PressAcademia Procedia - (PAP), ISSN: 2459-0762

PressAcademia Procedia

2nd World Conference on Technology, Innovation and Entrepreneurship
May 12- 14, 2017, Istanbul, Turkey. Edited by Sefer Sener

ENHANCED WEB CACHE REPLACEMENT POLICY BASED ON DATA MINING AND RFSD SCORING

DOI: 10.17261/Pressacademia.2017.603
PAP-WCTIE-V.5-2017(41)-p.293-298

Mohammad Edris Rajaby, Tuğrul Çavdar
Computer Engineering Department, Karadeniz Technical University, Trabzon 61080, Turkey. edris.rajaby@gmail.com, ulduz@ktu.edu.tr

ABSTRACT

By rapid growth of the Internet users and devices, the number of servers also increase simultaneously which causes the exponential increment of the internet traffic and static data. Handling these huge amounts of user requests and efficiently responding to them require high bandwidth links, powerful servers and robust equipment, which despite the availability of these requirements getting the full user satisfaction is extremely difficult and a tough challenge. In order to overcome the mentioned problem, the cache servers are being used as a suitable solution. The performance of web cache server directly depends on its replacement policies. Several cache replacement policies have been proposed in literature each having varied hit rate (HR) and byte hit rate (BHR) performances on different networks. The replacement policy proposed in this paper is a dynamic cache replacement policy which trains itself utilizing previous network logs and by exploiting the data mining clustering algorithm. Once the training step is completed, the proposed policy utilizes the normalization formulas to score each metric of the enquiries including recency, frequency, size and delay. Simulation results showed that the proposed policy has the optimum performance on different networks and it not only improved the performance of web cache server in term of HR and BHR, but also decreases the data retrieval time (Delay Ratio (DR)) of the cache servers.

Keywords: cache replacement policy, cache server, proxy server, scoring

1. INTRODUCTION

The recent tremendous developments in the technology field had negative effects on the web performance in terms of network congestion, server load, data retrieval time and bandwidth consumption. To tackle the above issues, different solutions are available such as incrementing the link and server capacities and deploying backup servers in multiple locations. In addition to being very costly, none of the mentioned enhancements can provide a solid and optimum resolution. Cache servers have been implemented as an efficient solution that is capable of decreasing the data retrieval time, reducing the network congestion, lowering the server loads, preventing the need for high bandwidth links and eventually resulting in customer satisfaction (Davison, 2001) and (Podlipnig & Böszörmenyi, 2003).

The cache server deployment is shown in figure 1. The performance of web cache server directly depends on its replacement policies (Zhang, 2015).

Various web cache replacement policies are proposed by (Zhang, 2015), (Sulaiman, Mariyam Shamsuddin, Forkan, & Abraham), (Ali, Sulaiman, & Ahmad, 2014), (Jeon , Lee, Cho, & Ahn, 2003), (Jarukasemratana & Murata, 2013) and (Abdalla, Sulaiman, & Ali, 2015). Besides the cache HR and BHR that play important roles in web cache server performance, DR also has a significant impact on the performance of web cache servers. Furthermore, most of the traditional and proposed algorithms are based on the Recency such as LRU (Least Recently Used), Frequency such as LFU (Least Frequently Used) and Size factor. However, traditional policies being used currently are not sufficient as their decision depends on one factor while ignoring most others (Abdalla, Sulaiman, & Ali, 2015).

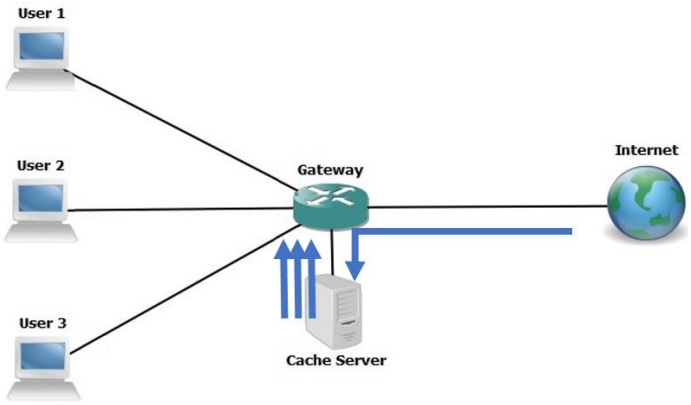

Figure 1: Deployment of cache server

Wong (2006), categorized the cache replacement policies into five categories and provided the design rationale along with the advantage of each category as bellow:

