Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 6141828, 9 pages
http://dx.doi.org/10.1155/2016/6141828

Research Article

Innovative Mobile E-Healthcare Systems: A New Rule-Based Cache Replacement Strategy Using Least Profit Values

Ramzi A. Haraty

Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon

Correspondence should be addressed to Ramzi A. Haraty; rharaty@lau.edu.lb

Received 26 October 2015; Accepted 29 December 2015

Academic Editor: Miltiadis D. Lytras

Providing and managing e-health data from heterogeneous and ubiquitous e-health service providers in a content distribution network (CDN) for providing e-health services is a challenging task. A content distribution network is normally utilized to cache e-health media contents such as real-time medical images and videos. Efficient management, storage, and caching of distributed e-health data in a CDN or in a cloud computing environment of mobile patients facilitate that doctors, health care professionals, and other e-health service providers have immediate access to e-health information for efficient decision making as well as better treatment. Caching is one of the key methods in distributed computing environments to improve the performance of data retrieval. To find which item in the cache can be evicted and replaced, cache replacement algorithms are used. Many caching approaches are proposed, but the SACCS—Scalable Asynchronous Cache Consistency Scheme—has proved to be more scalable than the others. In this work, we propose a new cache replacement algorithm—Profit SACCS—that is based on the rule-based least profit value. It replaces the least recently used strategy that SACCS uses. A comparison with different cache replacement strategies is also presented.

1. Introduction

Mobile cloud computing, which combines mobile computing and cloud computing, has gained momentum since 2009. The diverse characteristics of mobile devices and wireless network make the implementation of mobile cloud computing more complicated than for fixed clouds. One of the key factors in mobile cloud computing is the end to end delay in servicing a request. Data caching is one of the techniques widely used in wired and wireless networks to improve data access efficiency. A cache is a temporary storage of data likely to be used again. Caching succeeds in the area of computing because access patterns in typical computer applications exhibit locality of [1]. Caching is effective in reducing bandwidth demand and network latencies.

Caching frequently accessed data items is a very important and promising technique in mobile computing. In the client/server framework in a mobile computing environment, the clients are mobile units that communicate with data servers through a wireless associate, where information is accessed at all times. Cache management considers two dimensions: consistency and replacement policies. Cache invalidation keeps consistency between the server and the client cache. Cache replacement policies ascertain which data to be removed from the cache when no more space is there to hold a new data item.

Two approaches were proposed for maintaining cache consistency between users and servers: the stateful approach and the stateless approach. Combining both approaches, the Scalable Asynchronous Cache Consistency Scheme was proposed with the least recently used (LRU) replacement algorithm [2].

In this work, we implement the rule-based least profit value (R-LPV) with SACCS, which we call Profit SACCS, and compare it with other strategies of cache replacement. R-LPV is a cache replacement technique that considers both the caching parameters of a data item and the relationship of the item with the whole cache set. Relationships among data items are found using association rule-based data mining [3]. A profit function is designed to calculate the profit from caching a data item. Our contributions can be summarized as follows:

(i) We formally define the Scalable Asynchronous Cache Consistency Scheme and outline its functionality.

(ii) We proposed a rule-based least profit value as a replacement strategy for cache in a mobile e-healthcare system.

(iii) We present our enhanced SACCS algorithm that considers not only the data item caching parameters but also the relationship of this item with the other items in the cache.

(iv) We provide experimental results to ascertain the efficiency and adequateness of the least profit-based proposed solution.

The remainder of this paper is organized as follows: Section 2 presents related works. Section 3 presents Profit SACCS. The experimental results are presented in Section 4. And a conclusion is drawn in Section 5.

2. Related Work

In the past few years, mobile computing has been used profusely in computer technology [4]. Many problems arise from this computer field because the wireless connection is expensive and the bandwidth is limited. To save time in exchanging data between base stations and mobile users and to improve the performance of the system, data caching is imposed on mobile users in order to access the data recently used or most likely to be used in the future. Data caching is used in mobile computing environments to reduce the number of queries sent by clients to the server to access data needed. The recently used or likely to be used data is fetched in the caches of the mobile users in order to be reused in the future. Therefore, data caching improves the performance of mobile applications by reducing the expected data access delay. Caching performance is also affected by the cache replacement policy being used.

Cache replacements algorithms determine which data items in the cache will be replaced (i.e., choose a replacement victim) when the cache is full. Different strategies of selecting replacement victims are used to increase the performance of mobile systems [2, 5, 6]. Temporal locality, spatial locality, and semantic locality are three types of cache replacement policies that are pervasively in use today [7]. Temporal locality defines that the data being used will be reused in the future (e.g., most recently used). Spatial locality defines the data in a nearby place where the recently used data will be used again in the future [8]. The semantic locality means that a currently used data area could be accessed in the future [6].

To validate the data in their caches, mobile clients try to query the server congesting the networks' narrow bandwidth and getting stuck in the limited capability of the transmission. The invalidation method is a solution to this problem. It reduces the data transfer from the clients to the server and vice versa. Invalidation reports are proposed to invalidate out of use (or invalid) data in the mobile user caches [9]. After disconnections, reconnected users are prevented by using invalidation reports from discarding wastefully all their caches since some data may still be valid for use. Invalidation reports are based on two different methods:

(i) The Broadcasting Timestamp Strategy: updated data is represented in the report as a pair (ID, timestamp) where the ID is the updated data itself and the timestamp is the time at which the data was updated.

(ii) The Bit-Sequence Approach: the invalidation report gets bigger as the amount of the updated data gets larger. The Bit-Sequence Approach solves this problem by presenting each piece of data in the report by a bit where the bit sequence number in the report represents the data sequence number in the database. If the data has been updated, the value of the bit is 1 in the report or else the value of the bit is 0.

During the past few years, stimulating strategies were proposed to preserve data consistency in mobile computing systems. These strategies lead to a better system performance by reducing bandwidth utilization and query latency and saving power and energy for mobile clients [10]. The invalidation reports-based cache invalidation approach was used to preserve cache consistency. In this approach, invalidation reports (IRs) are periodically broadcasted by the server to the clients. When a client receives a request for some data, it waits for the next IR to invalidate or validate the data in its cache accordingly. If the data is invalid, the client sends a request to the server for a valid copy of the data. Otherwise, the client returns the requested data directly.

The latency for any client to answer a query depends on the length of the IR interval; as the IR interval gets longer, the answer to the query gets delayed. A solution to that delay was proposed by Cao to replicate the IR m times within the IR interval [11]. The replicated IRs, called updated invalidation reports, contain only the data items updated after the broadcast of the last IR. In this way, the client should wait the maximum of $1/m$ of the IR interval to answer a query for any data. This proposition reduced the query latency time.

For reducing the overhead of broadcasting reports and for improving the cache hit ratio, Shao and Lu proposed a substitution-based strategy to update the frequency of the data objects. Updating messages will be broadcasted instead of the invalidation messages [12]. Taking into consideration the intrafile and interfile relationships, Santhosh and Shi proposed a cache replacement algorithm based on semantic and file access patterns that are not random [13]. Nguyen and Dong [14] attempt to improve cache reusability among mobile clients that are disconnected. Anandharaj and Anitha [15] presented an architecture that achieves low latency and packet loss, as well as reduced network bandwidth consumption and data server workload. Zeitunlian and Haraty [16] presented a cache replacement algorithm based on time of all past references and hence considered the number of references to a data object.

In association rule-based cache replacement methods, the client access history is mined to get association rules and a confidence value [12, 17, 18]. Other caching variables (such as modification rate, cache invalidation delay, data messages size, and retrieval delay) of the resulting rules are used in

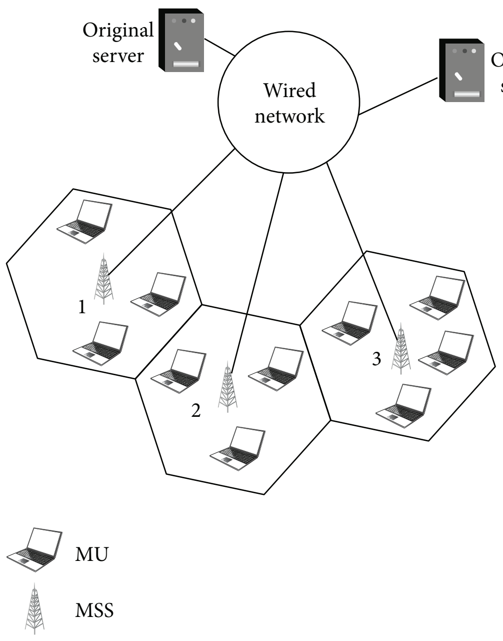

FIGURE 1: Wireless data communication system architecture.

computing the objective function [19–22] to determine which data item is to be replaced.

3. Profit SACCS

The mobile computing model is comprised of immobile stations or servers and hosts or mobile clients as shown in Figure 1 [23]. In this model, there are two types of cache consistency maintenance algorithms: stateless and stateful. In the stateless approach, the server is not informed about the client's cache content so it periodically broadcasts a data invalidation report to all mobile users (MUs). In such algorithms, a mobile support station (MSS) does not maintain any state information about its mobile user caches (MUCs), thus allowing simple database management for the server cache (SC) but poor scalability and ability to support user disconnectedness and mobility. On the other hand, the stateful approach is used with large database systems at the cost of complex server database management. The communication in these approaches is reliable between MUs and the MSS for IR broadcast, which means that an MU has to send back an acknowledgement for each IR received from the server broadcast. Thus, if a mobile user is disconnected, it does not receive any IR broadcast and the server, which does not get any acknowledgement, has to resend the IR again.

This paper builds on a novel cache consistency maintenance algorithm, called Scalable Asynchronous Cache Consistency Scheme (SACCS), which combines the positive features of both stateless and stateful approaches. Under unreliable communication environments, SACCS provides small stale cache hit probability; a stale cache hit for a data entry occurs at a connected MU when the MU misses the IR of the data entry and when the update time for the data entry is during its TTL (time-to-live period).

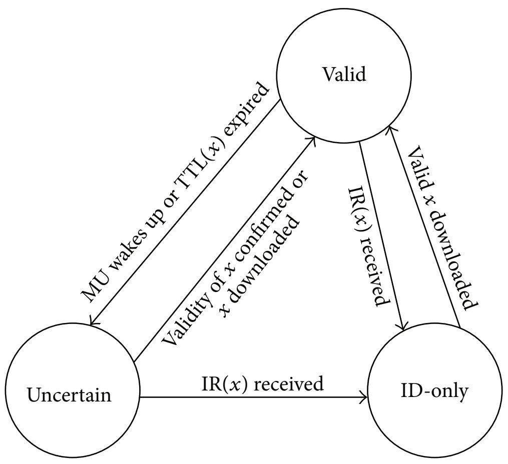
Valid
MU wakes up or TTL(x) expired
Validity of x confirmed or x downloaded
IR(x) received
Valid x downloaded
Uncertain
IR(x) received
ID-only

FIGURE 2: State diagram.

The MSS distinguishes only the valid state of MU objects, as opposed to the stateful strategy where all data objects have to be identified. And if we compare SACCS to stateless strategy, periodic broadcasting of IRs to MUs is not needed, so we have a less number of invalidation report messages.

The data structure used for the MSS and MU for each data object x is as follows:

(i) In the MSS: (d_x, t_x, f_x) where d_x is the data object; t_x is the last update time for the data object; and f_x is a flag bit.

(ii) In the MU: (d_x, ts_x, s_x) where d_x is the data object; ts_x is the timestamp indicating the last updated time for the cached data object; and s_x is a two-bit flag identifying four data entry states: *valid*, *uncertain*, *uncertain with a waiting query*, and *ID-only*.

There are three states for any data object: valid, uncertain, and ID-only as shown in Figure 2.

When the data is cached in the MU, then it is in valid state. When the data is cached in the MU but some disconnections occurred and when the MU is reconnected again then here the data is in uncertain state. When the data cached in the MU is updated and not valid, then the data is in ID-only state. The state changes from uncertain to valid or from ID-only to valid when the object is validated and downloaded. When the data object is in uncertain or valid state and an IR is broadcasted then the state of the data object changes to ID-only. Table 1 presents a description of the communication messages which occurs between the MSS and all the MUs.

The MSS takes the data entry from its cache and broadcasts it to all MUs after it receives any request from MU, and the flag is set to 1 in case it was 0. The timestamps of the data item in the MSS and the MU are compared in case an uncertain message from the MU is received. A confirmation message is broadcasted from the MSS in case the timestamps are equal; else, the valid data item is broadcasted to all MUs. The flag of data entry is set to 1 in case it was 0. In case the MSS gets an update message from the original server, then the data item is updated in the MSS database with a new estimated

Table 1: Communication messages.

Name	Sender	Receiver	Comments
Update(x, d'_x, t'_x)	Original servers	MSS	Indicating d_x has been updated to d'_x at time t'_x
VData(x, d_x, t_x)	MSS	MUs	Broadcast valid data object d_x with update time at t_x
IR(x)	MSS	MUs	Indicating cached d_x is invalid
Confirmation(x, t_x)	MSS	MUs	Indicating d_x is valid if $ts_x = t_x$
Query(x)	MUs	MSS	Query for data object d_x
Uncertain(x, ts_x)	MUs	MSS	Verifying if d_x in uncertain state with update time ts_x is valid

TTL. An IR is broadcasted to all MUs in case the flag of the data entry is 1 and then the flag is set to 0.

When an MU receives a message requesting a data object, the MU checks its local database or cache and the request is answered when the required object is valid, but a message will be sent to the MSS to get valid data in case the required object is in uncertain state. A message will be sent to the MSS to get new value of the data object in case the data entry is ID-only. An MU gets a *VData* message; the entry is refreshed in case the data in the cache is in an uncertain state; else, the MU responds to the request and caches the data object in case there is a query waiting for the data. The data object will be downloaded in case it was ID-only state. Data entries are set to ID-only in case an IR is received. The entry data will be refreshed when the MU receives a confirmation message and an uncertain entry is there corresponding to it.

Profit SACCS uses a rule-based least profit value (R-LPV). R-LPV considers not only the data item caching parameters but also the relationship of this item with the other items in the cache. At a mobile client, the data items that are in the cache are related to each other. Relationship between many different data items can be found or determined if a good observation of the history of these data items in the cache was held. We can know during a period of time the data item that will be accessed by the client in the near future by looking at the relationships between the data items that were already accessed by that client within that time period. An example of such a relationship is "*if* a client accesses *d1* and *d2*, *then* it accesses *d3* 80 percent of the time." The antecedent is the "if" part of the rule while the consequent is the "then" part. Data mining is used to find the association rules in the access history and to apply these rules to make the replacement decision. We will call them the caching rules. A caching rule $r_{x,y}$ is of the form $x \rightarrow d_y$, where x is subset of A, d_y belongs to A, and $x \cap \{d_y\}$ is empty. Where x is called antecedent of the rule and d_y is called consequent. An itemset is a set of data items; the size of the itemset is the summation of the number of data items in this itemset, and a k-itemset is an itemset of size equals to k. The percentage of sessions that contains x is called the support of an itemset x, and the support of the itemset which contains the items in both sides of the rule is called the support of the caching rule. The confidence of a caching rule $r_{x,y}$ (confidence ($r_{x,y}$)) is defined by the support of the rule divided by the support of the antecedent.

The problem of mining caching rules is to find all the association rules that have support and confidence greater than the user defined minimum support (minsup) and minimum confidence (minconf). Profit SACCS generates frequent itemsets from the client's access trace whose support is greater than a minimum support given as a parameter and then generates rules after being given the frequent itemsets and minimum confidence.

The profit gained due to data caching is the main factor in applying the R-LPV strategy. A replacement rule set for the data item is tabulated in order for the profit functions from caching this data item to be calculated. Data items to be replaced or cached have priorities determined by an eviction function, which each different replacement algorithm uses. In R-LPV, the profit function is calculated to determine the gain from caching a data item. Each item has a profit on which we depend if this item is to be evicted or not. The expected number of accesses c_y of d_y is calculated first in order to find the profit due to this item d_y.

The value c_y for an item d_y is calculated as the summation of the confidence $c_{x,y}$ of rules $r_{x,y}$, which belong to the replacement set Z_y as shown below:

$$c_y = \sum_{r_{x,y} \in Z_y} c_{x,y}. \tag{1}$$

Therefore, Profit_y for a cached item d_y is given as

$$\text{Profit}_y = c_y * ds_y, \tag{2}$$

where ds_y denotes delay-saving due to caching of item d_y and is computed as

$$ds_y = b_y - v - Pu_y * v_y. \tag{3}$$

Combining the above, we get $\text{Profit}_y = c_y x(b_y - v - Pu_y * v_y)$.

The objective of the cache replacement is to maximize the total profit value for cached data items. The R-LPV strategy chooses the cached data item with least profit and evicts it during each replacement.

In Profit SACCS, we first check the availability of the data in the cache. If the data is there in valid state, then we update the last reference's time, and the new value is calculated. We restore the heap property by adding the new value to the heap. Now if the data is in uncertain state, the new value is calculated at the new referenced time, and a message is sent for checking data validity. While if the data is ID-only or even not available in the cache, the data and its value are fetched. The time of the last reference is added and the new value is calculated, inserting it to the heap, and thus adjusting it. In case there is no room for this data to be downloaded then

```
Case 1. If $d_x$ is in the cache and valid,
        then calculate the R-LPV value
        return $d_x$ to the application
Case 2. If $d_x$ is in the cache and uncertain,
        then calculate the new R-LPV value
        send uncertain message to the server
Case 3. If $d_x$ is not cached or ID-only
        send cache missing message to the server.
        wait for message $M_x$ to appear at downlink channel
        if $M_x$ is confirmation, then set the state of $d_x$ as valid
                return $d_x$ to the application
        if $M_x$ is the data item $d_x$, then
                while there not enough space for $d_x$
                find minimum value $L$ = Minimum R-LPV value for data object $y$ belonging to the cache
                evict $d_y$ such that R-LPV value of $y = L$
                keep value of the evicted data object
                end while
        bring $M_x$ into cache
        calculate its R-LPV value
        return $M_x$ to the application
```

Algorithm 1: The Profit SACCS algorithm.

the data object with the least profit value is replaced with the new data object. The value of this data object inserted into the heap is calculated and the heap is restored.

Algorithm 1 depicts Profit SACCS. In Algorithm 1, d_x stands for the data object, M_x stands for the message for the data object, d_y stands for the data object that will be replaced, R-LPV is the value calculated for the data object, and L is the least profit value.

In the algorithm, ordering of data items is maintained by using the heap structure, taking into consideration the data items' R-LPV values. The least profit value data is the root of the heap. The data object at the root of the heap is the object to be evicted from the heap. The eviction of data objects is repeated until enough space is accumulated for the new data object to be inserted into the cache.

4. Experimental Results

We implemented the Profit SACCS algorithm and created a simulation environment written in the C++ programming language. We chose a one-cell setting that has a hundred mobile units where each mobile unit has a cache size of 300, a thousand data entries having different access commands, a random data size (in bytes), and a changeable average update interval (in seconds). Every mobile unit can be either in a sleep or in a wakeup state and the control of the sleep/wakeup process is done by a two-state Markov Chain [24]; thus, every mobile unit state may be changed on a randomly picked period from the following time intervals (500, 1000, 1500, 2000, and 2500 seconds). The sleep ratio can be 0.1, 0.3, 0.5, 0.7, or 0.9. The request arrival rate can be 1/10, 1/60, 1/110, 1/160, or 1/210. Note that when a mobile unit enters a sleep state, all of the requests that it receives are ignored. As for the query delay criteria, it is set at zero when a queried data entry is found in mobile unit cache; however, if the requested data object needs to be fetched from the mobile support station then the query delay is set to be the time span between query response and query initiation. For every query that is received on the mobile support station via an uplink channel, an uplink counter variable is incremented. As for simulating the mobile unit access pattern, a zipf-like distribution [25] is used with z equal to 1. A Poisson distribution [26] is used in the data entry update process and in the arrival requests. The downlink and uplink data transmissions use a channel with a bandwidth of 1250 bps. The message size for uplink and downlink is of size 64 bytes.

In order to evaluate the performance of our rule-based least profit value as a cache replacement policy for the SACCS, we ran a simulation and compared the results obtained to previously obtained results for various implementations: FIFO, LFU, LRU, MFU, MRU, and Random. In our comparison, we are interested in the following metrics: the total amount of hits and misses, the total hits and misses and their ratios over the simulation time, the total and the average delay over the simulation time, and the bytes per query as well as the data download per query values. Note that when a mobile unit gets a query for a certain data entry, it tries to find it in its cache. In case it is found, a cache hit counter gets incremented and the query needs no uplink. Thus, when aiming for a higher hit ratio there will be less uplinks per query. The factors that have been measured are total hit, total miss, miss ratio, bytes per query, and data download per query.

When query is received by a mobile unit, if the requested data object is valid in the cache, then no uplink is needed and a cache hit is calculated. When we have high hit ratio, then the uplink per query will be minor. The total hit and total query for R-LPV are the highest and are better than those of the others, as is shown in Figure 3. The figure depicts the total number of hits after running a complete simulation

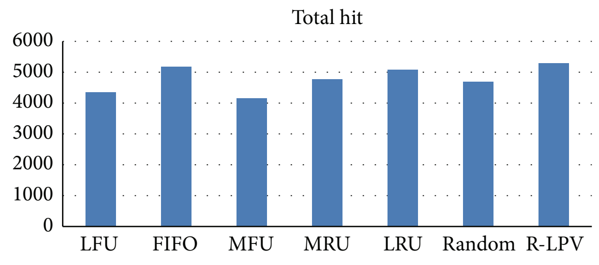

Figure 3: Total hit.

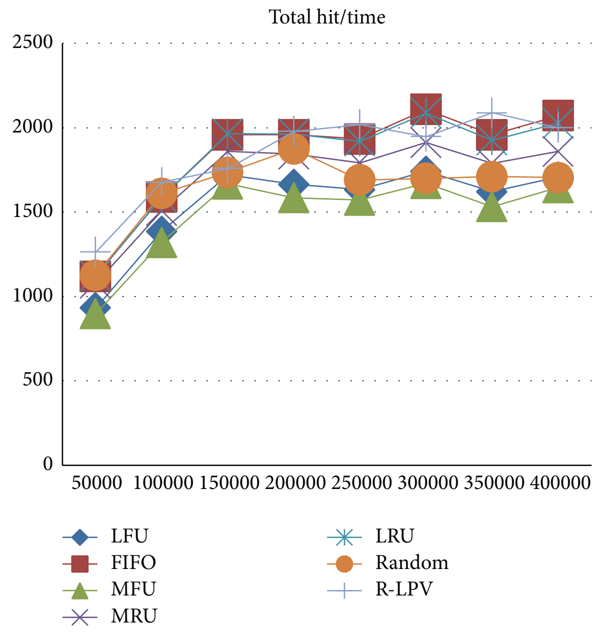

Figure 4: Hits per time.

using different cache replacement policies. This figure shows that the R-LPV cache replacement policy has improved the performance of the SACCS since it better manages the cache making the requested data available most of the time, which reduces invalidation reports broadcasts and as a consequence it avoids redundant network traffic.

Figures 4 and 5 show the number of hits and misses over different periods in the simulation time for the various cache replacement policies. The simulation was held over eight time units with an interval of 50000 msec of simulation time. Note that the miss/hit ratios are the sum of the unfound/found (resp., see Figures 6 and 7) data objects in the cache over the total number of data queried. It is significant that the least hit ratios are for the LFU and Random while the highest are for the FIFO and R-LPV.

As a crucial metric in measuring our system performance, we have drawn two graphs (Figures 8 and 9) that portray the total delay, as well as the average access delay over different periods during simulation time for the various cache replacement strategies. The notion of delay here refers to the time interval between the instant a request is issued and the time a response is received on a mobile unit. One solution for improving the access delay is by decreasing the uplink

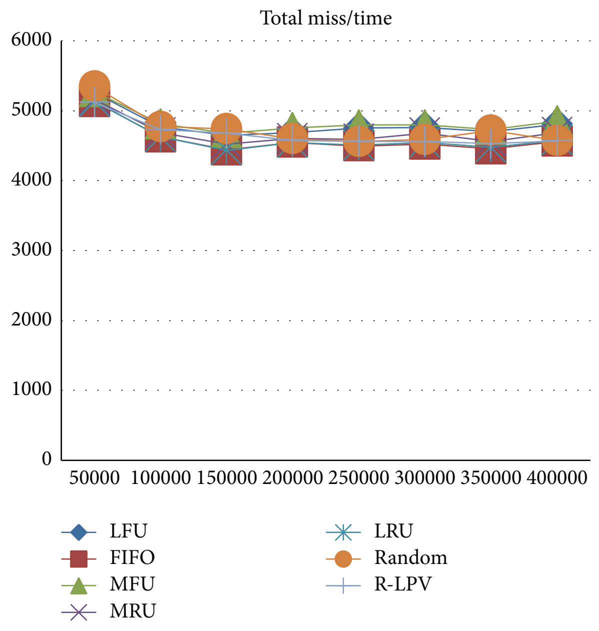

Figure 5: Misses per time.

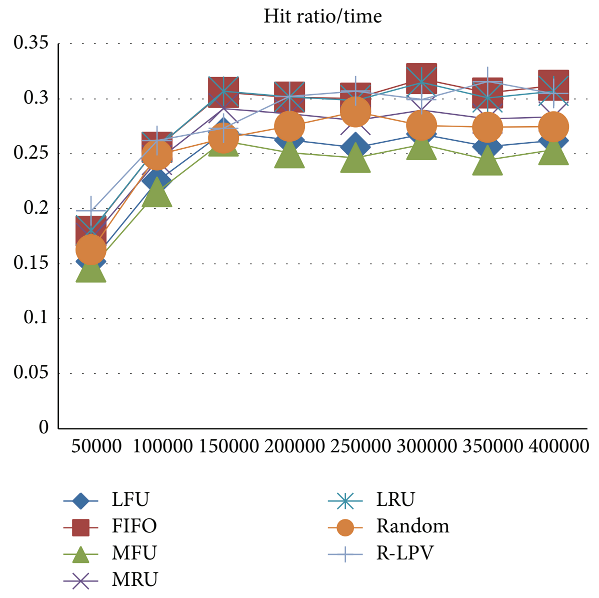

Figure 6: Hit ratio per time.

and downlink messages sizes and number. As shown in these graphs, the FIFO and R-LPV approaches are the best performers because they have the lowest values in terms of total delay and average access delay.

Figure 10 presents the number of bytes per query which directly affect power consumption and bandwidth usage. The graph shows that the FIFO and R-LPV approaches are the strategies that have the least number of bytes used over their queries. This is because these approaches consider the most recent data information and their innovative ways for choosing which data object to replace in case there is no

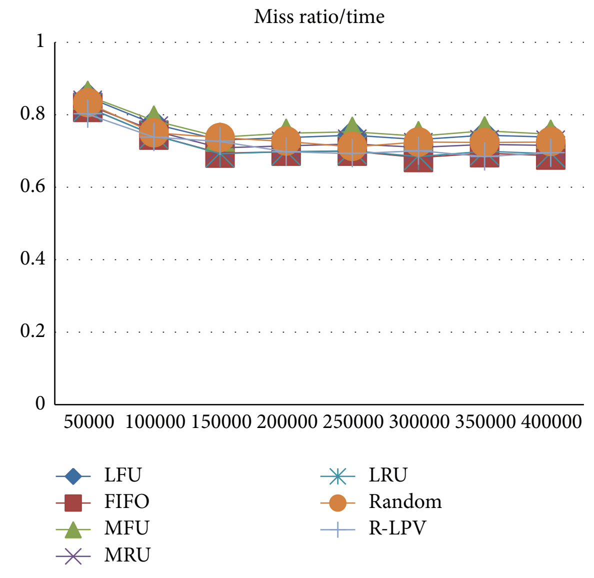

Figure 7: Miss ratio per time.

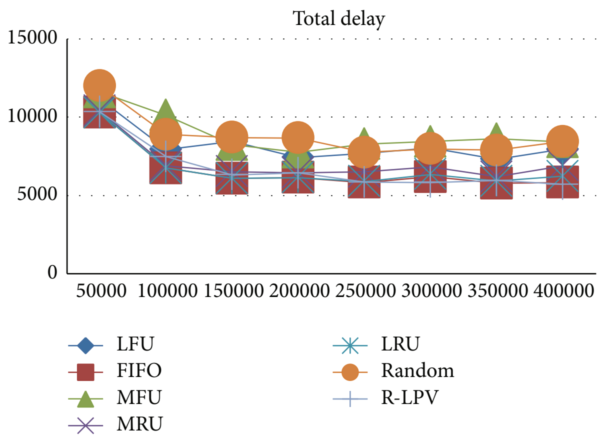

Figure 8: Total delay.

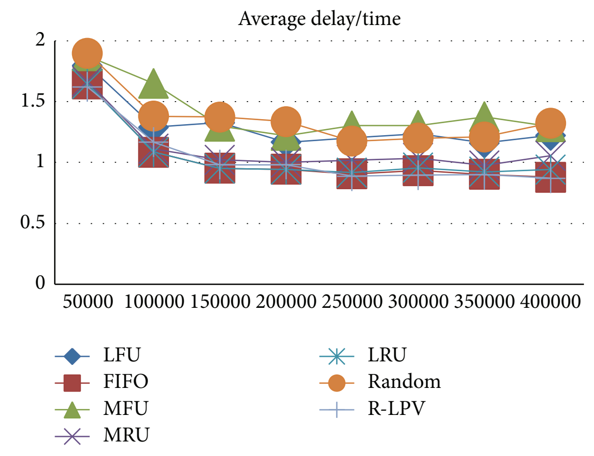

Figure 9: Average delay per time.

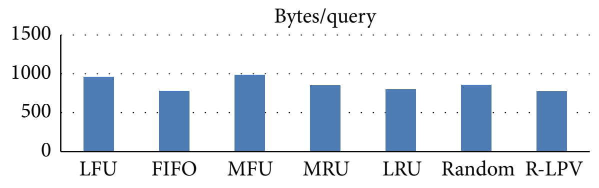

Figure 10: Bytes per query.

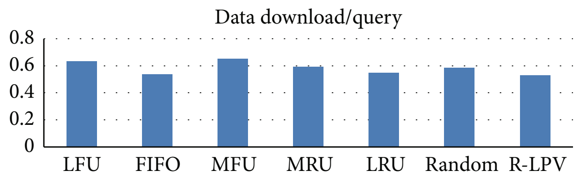

Figure 11: Data download per query.

available space in the cache list. For example, in the R-LPV approach, the data object that is least probable to be referenced in subsequent calls is the one that is removed.

Figure 11 shows the number of downloads per query for each cache replacement strategy. The R-LPV and the FIFO approaches have almost the same lowest values in this graph because they are the most efficient in selecting which data objects to replace, thus eliminating the need of unnecessary downloads, since R-LPV chooses data objects which are poorly related to the rest of the data entries in the cache. Note that since these strategies have the lowest download per query values, they are also less resource-demanding in terms of power consumption.

As we have seen in the results presented above, our R-LPV approach shows an improvement over the FIFO and the LRU cache replacement strategies. Our approach showed the highest total hit and hit ratio and the lowest total delay across all suggested replacement algorithms. These chosen metrics for comparison are crucial to study the overall system performance and the experiences that the users might encounter when handling their mobile units.

5. Conclusion

Caching is a key method to improve the performance of data retrieval in mobile computing environments. Since the mobile and the server communicate through a wireless network, then they may face many problems such as irregular connections, low bandwidth for uplink, and limited client resources. These problems can be relieved by caching at the mobile client, but caching has many limitations. To maintain cache consistency, many approaches have been suggested such as stateful, stateless, and SACCS approaches. But even with caching a problem arises which is limitation of cache size. In order to manage this limitation, replacement strategies have been suggested.

In this paper, we presented Profit SACCS—a new cache replacement algorithm that is based on the rule-based least profit value function. Both caching parameters of a data item and the relationship of this item with the cache set are considered. Relationships among items are found using association rule-based data mining. A profit function is designed to calculate the profit from caching an item. The objective is to maximize the total profit value for cached data items. The algorithm chooses the cached data item with least profit and evicts it during each replacement. The algorithm was compared with other strategies and it showed better results. As for future work, data mining techniques used to find relationships among items can be used for combining prefetching with caching to improve the data availability at the client.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was funded by the Lebanese American University, Beirut, Lebanon.

References

[1] S. G. Dykes and K. A. Robbins, "A viability analysis of cooperative proxy caching," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '01)*, vol. 3, pp. 1205–1214, IEEE, Anchorage, Alaska, USA, April 2001.

[2] Z. Wang, S. K. Das, H. Che, and M. Kumar, "A scalable asynchronous cache consistency scheme (SACCS) for mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 11, pp. 983–995, 2004.

[3] N. Chand, R. C. Joshi, and M. Manoj, "Energy efficient cache replacement in mobile computing environment," *Ubiquitous Computing and Communication Journal*, pp. 1–10, 2009.

[4] C. Li and L. Li, "Energy efficient resource management in mobile grid," *Mobile Information Systems*, vol. 6, no. 2, pp. 193–211, 2010.

[5] T. N. T. Minh and T. D. T. Bich, "An efficient cache invalidation strategy in mobile information systems," in *Proceedings of the IEEE International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF '10)*, pp. 1–4, IEEE, Hanoi, Vietnam, November 2010.

[6] A. B. Waluyo, B. Srinivasan, and D. Taniar, "A taxonomy of broadcast indexing schemes for multi channel data dissemination in mobile databases," in *Proceedings of the 18th International Conference on Advanced Information Networking and Applications (AINA '04)*, vol. 1, pp. 213–218, Fukuoka, Japan, March 2004.

[7] X. Ding, S. Jiang, and F. Chen, "A buffer cache management scheme exploiting both temporal and spatial localities," *ACM Transactions on Storage*, vol. 3, no. 2, article 5, 2007.

[8] A. B. Waluyo, B. Srinivasan, and D. Taniar, "Optimal broadcast channel for data dissemination in mobile database environment," in *Advanced Parallel Processing Technologies: 5th International Workshop, APPT 2003, Xiamen, China, September 17–19, 2003. Proceedings*, vol. 2834 of *Lecture Notes in Computer Science*, pp. 655–664, Springer, Berlin, Germany, 2003.

[9] H. Chavan and S. Sane, "Mobile database cache replacement policies: LRU and PPRRP," in *Advances in Computer Science and Information Technology*, vol. 131 of *Communications in Computer and Information Science*, pp. 523–531, Springer, Berlin, Germany, 2011.

[10] A. Gaddah and T. Kunz, "Extending mobility to publish/subscribe systems using a pro-active caching approach," *Mobile Information Systems*, vol. 6, no. 4, pp. 293–324, 2010.

[11] G. Cao, "A scalable low-latency cache invalidation strategy for mobile environments," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 5, pp. 1251–1265, 2003.

[12] X.-K. Shao and Y.-S. Lu, "Maintain cache consistency of mobile database using dynamical periodical broadcast strategy," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 2389–2393, Xi'an, China, November 2003.

[13] S. Santhosh and W. Shi, "A semantic-based cache replacement algorithm for mobile file access," in *Proceedings of the 14th International World Wide Web Conference (WWW '05)*, Chiba, Japan, May 2005.

[14] T. T. M. Nguyen and T. T. B. Dong, "An adaptive cache consistency strategy in a disconnected mobile wireless network," in *Proceedings of the IEEE International Conference on Computer Science and Automation Engineering (CSAE '11)*, pp. 256–260, Shanghai, China, June 2011.

[15] G. Anandharaj and R. Anitha, "A distributed cache management architecture for mobile computing environments," in *Proceedings of the IEEE International Advance Computing Conference (IACC '09)*, pp. 642–648, IEEE, Patiala, India, March 2009.

[16] A. Zeitunlian and R. A. Haraty, "An efficient cache replacement strategy for the hybrid cache consistency approach in a mobile environment," *World Academy of Science, Engineering and Technology*, no. 63, 2010.

[17] R. A. Haraty and J. Zeitouny, "Rule-based data mining cache replacement strategy," *International Journal of Data Warehousing and Mining*, vol. 9, no. 1, pp. 56–69, 2013.

[18] B. Jiang, X. Hu, Q. Wei, J. Song, C. Han, and M. Liang, "Weak ratio rules: a generalized Boolean association rules," *International Journal of Data Warehousing and Mining*, vol. 7, no. 3, pp. 50–87, 2011.

[19] R. A. Haraty and L. Turk, "A comparative study of replacement algorithms used in the scalable asynchronous cache consistency scheme," in *Proceedings of the 19th ISCA International Conference on Computer Applications in Industry and Engineering (CAINE '06)*, Las Vegas, Nev, USA, November 2006.

[20] A. Kahol, S. Khurana, S. K. S. Gupta, and P. K. Srimani, "A strategy to manage cache consistency in a disconnected distributed environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 7, pp. 686–700, 2001.

[21] M. S. Sulaiman Khan, M. K. Muyeba, F. Coenen, D. Reid, and H. Tawfik, "Finding associations in composite data sets: the CFARM algorithm," *International Journal of Data Warehousing and Mining*, vol. 7, no. 3, pp. 1–29, 2011.

[22] Y. S. Koh, R. Pears, and G. Dobbie, "Automatic item weight generation for pattern mining and its application," *International Journal of Data Warehousing and Mining*, vol. 7, no. 3, pp. 30–49, 2011.

[23] J. Jayaputera and D. Taniar, "Data retrieval for location-dependent queries in a multi-cell wireless environment," *Mobile Information Systems*, vol. 1, no. 2, pp. 91–108, 2005.

[24] D. Gamerman, *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*, CRC Press, Boca Raton, Fla, USA, 1997.

[25] G. K. Zipf, *Human Behavior and the Principle of Least Effort*, Addison-Wesley, Reading, Mass, USA, 1949.

[26] F. A. Haight, *Handbook of the Poisson distribution*, Publications in Operations Research, John Wiley & Sons, New York, NY, USA, 1967.