Performance comparison of cache replacement algorithms on various internet traffic

Mulki Indana Zulfa1,*, Adhistya Erna Permanasari2, Ari Fadli3, Waleed Ali4
1,3Department of Electrical Engineering, Jenderal Soedirman University
2Department of Electrical Engineering and Information Technology, Gadjah Mada University
4Department of Information Technology, King Abdulaziz University
1,3Jl. Mayjen Sungkono, Blater, Purbalingga 53371, Indonesia
2Jl. Grafika, No.2, Sinduadi, Sleman 52281, Indonesia
4Faculty of Computing and Information, Rabigh 21911, Saudi Arabia
*Corresponding email: mulki_indanazulfa@unsoed.ac.id

Received 19 December 2022, Revised 10 January 2023, Accepted 1 February 2023

Abstract — Internet users tend to skip and look for alternative websites if they have slow response times. For cloud network managers, implementing a caching strategy on the edge network can help lighten the workload of databases and application servers. The caching strategy is carried out by storing frequently accessed data objects in cache memory. Through this strategy, the speed of access to the same data becomes faster. Cache replacement is the main mechanism of the caching strategy. There are seven cache replacement algorithms with good performance that can be used, namely LRU, LFU, LFUDA, GDS, GDSF, SIZE, and FIFO. The algorithm is developed uniquely according to the internet traffic patterns encountered. Therefore, a particular cache replacement algorithm cannot be said to be superior to other algorithms. This paper presents a performance comparison simulation of the seven cache replacement algorithms on various internet traffic extracted from the public IRcache dataset. The results of this study indicate that the Hit Ratio (HR) performance is strongly influenced by cache size, cacheable and unique requests. The smaller the unique request that occurs, the greater the HR performance obtained. The LRU algorithm shows a very good HR performance to perform cache replacement work under normal internet conditions. However, when the access impulse phenomenon occurs, the GDSF algorithm is superior in obtaining HRs with limited cache memory capacity. The simulation results show that GDSF reaches a 50.75% HR while LRU is only 49.17% when access anomalies occur.

Keywords – cache memory, cache replacement, caching strategy, hit ratio

Copyright ©2023 JURNAL INFOTEL
All rights reserved.

I. INTRODUCTION

The economic growth of a nation is caused by the rapid development of the internet, which makes it easier to explore information and adds to the insights of economic actors [1], [2]. Internet technology also causes shifts in individual lifestyles that can improve the population’s welfare and strengthen the institutionalization of village communities [3], [4]. The number of internet and digital technology users has increased since the COVID-19 pandemic [5], [6], especially e-learning and e-banking users [7], [8]. In addition, data transactions in the internet world are getting bigger as 5G technology, edge computing, and the Internet of Things (IoT) evolve [9], [10].

Internet users tend to go through and look for alternative websites if they have a slow response time [11], [12]. Response time is the time it takes to measure the speed at which the browser fully loads a web page since it was first clicked [13], [14]. Response time is a problem that must be considered by internet-based application developers and cloud network infrastructure managers as it can improve the experience and comfort of their users [15], [16].

For internet-based application developers, one solution to overcome the problem of response time is to scale out applications using cache memory or In-memory Database (IMDB) [17], [18]. For cloud network managers, implementing caching strategies on the edge network can help ease the workload of
Table 1. Statistics of Four IRcache Datasets

<table>
<thead>
<tr>
<th>No.</th>
<th>Info</th>
<th>BO2</th>
<th>SV</th>
<th>UC</th>
<th>NY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Proxy location</td>
<td>Boulder, Colorado</td>
<td>Silicon Valley, California</td>
<td>Urbana Champaign</td>
<td>New York</td>
</tr>
<tr>
<td>2.</td>
<td>Total requests</td>
<td>666</td>
<td>666</td>
<td>666</td>
<td>666</td>
</tr>
<tr>
<td>3.</td>
<td>Cacheable requests</td>
<td>110</td>
<td>87</td>
<td>141</td>
<td>74</td>
</tr>
<tr>
<td>4.</td>
<td>Cacheable bytes</td>
<td>57629</td>
<td>44480</td>
<td>71383</td>
<td>37218</td>
</tr>
<tr>
<td>5.</td>
<td>Unique requests</td>
<td>386</td>
<td>527</td>
<td>388</td>
<td>267</td>
</tr>
</tbody>
</table>

II. RESEARCH METHOD

This section discusses research scenarios, cache replacement algorithms, IRcache dataset, and Hit Ratio (HR).

A. Research Scenarios

The contribution of this paper is to present the simulation results of comparing the HR performance of seven cache replacement algorithms on various internet traffic using the IRcache dataset consisting of four sub-datasets. Each sub-datasets consists of 666 records. Various types of internet traffic conditions are known from statistics on total requests, cacheable bytes, cacheable requests, and unique requests, as presented in Table 1.

Seven algorithms were run to simulate the cache replacement mechanism in four IRcache datasets. The performance evaluation of the algorithm is calculated using the HR by calculating the percentage of request data served successfully by the cache memory compared to the total request data. The work of the seven cache replacement algorithms, the IRcache dataset, and the HR are described in more detail in section II-B, section II-C, and section II-D, respectively.

B. Cache Replacement Algorithms

LFU and LRU are the most famous cache replacement algorithms with good HR performance. If there is a new cached data storage request in the cache memory, the LFU algorithm will first delete the cached data with a minor access count. At the same time, the LRU algorithm will delete the most recently used cached data. LRU algorithm performance is strongly influenced by access recency, while LFU is influenced by access count or the number of accesses to cached data. Arlitt et al. [42] then developed LFU by adding a wear factor called Dynamic Aging. Eq. (1) is used to calculate the aging factor \( L \) represented in the key-value variable \( K_{(g)} \). A variable \( F_{(g)} \) is an access count value of cached data that must be deleted to be replaced with new cached data entering the cache memory.

\[
K_{(g)} = L + F_{(g)} \tag{1}
\]

The SIZE algorithm will erase the cached data with the largest cache size, while FIFO will erase the cached data with the smallest cache size.
data in the order in which it first entered the cache memory.

LFU, LRU, SIZE, and FIFO algorithms do not consider network costs or other costs that affect caching decisions. Therefore, Cao et al. [43] propose a new key-value calculation by including the cost-aware variable. Eq. (2) is used to calculate $K_i(g)$ the value of the GDS algorithm.

$$K_i(g) = L + \frac{C_i(g)}{S_i(g)} \quad (2)$$

Parameter $L$ shows the running queue, its value starts from 0 and is updated with the min $K$ value of the lowest value in priority queue. Parameter $S_i(g)$ is the size of document $i$. Parameter $C_i(g)$ is the cost associated with bringing document $i$ into the cache.

In its development, the access count variable is necessary so that caching decisions can be taken more comprehensively by considering the cost awareness and popularity based on the value of the access count. Cherkasova et al. [44], therefore, proposed improving the performance of the GDS by proposing the GDSF. Eq. (3) calculates the key value of the GDSF.

$$K_i(g) = L + F_i(g) \times \frac{C_i(g)}{S_i(g)} \quad (3)$$

C. IRcache Dataset

This paper used the IRcache dataset to synthesize various internet traffic. The IRcache dataset was first released in 1999 by Alex Rousskov [45], who was later managed by the National Laboratory of Applied Network Research (NLANR) in the United States. The IRcache dataset is an accurate world proxy drawn from proxy servers spread across five the United States cities, namely Urbana-Champaign (UC), Boulder (BO2), Silicon Valley (SV), San Diego (SD), and New York (NY).

The IRcache dataset is perfect for simulating the cache replacement mechanism in the data caching process in cache memory. The IRcache dataset also represents the characteristics of internet traffic in actual conditions. In the last five years, this IRcache dataset is still used by Ibrahim et al. [46] in cache replacement research and Li et al. [47] in content caching optimization research. Table 2 presents the properties contained in the IRcache dataset.

<table>
<thead>
<tr>
<th>No.</th>
<th>Information</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Timestamp</td>
<td>The time when the data request.</td>
</tr>
<tr>
<td>2</td>
<td>Elapsed time</td>
<td>Time difference between connection accept and close.</td>
</tr>
<tr>
<td>3</td>
<td>Client address</td>
<td>The destination of target IP address.</td>
</tr>
<tr>
<td>4</td>
<td>HTTP code</td>
<td>HTTP status code.</td>
</tr>
<tr>
<td>5</td>
<td>Size</td>
<td>The number of data Bytes received.</td>
</tr>
<tr>
<td>6</td>
<td>Request method</td>
<td>HTTP request method.</td>
</tr>
<tr>
<td>7</td>
<td>URL</td>
<td>The destination of target URL.</td>
</tr>
<tr>
<td>8</td>
<td>Content type</td>
<td>HTTP content type.</td>
</tr>
<tr>
<td>9</td>
<td>Hierarchy data</td>
<td>The hierarchy of data request.</td>
</tr>
<tr>
<td>10</td>
<td>User ID</td>
<td>The user ID.</td>
</tr>
</tbody>
</table>

D. Hit Ratio

HR is the percentage of data service requests successfully served by cache memory ($r_i$) compared to the total data requests it receives ($N$). One cache hit indicates that one data request was successfully served by cache memory. If this condition occurs, then $r_i$ would be worth 1. The opposite of cache hit is cache miss, a condition if a request for data cannot be served by cache memory, so the requested data will be forwarded to the origin server. When the requested data has been obtained from the origin server, this data will be immediately stored in the cache memory. The value of the hit cache in that condition remains one, but the total number of data requests ($N$) is already two. Therefore, the resulting HR value of $\frac{1}{2} \times 100% = 50\%$ in this condition. Eq. (4) is used to calculate the HR value.

$$HR = \frac{\sum_{i=1}^{N} r_i}{N} \quad (4)$$

III. Result

Table 3 shows the average HR performance of seven cache replacement algorithms run on four IRcache sub-datasets. Based on Table 3, the most significant HR performance measurement is achieved in the NY dataset. In this dataset, the minimum HR performance is 48.41%, obtained by the FIFO algorithm, while the maximum HR performance is 50.75%, obtained by the GDSF algorithm. The SV dataset became the smallest in terms of HR performance. In this dataset, the maximum HR performance is 7.43%, obtained by the GDSF algorithm, while the minimum HR performance is 6.40%, obtained by the SIZE algorithm. The best HR values in Table 3 are marked in bold print, while the worst performances are marked in italic print.

The HR performance of the GDSF algorithm was superior in two datasets, namely SV and NY, while the LRU algorithm was superior in the BO2 and UC.
datasets. The LFU algorithm obtains the worst HR performance in two datasets: SV and UC. At the same time, the worst performance in the other datasets was obtained by the SIZE algorithm in the BO2 dataset and the FIFO algorithm in the NY dataset. Algorithms based on access recency, such as LRU, tend to be superior in the BO2, SV, and UC datasets, while algorithms based on key-value and access counts, such as GDSF LFU and LFUDA, are superior in the NY dataset.

Table 2 and Table 3 show the relationship. The smaller the cacheable and unique request that occurs, the greater the resulting HR performance. It is shown based on the performance HR in the NY dataset, which is superior to the other three datasets.

Table 3. The Average HR Performance

<table>
<thead>
<tr>
<th>No.</th>
<th>Algorithm</th>
<th>BO2</th>
<th>SV</th>
<th>UC</th>
<th>NY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>FIFO</td>
<td>20.87</td>
<td>6.94</td>
<td>31.88</td>
<td>47.99</td>
</tr>
<tr>
<td>2.</td>
<td>GDS</td>
<td>18.62</td>
<td>6.91</td>
<td>31.68</td>
<td>48.41</td>
</tr>
<tr>
<td>3.</td>
<td>GDSF</td>
<td>21.85</td>
<td>7.43</td>
<td>28.86</td>
<td>50.75</td>
</tr>
<tr>
<td>4.</td>
<td>LFU</td>
<td>21.52</td>
<td>7.57</td>
<td>25.78</td>
<td>49.28</td>
</tr>
<tr>
<td>5.</td>
<td>LFUDA</td>
<td>21.94</td>
<td>7.22</td>
<td>28.89</td>
<td>50.36</td>
</tr>
<tr>
<td>6.</td>
<td>LRU</td>
<td>22.21</td>
<td>7.33</td>
<td>32.60</td>
<td>49.17</td>
</tr>
<tr>
<td>7.</td>
<td>SIZE</td>
<td>17.42</td>
<td>6.4</td>
<td>25.5</td>
<td>49.34</td>
</tr>
</tbody>
</table>

In addition, the correlation between Table 2 and Table 3 also shows that the greater the unique request that occurs, the smaller the resulting HR performance. The more significant number of unique requests causes the lesser probability of saving cached data in the cache memory.

Fig. 1 to Fig. 4 shows the relationship between cache size and the resulting HR performance. The performance of the HR is directly proportional to the cache size. The replacement cache mechanism is more common in cache size with a small size. It causes not many cacheable requests to be stored, so the smaller the chance of a cache hit occurs. A rare hit cache causes the hit performance ratio to be low.

Based on Fig. 1 to Fig. 4, the GDSF algorithm achieves the highest HR in the NY dataset of 50.75%, while the FIFO algorithm HR value is the lowest at 47.90%. In the UC dataset, the LRU algorithm is the best with a HR of 32.60%, while LFU is the worst with a HR of 23.78%. In the SV dataset, the highest HR was achieved by the GDSF algorithm of 7.43%, while the lowest value was obtained by the LFU of 5.57%. Finally, the simulation results on the BO2 dataset show that the LRU algorithm is the best with a 22.21% HR, while the SIZE algorithm is the worst with a 17.42% HR.

The SIZE algorithm gives higher priority to small size cached data to occupy cache memory. It was intended that more cached data can be stored. However, the simulation results show that this strategy is not
effective in increasing the HR performance. It happens because those small-size cached are rarely accessed, then such cached data is often replaced with other data.

Fig. 1 to Fig. 4 also shows that the NY dataset is unfavorable for access recency-based cache replacement algorithms such as LRU. The LRU algorithm can excel over the HR performance in three datasets but not in the NY dataset. Based on Fig. 1 to Fig. 4, key-value and access frequency-based algorithms such as GDSF, LFU, and LFUDA obtained superior HR performance than the other four algorithms. Caching decisions based on the calculation of access count data on datasets with low unique request statistics will produce more cache hits because each cached data has the same caching probability. This fact is significantly inversely proportional to the SV dataset with the unique requests. Each cached data has a small caching probability in cache memory with minimal size because its position in the cache memory may be quickly replaced with other cached data.

Based on the HR performance in Fig. 1 to Fig. 4, the access pattern that occurs in the NY dataset is an anomaly. This small number of cacheable and unique requests illustrates that there has been impulse access at a particular time that only accesses a small portion of cached data. It is the leading cause of cacheable and unique request values in the NY dataset to be the smallest. If further attention is paid, this phenomenon of impulse access only occurs briefly. This phenomenon can be avoided by providing a cache memory of a larger size. Fig. 1 to Fig. 4 shows that the 150 (KB) cache size configuration can produce the same enormous HR performance (59.91%) in the seven existing cache replacement algorithms. This condition has never been found in the other three datasets, namely Urbana-Champaign (UC), Boulder (BO2), Silicon Valley (SV), San Diego (SD).

IV. DISCUSSION

Proxy servers generally utilize cache replacement algorithms to organize popular web-object storage in cache memory so that subsequent web-object requests can be served more quickly. In typical internet traffic conditions and during impulse access, key-value-based and cost-aware algorithms such as GDSF can maintain good HR performance, as shown in the simulation of NY and SV datasets. However, the LRU algorithm can be relied on in regular internet traffic, such as the HR performance results in the BO2 and UC datasets. Some researchers develop GDSF algorithms into WGDSSF [48] or WSCR [34] with quite convincing HR performance results. LRU algorithm has also been developed using metaheuristic optimization methods [49] or machine learning [50]. Both show better HR performance compared to the standard LRU version.

In addition to being used as a proxy server, the cache replacement algorithm discussed in this paper can also be adapted to perform caching on the server database. Complex query requests with joins to multiple tables can be stored in the memory buffer to speed up the same query request later. Caching the Cachematic framework [51] in caching the query result by building an abstract syntax tree to identify "where" and "join" clauses on each target query submitted by the user. The cache replacement mechanism can also be specifically utilized directly on in-memory databases, such as Redis [52] or Memcached [53], to store small-sized data objects that users often access.

V. CONCLUSION

The HR performance is strongly influenced by cache size, cacheable and unique request. The smaller the unique request, the greater the performance HR obtained. LRU algorithm shows excellent HR performance for cache replacement work in normal internet conditions. However, when there is an impulse access phenomenon, the GDSF algorithm is superior in acquiring a HR on limited cache memory capacity. The GDSF algorithm is more flexible to handle internet traffic under normal conditions and when access anomalies occur.

However, the LRU algorithm can be relied on in regular internet traffic, such as the HR performance results in the BO2 and UC datasets. The SIZE algorithm gives higher priority to small size cached data to occupy cache memory. It was intended that more cached data can be stored. However, the simulation results show that this strategy is not effective in increasing the HR performance. GDSF, LFU, and LFUDA obtained superior HR performance than the other four algorithms on dataset with high the unique requests.

REFERENCES


[41] M. I. Zulfa, R. Hartanto, and A. Erna Permanasari, Cached data offload framework based on hybrid least recently used and metaheuristic optimization methods, Gadjah Mada University, 2022.