1- Recency-based: Recently requested objects will be requested again in the near future. It is good when the same object is requested by many users at the same time.
2- Frequency-based: The most popular objects should be cached. It is efficient when the web sites having objects with quite steady popularities are accessed by users.
3- Size-based: Keeps smaller objects by removing larger one. It is good when information-based web sites are accessed by users.
4- Function-based: Higher hit ratio can be achieved by considering more parameters. It has better performance when the system has high processing and memory resources.
5- Randomized: The simplest structure with no need for high computation overhead. It is good for systems with limited processing and memory resources.

According to the simulation results of the proposed policies in the literature, the performance of different policies varied on different datasets and can't guarantee the optimum performance on all networks. Furthermore, the reconfigurability of the replacement policy is irrecusable criteria. The replacement policy proposed in this paper is a dynamic cache replacement policy that analyses the network according to previous logs and afterward based on the analyzation result weighs and scores the RFSD metrics. Subsequently, using the metric weights and scores, a general score will be generated for each unique object; and based on this general score the object with the lowest score will be deleted from the cache server memory. The paper is organized as follow: section 1 provides overall information about this paper, presents related works done in literature and introduces the proposed solution. Section 2 is Methodology and explains the proposed methods in detail. Section 3 presents the simulation results and finally section 4 summarizes the paper.

2. METHODOLOGY

Data Preproccessing: The data collected from online resources usually has many errors and cannot be used for data analysis purposes directly due to its incompleteness, inconsistency and lack of specific behaviors and trends. Data preprocessing is a verified method for resolving such issues and it is a crucial step in the data mining procedure. Analyzing data without preprocessing can produce ineligible results. Data pre-processing consists of cleaning, normalization, transformation, feature extraction and selection, etc. The outcome of data pre-processing is the final training data set that could be used in data mining process (Kumar & Reddy, 2014).

In order to be accepted as a cacheable URL, we selected the URLs which not contain substrings such as 'cgi-bin' or '?', a file extension such as '.cgi', and if the response produced from origin server has a proper status code (e.g., 200 Success) (Arlitt, Cherkasova, Dilley, Friedrich, & Jin, 2000).

The flowchart of the proposed cache replacement policy is shown in figure 2. After preprocessing step the proposed policy perform bellow phases:

2.1. RFSD Scoring

The bellow formula (1) is used to score the Recency, Frequency and Delay (data retrieval time) metrics.

$$Score_{rfd} = 8 \times \left(\frac{(Val_{cur} - Val_{min})}{(Val_{max} - Val_{min})}\right) + 1 \ldots (1)$$

The above formula (1) is modified to score the Size metric of the objects. Unlike the RFD, the size metric is needed to be scored in inverse order. The smaller objects take the higher scores and the bigger objects take the lower scores.

$$Score_{s} = 8 \times \left(\frac{(Val_{max} - Val_{cur})}{(Val_{max} - Val_{min})}\right) + 1 \ldots (2)$$

In the formula (1) and (2) shown above, $Score_{rfd}$ is the score of the recency, frequency and delay metrics, $Score_{s}$ is the score of the size metric, Val_{max} is the maximum value of the corresponding metrics, Val_{cur} is the current objects metric value and the Val_{min} is the minimum value of the corresponding metrics. 8 and 1 are the constant numbers which were used to score the metrics between 1 and 9.

2.2. Policy Adjustment Using K-Means Algorithm

One of the common and popular clustering algorithms in Data Mining is the K-Means algorithm which uses the distance between each data point and the centroid of the cluster to partition the data points. In this method, the data points are assigned to the nearest centroid according to a specifically selected proximity measurement. Afterward, the centroids of all clusters are updated and these two steps are repeated until the centroids are stable. There are varied measurements in K-Means algorithm for computing the cluster centroids such as Manhattan distance, Euclidean distance, and Cosine similarity. Overall, the Euclidean measure is commonly used in K-means algorithm (Aggarwal & Reddy, 2013). Euclidean distance was used in this paper according to formula (3).

$$d(m,n) = \sqrt{|m_1 - n_1|^2 + |m_2 - n_2|^2 + \cdots + |m_i - n_i|^2} \ldots (3)$$

Where *m* and *n* are the data points and *d* is the distance between them.

To train the proposed algorithm based on the previous logs, the number of clusters K was defined as one cluster. The algorithm should calculate the mean of the cluster by calculating the mean for each variable (RFSD) as illustrated in table 1.

Table 1: Calculated means for cluster

Variables	Dataset (a)	Dataset (b)
Recency	6.453	7.054
Frequency	5.002	6.765
Size	7.012	4.343
Delay	4.234	3.243

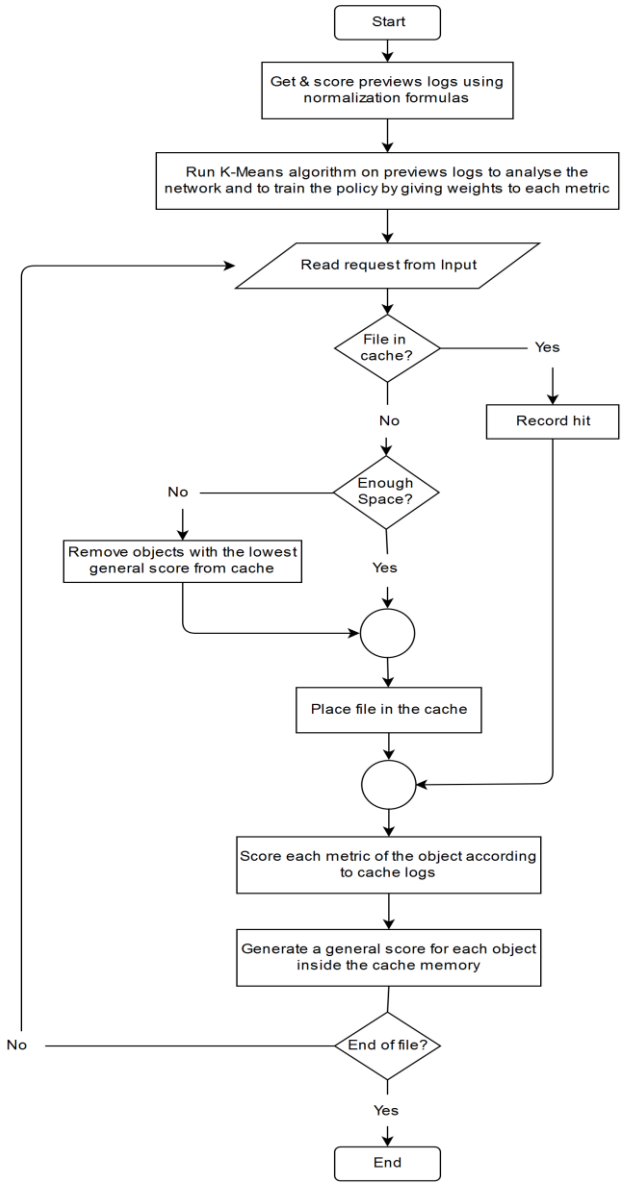

Figure 2: Flowchart of the proposed policy

The proposed algorithm weighs the metrics according to the means calculated by the K-means clustering algorithm. For example based on the above results in table 1 for the dataset (a) the algorithm weighs the size as 10^4, the recency as 10^3, the frequency as 10^2 and the delay as 10. If an object has the scores of recency= 8, frequency= 7, size= 9 and delay equal=5, the general score generated by policy for this object should be 9875.

3. SIMULATION RESULTS

The dataset used in this experiment was the BU Web traces (Boston University, 1995) dataset provided by Cunha of Boston University. This dataset is contained of 9633 files, and recorded 1,143,839 web requests from different clients during six months. BU traces consist of 37 client machines divided into two sets: undergraduate students set (called 272 set) and graduate students set (called B19 set). In this work, the dataset 272 is called as the dataset (a) and the dataset B19 as the dataset (b). The dataset (b) has 32 machines and the dataset (a) has 5 machines (Ali & Siti, 2009). In this experiment, one day logs (13-December) of the dataset (a) was used to train the policy and one day logs (14-December) of the dataset (a) to test the performance of the policy. Furthermore, to prove that the proposed policy has the optimum performance on all networks the policy has been tested on the second dataset (database (b)). In the second dataset, one day traces (17-January) was used to train the policy and one day traces (18-January) to test the performance of the policy.

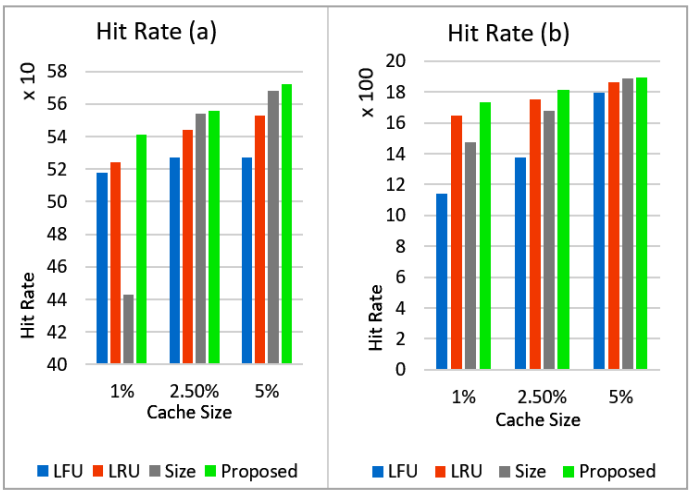

Figure 3: Hit rate of the policies for dataset (a) and dataset (b)

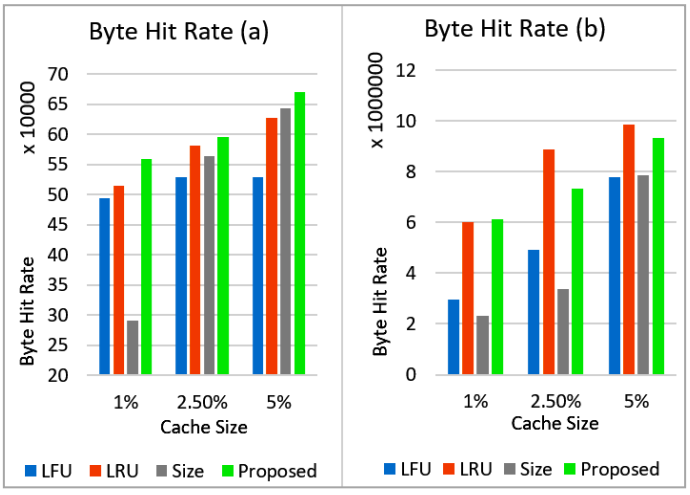

Figure 4: Byte hit rate of the policies for dataset (a) and (b)

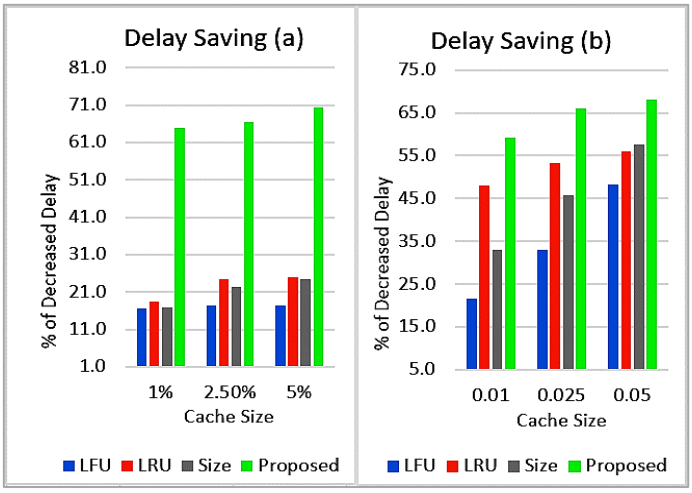

Figure 5: Delay saving of the policies for dataset (a) and dataset (b)

In cases where the size of the cache is very large, almost all policies have good performance. The best policy is the one that has the best performance in case of limited and small cache size. According to the simulation results as demonstrated in figurs. 3, 4 and 5, the proposed cache replacement policy is compared with the conventional algorithms such as LRU (Least Recently Used), LFU (Least Frequently Used) and Size. As shown in figure 3. the proposed policy has the best performance in datasets (a) and (b) in term of HR, and based on figure 4. the proposed policy provides the best performance for BHR as well in cases that the cache size is small. Moreover, as proved in figure 5. one of the significant competencies that the proposed algorithm has is its terrific increment of cache server performance in terms of delay ratio compared to traditional algorithms such as LFU, LRU and Size.

5. CONCLUSION

Cache servers play a significant role to improve the performance of the Internet, and the performance of cache servers directly depends on the cache replacement policies used in them. In this work, a new cache replacement policy has been proposed which is based on the data mining k-means clustering algorithm and RFSD scoring method. Numerous replacement policies have been proposed in literature, but most of them used one or two metrics of the requested objects, their performance on various networks differs and they do not provide solid and optimum solution on all networks. Using the k-means algorithm and previous network logs, the proposed policy is capable of adjusting itself to any network structure. It scores the RFSD metrics by modifying the normalization formulas and generates a general score for each object. Subsequently, the object with the lowest general score will be deleted when the cache is full and the new requested object will be cached.

The performance analysis shows that one of the significant competencies that the proposed algorithm has is its terrific increment of cache server performance in terms of delay ratio compared to traditional algorithms such as LFU, SIZE and LRU. Concurrently, it is demonstrated that the proposed policy remarkably improves the performance of the cache servers with regard to HR and BHR on any network.

REFERENCES

Abdalla, A., Sulaiman, S., & Ali, W. (2015). Intelligent Web Objects Prediction Approach in Web Proxy Cache Using Supervised Machine Learning and Feature Selection. Int. J. Advance Soft Compu. Appl, 7(3).

Aggarwal, C. C., & Reddy, C. K. (2013). Data clustering: algorithms and applications. Boca Raton: CRC Press.

Ali, W., & Siti, M. (2009). Intelligent client-side web caching scheme based on least recently used algorithm and neuro-fuzzy system. International Symposium on Neural Networks. Berlin.

Ali, W., Sulaiman, S., & Ahmad, N. (2014). Performance Improvement of Least-Recently-Used Policy in Web Proxy Cache Replacement Using Supervised Machine Learning. International Journal of Advances in Soft Computing & Its Applications, 6(1).

Arlitt, M., Cherkasova, L., Dilley, J., Friedrich, R., & Jin, T. (2000). Evaluating content management techniques for web proxy caches. ACM SIGMETRICS Performance Evaluation Review, 27(4), 3 - 11.

Boston University, (1995). BU Web Trace, http://ita.ee.lbl.gov/html/contrib/BU-Web-Client.html.

Davison, B. (2001). A Web caching primer. IEEE internet computing, 38 - 45.

Jarukasemratana, S., & Murata, T. (2013). Web Caching Replacement Algorithm Based on Web Usage Data. New Generation Computing, 31(4), 311–329.

Jeon , J., Lee, G., Cho, H., & Ahn, B. (2003). A prefetching Web caching method using adaptive search patterns. 2003 IEEE Pacific Rim Conference on Communications Computers and Signal Processing (PACRIM 2003) (Cat. No.03CH37490) . IEEE.

Kumar, P., & Reddy, D. (2014). Novel Web Proxy Cache Replacement Algorithms using Machine Learning Techniques for Performance Enhancement. INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY, 3(1), 339-346.

Podlipnig, S., & Böszörmenyi, L. (2003). A survey of Web cache replacement strategies. ACM Computing Surveys (CSUR), 374-398.

Sulaiman, S., Mariyam Shamsuddin, S., Forkan, F., & Abraham, A. (n.d.). Intelligent Web caching using neurocomputing and particle swarm optimization algorithm. 2008 Second Asia International Conference on Modelling & Simulation (AMS). IEEE.

Wong, K.-Y. (2006). Web cache replacement policies: a pragmatic approach. IEEE Network, 20(1), 28-34.

Zhang, J. (2015). Replacement Strategy of Web Cache Based on Data Mining . 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC).