the server in order to maintain coherency 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 $(n+1)^{th}$ 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 $(n+1)^{th}$ update.

Since 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 $(n-1)^{th}$ 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 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 actually still valid. By suitably adjusting the AVI based on the update intervals, the values of FVP and FIP can be kept rather small.

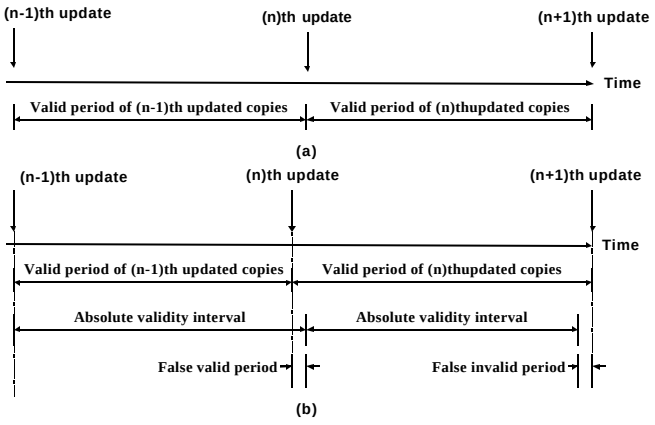

Figure 1 (a) Validity period, (b) AVI model

4 Cache Invalidation by AVI

4.1 The Principles

We have defined AVI and the validity period of data items at the client cache. 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). 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.

In practice, however, as explained in the previous section that the arrivals of some data items can be sporadic, 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, FVP will be longer than the previous estimate. The change in 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. 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 experience frequent link disconnection, and it is hard to verify the new AVI to the current cached copy of a disconnected mobile client.

Thus, we conclude that an invalidation report is needed to notify the client 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 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 better 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 these will result in a 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 AVI, and let the client make an explicit request when the data is actually accessed.

4.2 Server Algorithm

The server algorithm consists of two parts, invalidation report generation and AVI adjustment. AVI adjustment refers to the modification of the AVI values of the data items to achieve the desired level of the cache coherence.

4.2.1 Invalidation Report Generation

In order to notify the mobile clients of 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:

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

where $T_{update(i,n)}$ 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. AVI tolerance is designed to tackle randomness in update interval. Since it is not possible to predict the occurrences of update events, the update interval and AVI will not be perfectly matched. In order words, the current AVI is considered 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.

4.2.2 AVI Adjustment

Due to the dynamic nature of a data item, continuous adjustment on its AVI is needed to minimize the values of FVP and FIP. Since AVI is estimated by past update intervals, dramatic shifts of the mean update interval will lead to a large FVP or FIP. Furthermore, the continuous mismatch of AVI and update intervals means that a data item will be included in the invalidation reports for an extended period of time, increasing the invalidation report size and degrading overall system performance.

The minimum, mean and maximum values of the historical update intervals can be used to estimate the AVI of a data item depending on the desired degree of cache coherence. Using the minimum update interval as the item's AVI will result in the smallest AVI and the highest degree of cache coherence as FVP is minimized. However, the drawback is that FIP will be large. The

reverse is true when the maximum historical update interval is used. In the application of minimum and maximum values for the calculation of the new AVI, the value should be taken from the set of *n* preceding updates. This requires more complex handling but does allow the AVI value to adapt to changes in update intervals more effectively.

Similarly, when the mean value of update interval is used to estimate AVI, the AVI of a data item is obtained by taking the average for a number of preceding update intervals. Using the mean allows us to balance FIP and FVP and this approach is used in our simulation experiments described in Section 5 below.

4.3 Client Algorithm

Cache invalidation on the client side is divided into two parts, implicit invalidation and explicit invalidation.

4.3.1 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 a mobile client, the transaction will examine the update time and the 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 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 a cached item's AVI has expired, the data item 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.

