JURNAL INFOTEL
Informatics - Telecommunication - Electronics
Website: http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN: 2085-3688; e-ISSN: 2460-0997

Performance comparison of cache replacement algorithms on various internet traffic

Mulki Indana Zulfa[1,*], Adhistya Erna Permanasari[2], Ari Fadli[3], Waleed Ali[4]
[1,3]Department of Electrical Engineering, Jenderal Soedirman University
[2]Department of Electrical Engineering and Information Technology, Gadjah Mada University
[4]Department of Information Technology, King Abdulaziz University
[1,3]Jl. Mayjen Sungkono, Blater, Purbalingga 53371, Indonesia
[2]Jl. Grafika, No.2, Sinduadi, Sleman 52281, Indonesia
[4]Faculty 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

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

No.	Info	BO2	SV	UC	NY
1.	Proxy location	Boulder, Colorado	Silicon Valley, California	Urbana Champaign	New York
2.	Total requests	666	666	666	666
3.	Cacheable requests	110	87	141	74
4.	Cacheable bytes	57629	44480	71583	37218
5.	Unique requests	386	527	388	267

databases and application servers [19], [20]. Caching strategies can also save bandwidth and reduce network latency [21], [22]. The caching strategy is carried out by storing data objects accessed often at specific locations [23], [24]. Through this strategy, the same speed of data access becomes faster because the data has been stored before [25]. The right caching strategy can reduce the exact data requests to the origin server to speed up the response time on the client side.

Caching strategy is a relatively broad field of research because this strategy is widely used in several areas such as cloud networks [19], [26], browser applications [27], [28], operating systems [29], embedded systems [30], mobile network [31], [32], and telecommunications [20]. The primary mechanism of caching strategy is cache replacement, changing the content of cached data in cache memory because of its limited capacity [33], [34]. Researchers have developed several cache replacement algorithms, such as Least Recently Used (LRU), Least Frequently Used (LFU), Least Recently Used Dynamic Aging (LFUDA), Greedy-Dual Size (GDS), Greedy-Dual Size Frequency (GDSF), SIZE, and First-in-First-out (FIFO). The seven algorithms are generally used in the benchmarking process in caching strategy research. Practically some of these algorithms have also been adopted in caching systems contained in proxy servers such as Squid and Varnish.

The selection of the cache replacement algorithm must be carried out comprehensively so that it does not reduce the application's performance or cause traffic congestion [35]. The researchers uniquely developed the cache replacement mechanism following the pattern of internet traffic [36]. Therefore, specific cache replacement algorithms cannot be considered superior to other cache replacement algorithms [37], [38].

This paper aims to compare the performance of the seven caching algorithms on different internet access patterns. The internet traffic pattern tends to be random, following Zipf's law [39], [40]. However, in certain conditions, impulse access can occur like a viral phenomenon that occurs quickly and then gradually returns to normal [41]. Therefore, this paper can be an essential reference for researchers caching strategies to develop new cache replacement algorithms based on the internet access patterns they face.

For a better understanding, the rest of this paper is organized as follows. In section II, we provide the research method, followed by the result in section III. Section IV presents the discussion of the result. Finally, we provide the conclusion in section V.

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

Table 2. The Information Contained in the Cache Dataset

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

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 $K_{(g)}$ the value of the GDS algorithm.

$$K_{(g)} = L + \frac{C_{(g)}}{S_{(g)}} \tag{2}$$

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 $S_{(g)}$ is the size of document i. Parameter $C_{(g)}$ 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.

Parameters L, $C_{(g)}$ and $S_{(g)}$ are the same as those parameters in the GDS algorithm. Parameter $F_{(g)}$ indicates the number of documents (g)

$$K_{(g)} = L + F_{(g)} \times \frac{C_{(g)}}{S_{(g)}} \tag{3}$$

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.

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 $\frac{1}{2}\times100\%$ = 50% in this condition. Eq. (4) is used to calculate the HR value.

$$\text{HR} = \frac{\sum_{i=1}^{N} r_i}{N} \tag{4}$$

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

No.	Algorithm	BO2	SV	UC	NY
1.	FIFO	20.87	6.94	31.88	*47.9*
2.	GDS	18.62	6.91	31.68	48.41
3.	GDSF	21.85	**7.43**	28.86	**50.75**
4.	LFU	21.52	*5.57*	*23.78*	49.28
5.	LFUDA	21.94	7.22	28.89	50.36
6.	LRU	**22.21**	7.3	**32.6**	49.17
7.	SIZE	*17.42*	6.4	25.5	49.54

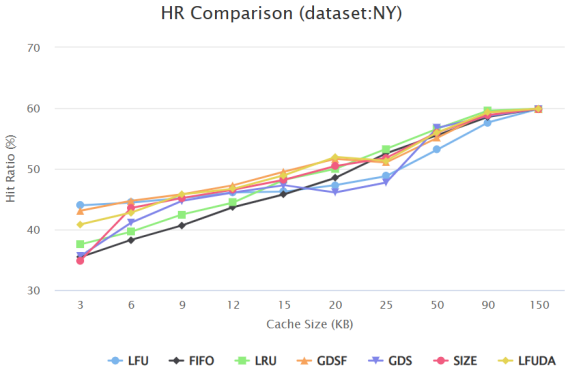

Fig. 1. The HR performance on four IRcache datasets NY.

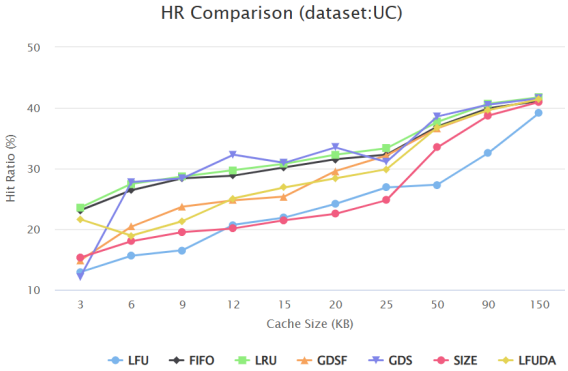

Fig. 2. The HR performance on four IRcache datasets UC.

Urbana-Champaign (UC), Boulder (BO2), Silicon Valley (SV), San Diego (SD), and New York (NY) are five cities in the United States where proxy servers are located. The proxy log retrieved from the server is the IRcache Dataset. This dataset is suitable for testing the cached data offloading method because it can illustrate the characteristics of internet traffic in real conditions.

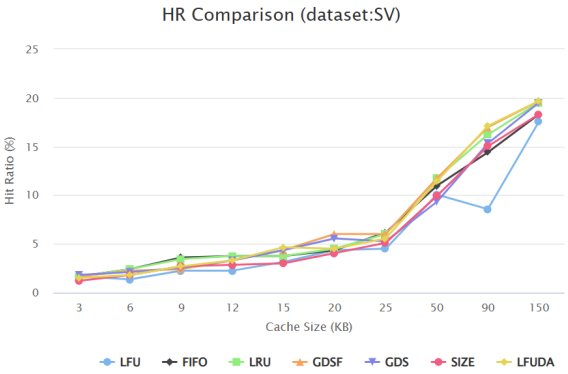

Fig. 3. The HR performance on four IRcache datasets SV.

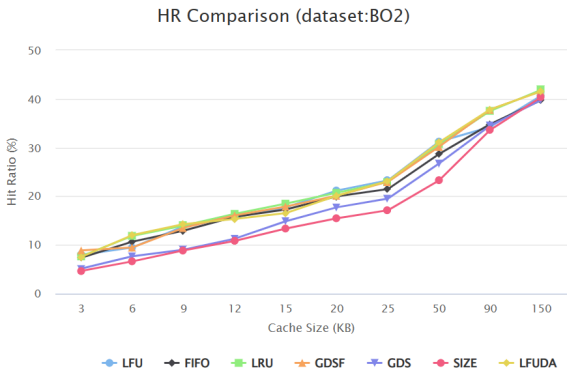

Fig. 4. The HR performance on four IRcache datasets BO2.

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 WGDSF [48] or WSCRP [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

[1] T. Mariyati, Pembangunan desa dengan memanfaatkan strategi pemerataan akses internet dan penyebaran informasi, *Bul. Pos dan Telekomun.*, vol. 7, no. 3, pp. 2550, 2009.

[2] T. Mariyati, Public policy implementation strategy in encouraging acceleration of internet users development, *Bul. Pos dan Telekomun.*, vol. 11, no. 2, pp. 147158, 2013.

[3] T. Mariyati, Efek pertumbuhan ekonomi dalam proses pengembangan telekomunikasi pedesaan, *Bul. Pos dan Telekomun.*, vol. 7, no. 1, pp. 132, 2009.

[4] T. Mariyati, Strategi pengembangan teknologi informasi dan komunikasi TIK serta pengaruhnya terhadap pertumbuhan ekonomi dan daya saing, *Bul. Pos dan Telekomun.*, vol. 7, no. 2, pp. 4994, 2009.

[5] R. De', N. Pandey, and A. Pal, Impact of digital surge during Covid-19 pandemic: A viewpoint on research and practice, *Int. J. Inf. Manage.*, vol. 55, no. June, p. 102171, Dec. 2020, doi: 10.1016/j.ijinfomgt.2020.102171.

[6] G. Nimrod, Changes in internet use when coping with stress: older adults during the COVID-19 pandemic, *Am. J. Geriatr. Psychiatry*, vol. 28, no. 10, pp. 10201024, 2020, doi: 10.1016/j.jagp.2020.07.010.

[7] C. A. Azlan, J. H. D. Wong, L. K. Tan, M. S. N. A. D. Huri, N. M. Ung, V. Pallath, C. P. L. Tan, C. H. Yeong, and K. H. Ng, Teaching and learning of postgraduate medical physics using Internet-based e-learning during the COVID-19 pandemic A case study from Malaysia, *Phys. Medica*, vol. 80, no. October, pp. 1016, 2020, doi: 10.1016/j.ejmp.2020.10.002.

[8] M. Naeem and W. Ozuem, The role of social media in internet banking transition during COVID-19 pandemic: Using multiple methods and sources in qualitative research, *J. Retail. Consum. Serv.*, vol. 60, no. January, p. 102483, 2021, doi: 10.1016/j.jretconser.2021.102483.

[9] Y. Sai, D. Fan, and M. Fan, Cooperative and efficient content caching and distribution mechanism in 5G network, *Comput. Commun.*, vol. 161, no. July, pp. 183190, 2020, doi: 10.1016/j.comcom.2020.07.030.

[10] C. Zhang, Design and application of fog computing and Internet of Things service platform for smart city, *Futur. Gener. Comput. Syst.*, vol. 112, pp. 630640, 2020, doi: 10.1016/j.future.2020.06.016.

[11] J. Thomas, Are ASEANs internet speeds world class?, The Asean Post, 2019. .

[12] A. Saverimoutou, B. Mathieu, and S. Vaton, Influence of internet protocols and CDN on web browsing, in *2019 10th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2019 - Proceedings and Workshop*, 2019, pp. 15, doi: 10.1109/NTMS.2019.8763827.

[13] A. Gasparyan, Most important metrics for your website performance, *Monitis*. 2019, Accessed: Oct. 26, 2019. [Online]. