A Recommended Replacement Algorithm for the Scalable Asynchronous Cache Consistency Scheme

Ramzi A. Haraty(✉) and Lama Hasan Nahas

Department of Computer Science and Mathematics,
Lebanese American University, Beirut, Lebanon
rharaty@lau.edu.lb, lama.nahas@lau.edu

Abstract. Acknowledging the widespread prevalence of mobile computing and the prominence of data caching in mobile networks has led to a myriad of efforts to alleviate their associated problems. These challenges are inherited from the mobile environment because of low bandwidth, limited battery power, and mobile disconnectedness. Many caching techniques were presented in literature; however, Scalable Asynchronous Cache Consistency Scheme (SACCS) that proved to be highly scalable with minimum database management overhead. SACCS was initially implemented with Least Recently Used (LRU) as a cache replacement policy. In this project we adopted the Extended LRU (E-LRU) as a cache replacement strategy to be applied in SACCS. A simulation of SACCS was done with the E-LRU, Least Recently Used, Most Recently Used, Most Frequently Used, and Least Frequently Used; and the comparative evaluation showed that SACCS with E-LRU is superior in terms of delay time, hit ratio, miss ratio and data downloaded per query.

Keywords: Mobile computing · Cache consistency · Replacement policy · Invalidation strategy · Hybrid algorithms

1 Introduction

Mobile computing allows greater ease of communication and versatility in using technology. Mobiles support a wide range of applications, allowing people to retrieve subtle information, such as stock prices, in real time. However, mobility comes hand-in-hand with several issues resulting from frequent disconnections and limited resources. Caching is storing desired data in local storage to improve data availability. It is an effective way of enhancing system performance. It improves the bandwidth utilization, while minimizing the query delay time and battery consumption. Nevertheless, maintaining cache consistency and a rigorous cache replacement policy are vital for successful caching. Cache consistency is usually actualized using either stateful or stateless approaches. The former is used for larger scale database systems, yet compromises nontrivial overhead in attempting to manage the server database. Meanwhile, according to the latter, namely the stateless approach, the mobile user's cache content is kept outside the awareness of the server. Yet, unlike the stateful

© Springer Nature Singapore Pte Ltd. 2018
K.J. Kim et al. (eds.), *IT Convergence and Security 2017*,
Lecture Notes in Electrical Engineering 449,
DOI 10.1007/978-981-10-6451-7_11

approach, it is not efficient when applied to large scale database systems, and is not capable of dealing with the user's mobility and disconnectedness [1].

A combination of both aforementioned approaches was proposed through what was termed "Scalable Asynchronous Cache Consistency Scheme". However, an effective replacement policy would substantially contribute to the success of the promising scheme. Based on the limited size of mobile's internal memory, a replacement strategy is needed to determine which data should be evicted when the mobile cache is full. The algorithm is a key factor, as it notably affects the performance of the whole scheme. Many replacement algorithms have been proposed in literature, and we're recommending an algorithm that proved to better suit SACCS.

The vast development in the computing realm allowed for massive advancement in real life applications, and there is still a wide range of improvement for future work. The client/server environment in today's wireless distributed networks gives more capabilities for mobile users. Nevertheless, it's a challenging environment due to bandwidth constraints and to the nature of mobile units that allows frequent disconnectedness from network and limited battery power. It allows the user to connect from different access points and to preserve their connection even when displaced [2].

In this work, we implemented an extended version of Least Recently Used replacement algorithm with SACCS. The E-LRU considers size and recency of usage when choosing the data to be replaced. The rest of the paper is organized as follows: Sect. 2 presents a literature review of cache consistency approaches and replacement policy strategies. Section 3 thoroughly explains the SACCS algorithm; Sect. 4 presents E-LRU, our recommended replacement policy for SACCS. Section 5 shows and compares the simulation results of the replacement policies integrated with SACCS. Finally, Sect. 6 concludes the study and suggests future work.

2 Literature Review

2.1 Data Caching Technique

Data caching is saving a copy of data in the client's side memory to avoid retrieving it again from source node soon. Data caching proved to be an extremely effective technique to preserve scarce resources. Its effect is further manifested in cases of small database and frequent queries, and when the communication cost is high [3]. Caching technology is extensively used in software and hardware, and is more crucial in a mobile environment in order to mitigate the usage of bandwidth, memory and energy. Cache consistency and replacement policy are two main pillars for successful cache management. Cache consistency techniques guarantee the validity of data stored in a mobile cache; while replacement policies determine the data item/s to be evicted from the local cache when it is too full to accommodate any new caches [4].

2.2 Cache Replacement Policy

Knowing the limited size of memory in mobile units, the subset dedicated for cache is obviously limited and valuable. Hence, when the cache is replete, a replacement policy is needed as a criterion to decide which data should be dropped from cache to make room for a new data item [5]. Cache replacement policies fall into three categories:

- Temporal locality expects recently used data items to be revisited shortly.
- Spatial locality anticipates that data nearby recently referenced data could be accessed in a future time.
- Semantic locality signifies that recently accessed data area is more likely to be accessed again soon.

Several factors play a role in deciding on the data items being replaced, the top of which are the following: access probability, recency of data access, access frequency, data size, fetching and communication cost, update rate, distance, etc. [6].

In a wireless environment, more parameters should be considered, such as connectivity, bandwidth and location. Many policies are based on a function that combines different parameters.

Some promising policies are modified to enhance their performance or to better fit in a different environment. Therefore policies have many variants; LRU for example uses temporal locality and has variants such as LRU-k, LRU-THOLD, E-LRU, etc. One of the LRU refined replacement policy is LRU-MIN which focuses on the size of data items being added to or removed from cache. It finds all cached data items of size equal to or greater than the size of the newly arrived data item, denoted as *Size-A*, then deletes the least recently used data among them. If all cached data items are smaller than *Size-A*, then LRU-MIN finds all cached data items of size equal to or greater than half of *Size-A*. In this case, the two least recently used data items will be deleted, and so on until enough space has been emptied [7]. Many other value function cache replacement strategies were presented in the literature, for instance SAIU, SAUD, Min SAUD, On Bound Selection, etc. [8].

2.2.1 Location-Based Cache Replacement

A wide range of cache replacement policies became more location oriented after the propagation of location-dependent information services via mobile devices. Effective parameters in these policies are distance, valid scope area and direction. Manhattan was one of the early introduced location aware policies. Farther Away Replacement (FAR) is a spatial locality approach that removes the farthest data item from the client's location. Predicted Region Based Replacement Policy (PRRP) is another location based policy. It relies on the size of data in the cache as well as the predicted region where the client will shortly arrive [9].

2.2.2 Coordinated Cache Replacement

Energy-efficient Coordinated cache Replacement Problem (ECORP) chose energy efficiency to be the main performance objective [10]. In ECORP and Dynamic ECORP DP adjacent nodes only store different data items in an attempt to better utilize the collective cache.

2.3 Invalidation Reports

Mobile data is prone to inconsistency since mobiles may reply to a query using local cache, while the requested data item is being updated in the server database. Broadcasting the updated data to all mobile units preserves consistency, yet it is costly and impractical. Invalidation Report (IR) technique proved to be more efficient since an IR may only contain the IDs and timestamps of the altered data; or the IR may contain a bit sequence that resembles the database, and assign bit value 1 to the updated data and 0 for the rest. Timestamp and bit sequence techniques significantly limit the overhead of transferring data items, since their size is lesser than the size of the actual data. Thus, IR approaches avoid congesting the network with unnecessary traffic. The role of IR is to invalidate data available in mobile units' caches. Whenever a mobile unit receives a data request from its user, it checks if data is available in its cache; and, if it is not available, the MU immediately sends a query to its MSS requesting the data. Otherwise, if the data was present in the cache, the mobile unit (MU) waits for the following IR to confirm the validity of the cache data so it can be sent to the user. If the IR indicates that the cache item is invalid, an original copy is imported from MSS and forwarded to the user. The IR interval length determines the mobile unit's pace of answering any query. The longer the IR interval gets the more delay that occurs in answering queries. Replicating IR *m* times within the IR interval was one approach to decrease the latency. Thus, the upper bound of the query latency is $1/m$ of the IR interval time. The additional IRs in this approach, called Updated Interval Reports, strictly contain the data items updated after the last IR [11]. Broadcasting IRs has many advantages, yet missing an IR threatens the cache consistency; unfortunately it is possible for an MU to lose an IR especially in the case of frequent disconnection. One antidote is to send back an acknowledgment for every IR received, but this will significantly increase the communication overhead [12].

2.3.1 Stateful Invalidation Schemes

It is rare to find stateful cache consistency maintenance approaches recommended for wireless networks. Kahol et al. [13] proposed an Asynchronous Stateful algorithm denoted AS, where the server keeps track of all data items in every MU, and sends IRs to MUs only when they are connected. In case of sleep, the IR is buffered at MSS to be resent later. However, MSS can't differentiate between a missing IR and a disconnected MU; therefore, it is necessary to allow a maximum number of retransmissions. A stateful approach for cooperative cache consistency called Greedy Walk-based Selective Push (GWSP) was proposed by Huang et al. [14]. In GWSP, the source node saves the Time to Live (TTL) and the request rate of every cached item in all caching nodes. This information is used to dynamically choose which caching node should receive the updated version of a data item. Whenever a cached data is updated in the source node, a greedy walk technique is used to deliver the updated data to the chosen caching nodes only.

3 The Scalable Asynchronous Cache Consistency Scheme

In this work, we consider the data communication system to consist of a wire-connected network that links many main servers to distributed Mobile Support Stations (MSS). The MSS is wirelessly connected to several mobile units. SACCS maintains cache consistency between the cache of MSS and the local cache of Mobile Units (MUs). SACCS assumes that the cache consistency between servers and MSSs is taken care of. Each data entry in the cache has three states: valid, uncertain and ID-only. Mobile user data cache can be used when requested if “valid”. If the MU receives an IR that invalidates a valid or uncertain data item, only the data ID will remain in cache and the valid data will be downloaded later if needed. The state of “valid” cache entry will become “uncertain” when its TTL ends. Also, all “valid” data in the cache will become “uncertain” in case the mobile unit disconnects and reconnects to the network (sleep/wake situation). When an “uncertain” cache entry is requested, the MSS will either approve its validity or it will send the updated version to be downloaded [1].

4 Cache Replacement Policy for SACCS

Whenever the size of the cache reaches its limit, the replacement policy determines which data is to be evicted in order to allow new data into the cache. To make its decision, the Extended-LRU relies on size, frequency of references as well as recency. E-LRU sets a threshold for acceptable size of a cached data item. If the data item size is more than 50% of the cache size, the query requesting this data will be answered directly without importing the data item into the cache. E-LRU prioritizes the data item that is referenced only once for eviction. If more than one data item in the cache is referenced once, the data item least recently referenced among them will be deleted. In case all data in the cache is referenced more than once, the data item with the longest inter arrival time between its last two referrals will be the victim. In case more than one data item has the same maximum inter arrival time, the one with less TTL remaining will be evicted [6]. E-LRU is a replacement policy suggested for various systems. In this study, we recommended it with SACCS and compared it with well-known cache replacement policies.

5 Experimental Results

A simulation of SACCS was performed with five replacement policies: E-LRU, LRU, MRU, LFU and MFU. The simulation environment was unified by fixing the number of mobile clients, the size of cache, the number of data items, etc. Other parameters were randomly determined, such as: the size of data items, update rate, the bandwidth, the uplink and downlink message size, etc. The sleep/wake time and frequency were randomly chosen from a predetermined set. The simulation was run in eight time slots, varying from 50,000 microseconds to 400,000 in 50,000 intervals. The simulation environment assumes that the network architecture consists of one cell. The performance evaluation metrics consists of the hits/misses total, and the latency reflected by the average delay and the total delay time.

5.1 Hit and Miss Ratio

When the requested data item is found in the cache, the query is counted as one hit. On the other hand, when the query needs to download the requested data from the source node because it is not available in mobile client cache, this query is counted as one miss. Certainly, the algorithm with a higher number of hits and lower number of misses is more likely to outperform other algorithms. The E-LRU total hits reached 5,216 queries during the simulation period, where the second best replacement algorithm was LRU with 5,086 hits as depicted in Fig. 1.

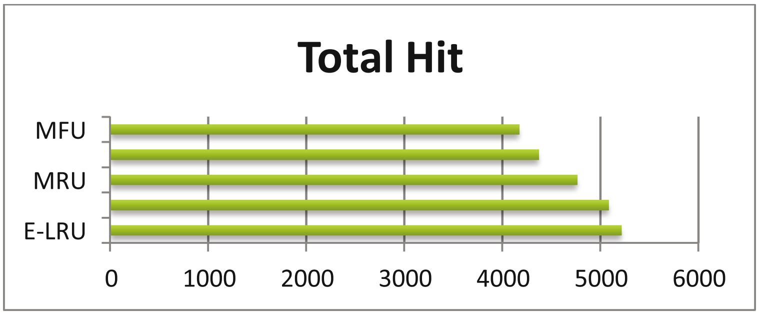

Fig. 1. Total hit

The total missed queries summed up to 11527 with E-LRU, a number far less than other replacement policies: 11666, 11953, 12342 and 12519 in LRU, MRU, LFU and MFU respectively. This is shown in Fig. 2.

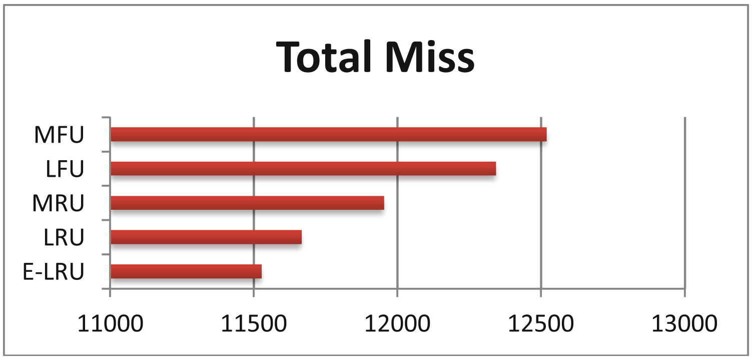

Fig. 2. Total miss

E-LRU takes size into consideration when taking the replacement decision. It does not approve removing many items form cache to import one large item into cache. There is no data item that is permitted to occupy more than half of the size of MU cache. This sounds logical since it allows better exploitation of the cache memory. Size check is definitely an improvement over LRU. The impact of this extra check is clearly demonstrated in the high numbers of hits E-LRU achieved.

5.2 Latency

Another important performance metric is the latency. The latency is calculated by measuring the time the mobile client waits between issuing a query and receiving a response to it. In our simulation, the latency was considered negligible in case of a hit. The total delay over the whole simulation time summed up to 15248 with E-LRE, 15346 with LRU, 16823 with MRU, 19483 with LFU and the longest total delay was 21,751 with MFU (see Fig. 3). Also, the average delay reflected the same result.

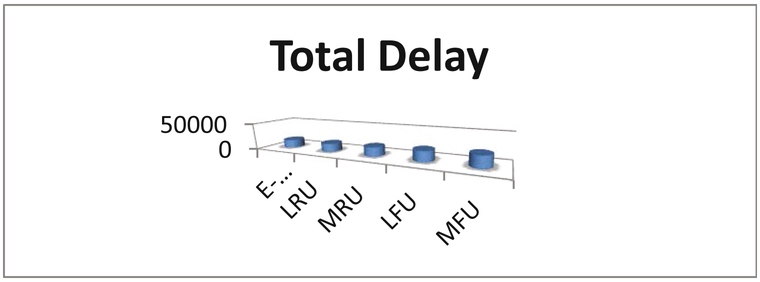

Fig. 3. Total delay

Also, the average delay, presented in Fig. 4, reflected the same results as the total delay. Query delay degrades dramatically upon improving data availability. Deleting data items referenced only once allows space for data items that are frequently referenced and more likely to be referenced again.

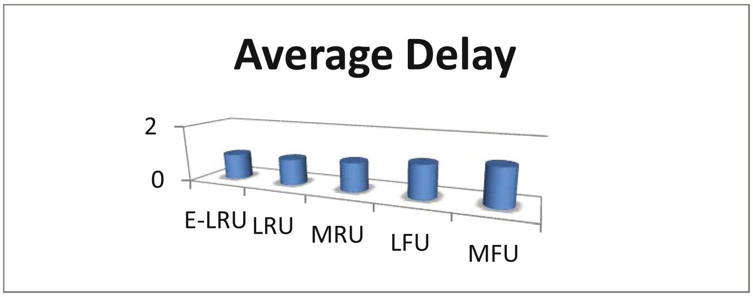

Fig. 4. Average delay

6 Conclusion

Mobile computing is imperative to the rapid advancement of technology, while in parallel, data caching is crucial to the efficiency and pragmatism of mobile computing. Ergo, dealing with the complications that arise from the dynamic environment of mobiles is inevitable. Given a rigorous highly scalable scheme as SACCS, finding a matching replacement policy is vital to its success. Initially, SACCS was implemented with the basic LRU. In this work, we recommended a cache replacement policy for SACCS that outperforms it. Simulating SACCS with various replacement policies shows the immense impact the replacement policies have on overall performance. The comparative evaluation shows that our recommended cache replacement policy, E-LRU, is superior to basic approaches (LRU, MRU, LFU and MFU) in terms of hit ratio, miss ratio, query delay and bytes downloaded per query.

There is no one perfect cache replacement policy that fits in every scenario. Therefore, selecting the right cache replacement algorithm out of the enormous variety presented in literature could always lead to better results. Moreover, comparing replacement algorithm against other advanced replacement policies will be more indicative than comparing with basic policies. In our future work, we would like to further improve SACCS by examining more replacement algorithms, and studying their performance in comparison with E-LRU and other promising policies.

References

1. Wang, Z., Das, S.K., Che, H., Kumar, M.: A scalable asynchronous cache consistency scheme (SACCS) for mobile environments. IEEE Trans. Parallel Distrib. Syst. **15**(11), 983–995 (2004)
2. Zeitunlian, A., Haraty, R.A.: An efficient cache replacement strategy for the hybrid cache consistency approach in a mobile environment. In: Proceedings of the International Conference on Computer, Electrical, and Systems Science, and Engineering (ICCESSE 2010), Rio De Janeiro, Brazil, March 2010
3. Barbara, D., Imielinksi, T.: Sleepers and workaholics: caching strategies in mobile environments. ACM SIGMOD Record **23**(2), 1–12 (1994)
4. Haraty, Ramzi A.: Innovative mobile e-Healthcare systems: a new rule-based cache replacement strategy using least profit values. Mobile Inf. Syst. (2016). doi:10.1155/2016/6141828
5. Jane, M.M., Nouh, F.Y., Nadarajan, R., Safar, M.: Network distance based cache replacement policy for location-dependent data in mobile environment. Paper presented at the Ninth International Conference on Mobile Data Management Workshops. IEEE, Beijing (2008)
6. Joy, P.T., Jacob, K.P.: Cache replacement strategies for mobile data caching. Int. J. Ad hoc Sens. Ubiquitous Comput. (IJASUC) **3**(4), 99 (2012)
7. Abrams, M., Standridge, C., Abdulla, G., Williams, S., Fox, E.: Caching proxies: limitations and potential. In: Paper presented at the Fourth International World Wide Web Conference: The Web Revolution, Boston, MA, USA (1995)

8. Hattab, E., Qawasmeh, S.: A survey of replacement policies for mobile web caching. In: Paper presented at the 2015 International Conference on Developments of E-Systems Engineering, pp. 41–46. IEEE, Dubai (2015)
9. Kumar, A., Misra, M., Sarje, A.: A new cache replacement policy for location dependent data in mobile environment. In: Paper presented at the 2006 IFIP International Conference on Wireless and Optical Communications Networks. IEEE, Bangalore (2006)
10. Li, W., Chan, E., Chen, D.: Energy-efficient cache replacement policies for cooperative caching in mobile ad hoc network. In: Paper presented at the Wireless Communications and Networking Conference, 2007, pp. 3347–3352. IEEE, Hong Kong (2007)
11. Haraty, R.A., Lana Turk, L.:. A comparative study of replacement algorithms used in the scalable asynchronous cache consistency scheme. In: Proceedings of the ISCA 19th International Conference on Computer Applications in Industry and Engineering, Las Vegas, USA (2006)
12. Haraty, Ramzi A., Zeitouny, Joe: Rule-based data mining cache replacement strategy. Int. J. Data Warehouse. Min. **9**(1), 56–69 (2013). doi:10.4018/IJDWM. ISSN: 1548-3924, EISSN: 1548-3932
13. Kahol, A., Khurana, S., Gupta, S.K.S., Srimani, P.K.: A strategy to manage cache consistency in a disconnected distributed environment. IEEE Trans. Parallel Distrib. Syst. **12** (7), 686–700 (2001)
14. Huang, Y., Jin, B., Cao, J., Sun, G., Feng, Y.: A selective push algorithm for cooperative cache consistency maintenance over MANETs. In: Kuo, T., Sha, E., Guo, M., Yang, L.T. (eds.) Lecture Notes in Computer Science, vol. 4808, pp. 650–660. Springer, Berlin (2007)