A Novel Cache Replacement Algorithm of EPCIS

Wei Chen
School of Information Engineering
Yancheng Institute of Technology
Yancheng, China
ee6789@outlook.com

Yongwang Gong
School of Information Engineering
Yancheng Institute of Technology
Yancheng, China
5448787@163.com

Zhi Gao
School of Information Engineering
Yancheng Institute of Technology
Yancheng, China
2131543231@163.com

Xiufang Xu
School of Information Engineering
Yancheng Institute of Technology
Yancheng, China
sdfsd@163.com

Xing Shao
School of Information Engineering
Yancheng Institute of Technology
Yancheng, China
5658687@163.com

Limao Han
School of Information Engineering
Yancheng Institute of Technology
Yancheng, China
23123dads@163.com

Abstract—To shorten the response time of EPCIS (Electronic Product Code Information Services) server, this paper proposes the structure of EPCIS based on proxy server and a novel cache replacement algorithm of EPCIS. A queue is built in the proxy server. The object which is more likely to be requested is moved to the head of queue, and the objects which have not been requested by now are moved to the tail of queue. The experimental results show that the new algorithm can efficiently shorten the response time of EPCIS system.

Keywords-EPCIS; proxy; cache replacement algorithm

I. INTRODUCTION

In the 21st century, the information technology is developed rapidly. The IoT (Internet of Things) is one of the most important parts of information technology [1]. Through the use of IoT, all the things on the world can be connected. As a result, a lot of information can be exchanged in IoT. Through the analysis of data, a lot of useful knowledge can be acquired by the researcher. Some advanced technologies are used in IoT, such as intellisense [2]. IoT has played an important role in network confluence, so it has been called the third wave of information industry, which is after computer and Internet [3].

EPCglobal is an industry commission and a nonprofit organization, responsible for the globalization of the EPC network standard. Its establishment is to have a more rapid, automatic, accurate good recognition in the supply chain. Since the establishment of EPCglobal, a lot of research works have been done by the researcher. The most remarkable thing is an IoT architecture, which makes the Iot easier to achieve.

The IoT architecture includes: web server, soap server, RFID, Object Naming Service, EPCIS (Electronic Product Code Information Services). Each industry can build its own IoT easily complying with the EPCglobal standard. Through the use of IoT, each industry can control its supply chain effectively. The logistics information of the goods can be conveniently used by the user. As a result, the entire firm's intelligence level will be greatly improved and the operating costs will be effectively reduced.

EPCIS system is an important part of IoT, which is proposed by EPCglobal [4]. All data information related with EPC (Electronic Product Code) is stored in EPCIS system, such as producer, logistics [5]. This information can be acquired by users through sending request to EPCIS system [6]. In order to reduce the user's waiting time, all requests need to be answered quickly by EPCIS system [7].

EPCIS server is composed by five components: web server, SOAP (Simple Object Access Protocol) engine, Service processing module, data source adapter module and data storage module [8]. The client's request is answered by web server [9]. The main disadvantage of this structure is: the number of users will increase dramatically with the rapid development of IoT, so the load of web server will be heavy. As a result, the response time of EPCIS system will be increased. In the worst case, EPCIS system will be paralyzed, and the user cannot get EPCIS service [10].

In order to make EPCIS system work in the right way, the structure of queueEPCIS and a novel cache replacement algorithm are proposed. The experimental

results show that queueEPCIS system can answer the client's requests efficiently.

The remainder of the paper is organized as follows. Section 2 describes the structure of queueEPCIS and cache replacement algorithm. Section 3 compares the results produced by queueEPCIS system with those produced by EPCIS system. Finally, Section 4 concludes the paper.

II. QUEUEEPCIS SYSTEM

A. the structure of queueEPCIS system

The structure of queueEPCIS system is shown in Fig. 1. The main idea of queueEPCIS system is: EPCIS proxy server is built, and put in the place near the client. Cache with large capacity is constructed in EPCIS proxy server, which is used to store the requested data. By using the queueEPCIS system, a lot of requests can be answered by EPCIS proxy server. As a result, the load of EPCIS server is lightened effectively. Because the distance between EPCIS proxy server and the client is shortened, the response time of EPCIS system is shortened remarkably.

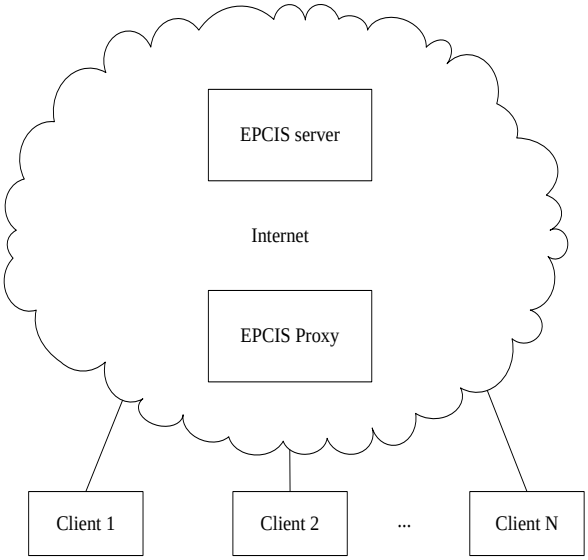

Figure 1. The structure of queueEPCIS

B. queueEPCIS algorithm

The main idea of queueEPCIS algorithm is: firstly, the data objects in EPCIS proxy are stored in a queue. The data object which is more likely to be requested is moved to the queue's head by calculation. The rest are moved to the tail of queue. If the data object has not been requested again, it will be removed out of the cache.

queueEPCIS algorithm:
Input: EPC code
Output: query result

1.flag=0

2.while (InCache(Object)=0) do

3. if CacheFull()=0 and LeftCapacity() ≧ Object.size then

4. Cache←Object

5. queue's length++

6. flag=1

7. end if

8.end while

9.for i=1 to queue's length do

10. if code=code(queue[i]) and flag=1 then

11. store queue[i] in queue's head

12. Number(queue[i])++

13. if i>1 then

14. queue[i] ←queue[i-1]

15. end if

16. return queue[i]

17. end if

18.end for

19.for i= queue's length to 2 do

20. queue[i] ←queue[i-1]

21.end for

22.store record with code in queue[1]

23.return queue[1]

The queue is traversed from the first line to the eighth line of queueEPCIS algorithm. If the proxy's cache can store the EPC information, it will be stored in the proxy. In the tenth line, the code of ith element in the queue is checked. If its value equals the value being searched, then the ith element in the queue is moved to the queue's head in the eleventh line. From the thirteenth line to the fifteenth line, the other elements in the queue are moved to queue's tail in turn. If the corresponding record is not found in the queue, then the (i-1)th element in the queue is moved to the location of the ith element from the nineteenth line to the twenty first line, and the element being searched is put in queue[1] in the twenty second line.

In the worst case, the queue is traversed by the circulation, so the time complexity of the algorithm is O(N), N is the length of the queue.

III. EXPERIMENTAL RESULTS

The experiments are conducted on windows xp sp2 operating system with 2GB memory.

In the first set of experiments, cpu frequency is 2.8Ghz, and the cache size in the proxy server is 300G. The average response time of two structures is compared at different times. The result of experiments is shown in Fig. 2. As can be seen from Fig. 2, on the average, the average response time of EPCIS system is 1.7 times longer than the structure of queueEPCIS system. Especially when ten minutes have past, the average response time of EPCIS system is 12 times longer than queueEPCIS. This is because more data information which users put more emphasis on can be stored in the cache of queueEPCIS system as time increases. As a result, requests sponsored by the client can be answered largely in the cache, which makes the average response time be reduced.

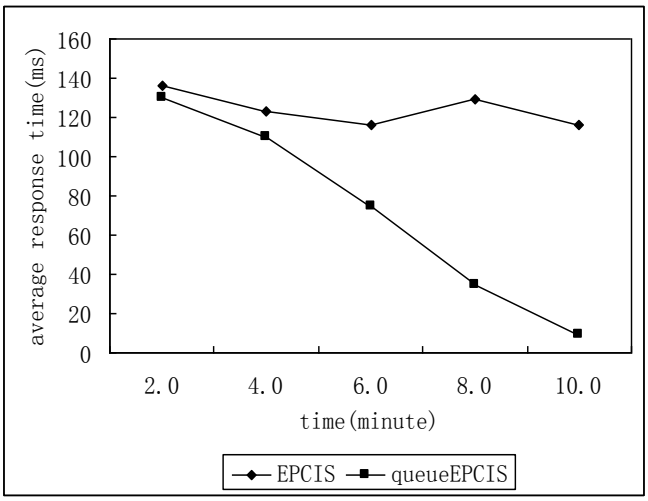

Figure 2. Two systems' average response time at different times

In the second set of experiments, cpu frequency is 2.8Ghz. The average response time of two systems with different sizes of cache space is compared. The result of experiments is shown in Fig. 3. As can be seen from Fig. 3, the average response time of EPCIS system is 1.3 times longer than queueEPCIS system with different cache space. This is because more data information is stored in queueEPCIS system as the cache space increases. As a result, the client's requests can be answered directly by the proxy server.

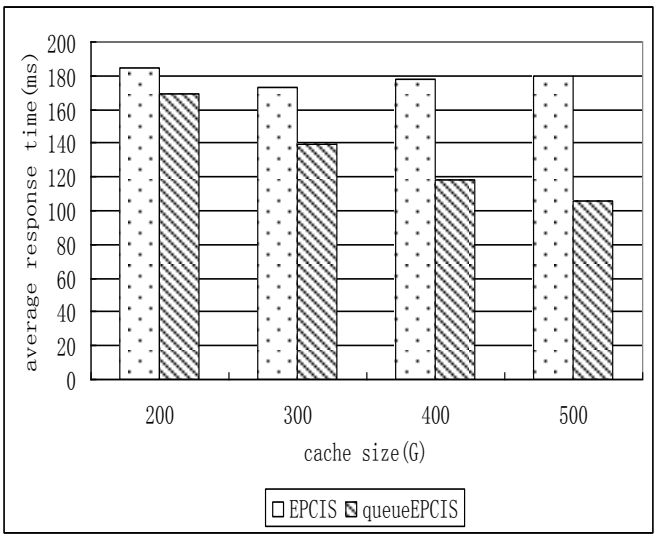

Figure 3. Two systems' average response time with different caches

In the third set of experiments, the cache space is set to 1000G. The average response time of two systems with different cpu frequencies is compared. The result of experiments is shown in Fig. 4. As can be seen from Fig. 4, the average response time of EPCIS system is 1.5 times longer than queueEPCIS system. This is because queueEPCIS system contains cache space, the higher cpu frequency is, the shorter the average response time is.

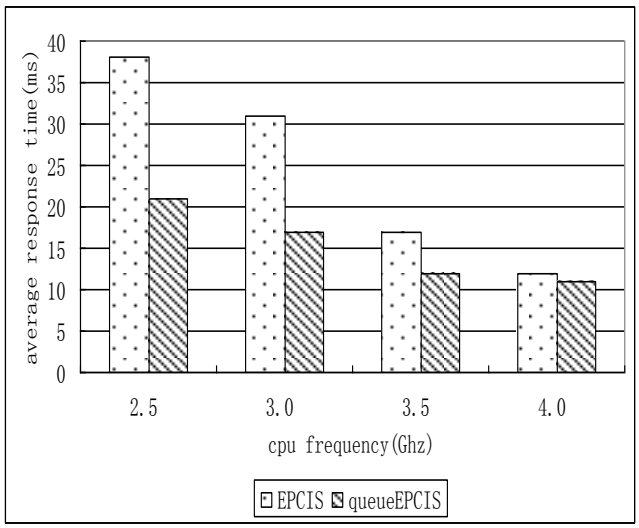

Figure 4. Two systems' average response time with different cpu frequencies

In the fourth set of experiments, the average response time of two systems with different data request rates is compared. The result of experiments is shown in Fig. 5. As can be seen from Fig. 5, the average response time of EPCIS system is 1.2 times longer than queueEPCIS system. This is because cache is contained in queueEPCIS system, and the hitting probability of cache space increases when the data request rate gets higher. As a result, the average response time is reduced.

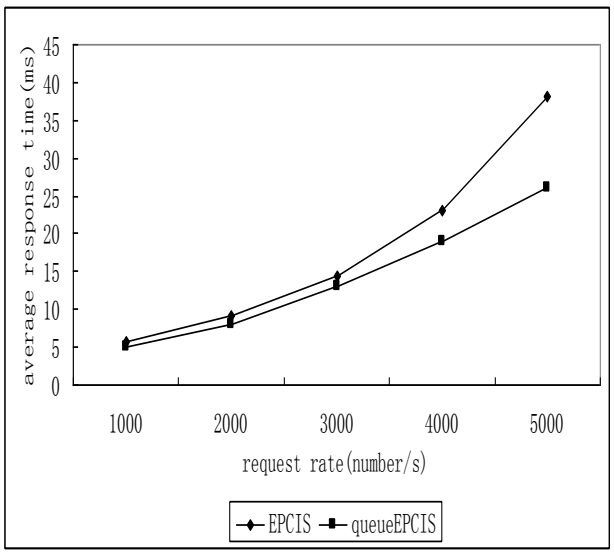

Figure 5. Two systems' average response time with different request rates

In the fifth set of experiments, the average response time of two systems with different memories is compared. The result of experiments is shown in Fig. 6. As can be seen from Fig. 6, the average response time of EPCIS system is 1.5 times longer than queueEPCIS system. This is because the bigger the momory is, the quicker the client's requests can be answered. As a result, the average response time is reduced.

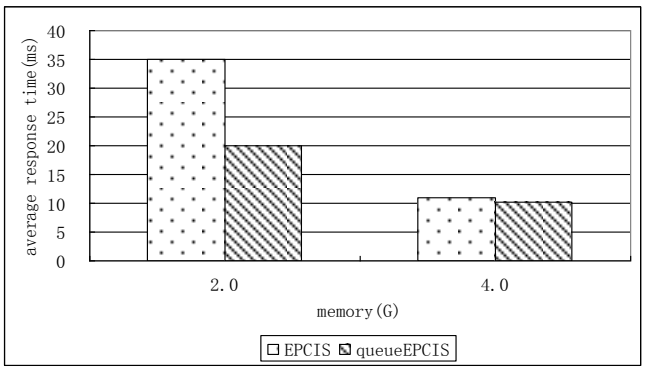

Figure 6. Two systems' average response time with different memories

In the sixth set of experiments, the average response time of two systems with different object sizes is compared. The result of experiments is shown in Fig. 7. As can be seen from Fig. 7, the average response time of EPCIS system is 1.2 times longer than queueEPCIS system.

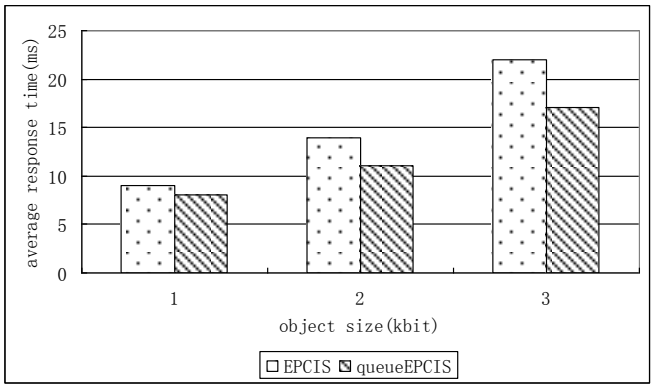

Figure 7. Two systems' average response time with different object sizes

In the seventh set of experiments, the average response time of two systems with different cpu L1 cache sizes is compared. The result of experiments is shown in Fig. 8. As can be seen from Fig. 8, the average response time of EPCIS system is 1.3 times longer than queueEPCIS system.

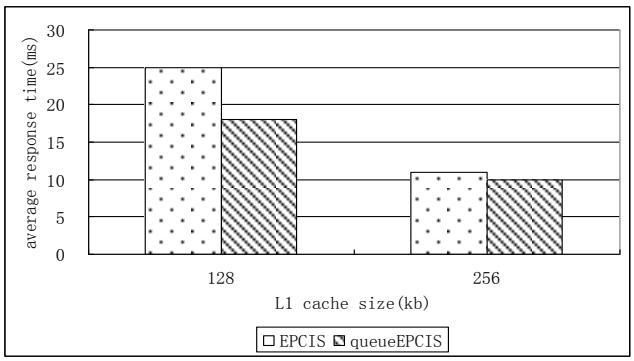

Figure 8. Two systems' average response time with different L1 cache sizes

IV. CONCLUSIONS

In this paper, the structure of queueEPCIS system based on proxy server is used, and new cache replacement algorithm is proposed. As a result, new EPCIS system is constructed. Data information requested frequently can be stored in the proxy server by this system, while those data information not requested frequently can be replaced out of the proxy server. The experimental results show that the response time is shortened effectively by queueEPCIS system. How to code the data information of queueEPCIS effectively will be researched in future.

Acknowledgment

The project is supported by the Department of Science and Technology of Jiangsu Province (No. BE2014679) and the Natural Science Research Project of Jiangsu Province (No. 15KJB520034).

References

[1] L. Atzori, A. Iera and G. Morabito, The Internet of Things: A survey, Computer Networks, vol.54, no.15, pp.2787-2805, 2010.

[2] L. Zhou, L. Song and C. Xie, Applications of Internet of Things in the facility agriculture, Computer and Computing Technologies in Agriculture VI, vol.392, no.1, pp.297-303, 2013.

[3] Y. Liu and G. Zhou, Key technologies and applications of Internet of Things, 2012 Fifth International Conference on Intelligent Computation Technology and Automation, Zhangjiajie, Hunan, pp.197-200, 2012.

[4] U. Barchetti and A. Implementation and testing of an EPCglobal-aware discovery service for item-level traceability, 2009 International Conference on Ultra Modern Telecommunications, Petersburg, pp.1-8, 2009.

[5] H. A. Ringsberg and V. Mirzabeiki, Effects on logistic operations from RFID- and EPCIS-enabled traceability, British Food Journal, vol.116, no.1, pp.104-124, 2014.

[6] W. Wang. Complex event processing in EPC sensor network middleware for Both RFID and WSN, 2008 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing, Orlando, pp.165-169, 2008.

[7] J. Muller, An aggregating discovery service for the EPCglobal network, 2010 43rd Hawaii International Conference on System Sciences, Honolulu, Hawaii, pp.1-9, 2010.

[8] D. Guinard and M. Mueller, Giving RFID a rest: Building a Web-Enabled EPCIS, Internet of Things, Tokyo, pp.1-8, 2010.

[9] S. Turchi, Designing EPCIS through linked data and rest principles, 2012 20th International Conference on Software,Telecommunications and Computer Networks, Split, pp.1-6, 2012.

[10] M. Schapranow, RFID event data processing: An architecture for storing and searching, The Fourth International Workshop on RFID Technology, Portugal, pp.1-15, 2010.