4.3.2 Explicit Invalidation

Besides implicit invalidation, a cached item can also be invalidated explicitly by invalidation reports broadcast from the server. If the update interval of a data item violates the AVI assumption (eqn. 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 an 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 AVI and the update interval of a data item 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

We have compared the performance of IAVI with two efficient schemes, Bit-Sequence (BS) [JEHL95] and Timestamp (TS) [BI94]. Furthermore, 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 real-life as it requires the generation of updates regarding the content of mobile client caches continuously to the mobile server.

5.1 System Model and Parameters

Figure 2 depicts the simulation model of a mobile computing system, which uses data broadcast to disseminate data items. Data items are categorized into several groups and stored in the database at the mobile server. The Update Process continuously generates updates to update the data items at the mobile server according to the group's update interval. The update logs are generated and stored in the System Monitor temporarily. The Mobile Server retrieves the logs periodically for generating invalidation reports.

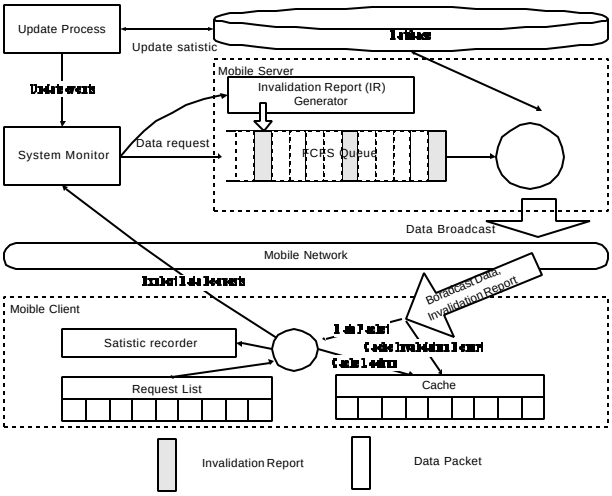

Figure 2 Simulation model

Mobile clients generate transactions and each mobile transaction consists of a set of data requests.

Once a transaction is generated, the mobile client will try to fulfill the data requirements of the transaction with its local cache. For data requests that cannot be fulfilled by the cached items, a message will be sent to the mobile server to request the data explicitly. If the validity of a cached item cannot be determined (for example after the client has just reconnected to the network after disconnection), the data request message will be deferred until the invalidation report is received. Once all the data requests of a transaction are fulfilled, the transaction will commit and the mobile client will either go into a doze mode or generate another transaction after a think time. The client is disconnected from the mobile network when it enters the doze mode.

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

Database size	100000 items
Items size*	256 bits
Update interval	3600 sec.
Update interval variance	-10%~ +10% uniform
Update processes group	4 groups
Group size distribution	5k, 10k, 15k and the rest of database size
Relative update interval* *	1.0, 2.0, 4.0 and 8.0
Number of mobile clients	100
Disconnect probability, interval	0.1, 4000 sec.
Access hot spot, probability	100 items, 0.8
Mean think time, query length	100 sec., 10 items
Cache size, replacement policy	50 items, LRU

*An extra 32 bits are required in IAVI scheme to store the AVI value of the data item.
** The relative update interval is a multiple of *update interval.*
Mobile client parameters

5.2 Results

5.2.1 Impact of Database Size

Figures 3 and 4 depict the mean response time of the transactions and invalidation report size against the database size, respectively. In Figure 3, we observe that PS and IAVI outperform TS and BS (smaller mean response time) and their performance is relatively unaffected by changes in database size. The performance of BS is affected significantly by the size of the database. When the database size is large, it is even worse than TS whose performance is consistently much worse than PS and IAVI. The poor performance of BS when the database size is large is due to the large invalidation report as can been seen in Figure 4. The size of invalidation report increases with the size of the database according to the formula $2N + \log 2N \times Time\_bit\_size$, where N is the size of the database and *Time_bit_size* is the number of bits needed to represent update time [JEHL95] whereas the invalidation report sizes of TS, IAVI and PS depend on the update volume.

5.2.2 Impact of Update Rate

As shown in Figure 5, the mean response time of TS decreases dramatically with an increase in update interval while those of the other schemes are relatively unaffected. The drop in response time of TS is due to the decrease in the invalidation report size when the update interval is longer, as can be seen in Figure 6. As the invalidation report of TS includes all update records within the window frame, report size is heavily influenced by the update interval. However, IAVI only records the update records that violate its AVI value and hence is less affected by changes in update interval.

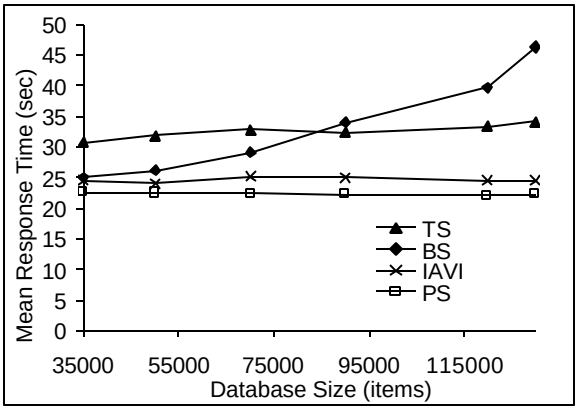

Figure 3: Mean Response Times vs Database Size

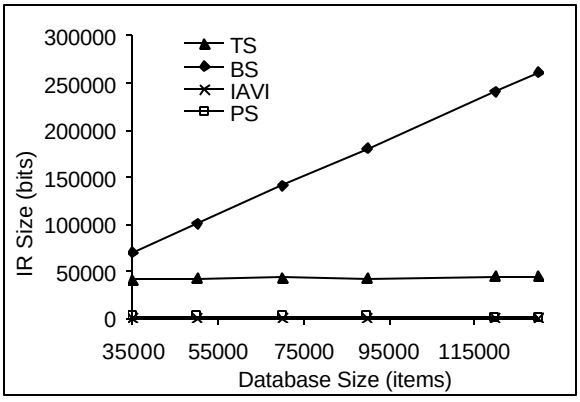

Figure 4: Invalidation Report Size vs Database Size

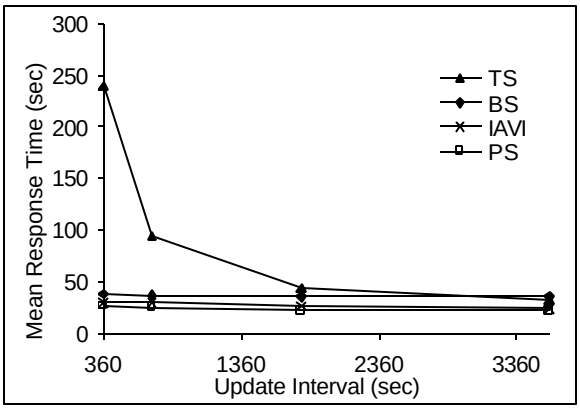

Figure 5: Mean Response Time vs Update Interval

5.2.3 Impact of the Cache Size

Figures 7 and 8 depict the performance of the cache invalidation schemes when different cache sizes are used. As expected, an increase in cache size reduces the mean response time and increases the cache hit rate. IAVI outperforms TS and BT significantly especially when the cache size is small. It is due to the fact that the false valid rate increases with cache size for all schemes but IAVI is relatively unaffected. This is because when the cache size increases, it is more likely that stale items will remain in the cache, resulting in an increase in the false valid rate. However, with IAVI, these items will be removed automatically when their AVIs expire.

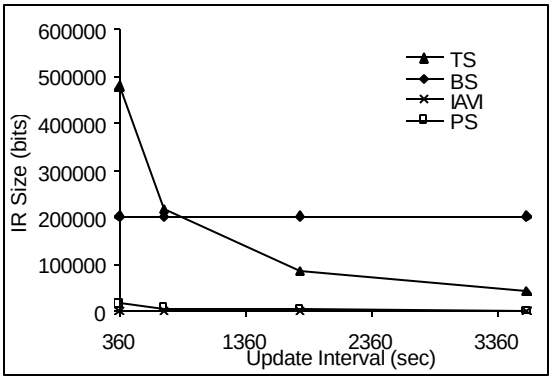

Figure 6: Invalidation Report Size vs Update Interval

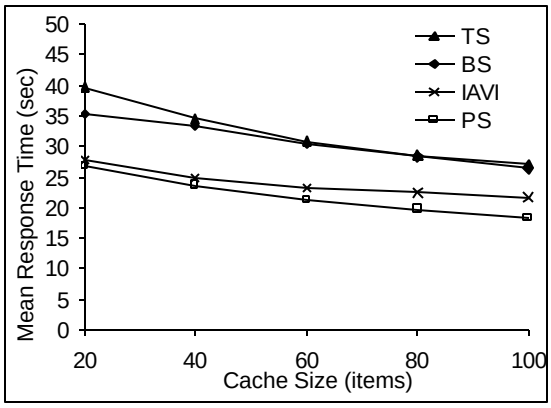

Figure 7: Mean Response Time vs Cache Size

6 Conclusions

In a mobile computing environment, mobile clients are usually equipped with local cache for reducing latency in data accesses. However, the frequent disconnection of mobile clients from the network and updates occurring at the mobile server introduces the problem of cache incoherence. Several cache invalidation schemes, such as Timestamps and Bit-sequence have been proposed to maintain cache coherence efficiently. The former sends out detailed update records within the predefined window frame to its client for cache invalidation while the latter organizes and records the update records in a report whose size depends on the database size. Therefore, the invalidation reports of both the Timestamps and Bit-sequence schemes are dependent on the actual updates occurring within the window frame and the size of database.

In this paper, we proposed the IAVI scheme for cache invalidation based on the real-time properties of the data items 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, although IAVI uses invalidation reports to inform the mobile clients about changes in AVI, the size of an invalidation report can still be reduced significantly.

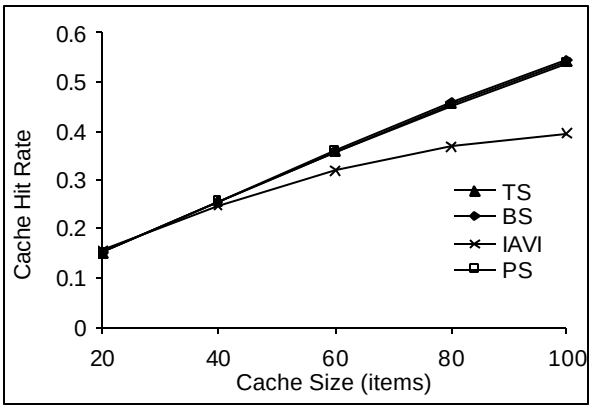

Figure 8: Cache Hit Rate vs Cache Size

References

[AAFZ95] Acharya, S., Alonso, R., Franklin, M., Zdonik, S., "Broadcast Disk: Data Management for Asymmetric Communication Environments", *Proc. ACM SIGMOD*, 1995.

[BI94] Barbara, D. and Imielinski, T., "Sleepers and Workaholics: Caching Strategies in Mobile Environments", *Proc. ACM SIGMOD*, 1994.

[DCKV97] Datta, A., Celik, A., Kim, J. and VanderMeer, D.E., "Adaptive Broadcast Protocol to Support Power Conservant Retrieval by Mobile Users", *Proc. 13th International Conference on Data Engineering*, 1997.

[HL97] 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.

[JEHL95] Jing, J., Elmargarmid, 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, 1995.

[SLR99] Srinivasan, R., Liang C., Ramamritham, K., "Maintaining Temporal Coherency of Virtual Data Warehouse", *Proc. Real Time Systems Symposium*, 1999.

[WYC96] Wu, K.L., Yu, P.S. and Chen, M.S., "Energy-efficient Caching for Wireless Mobile Computing", *Proc. 20th International Conf, in Data Engineering*, Feb 1996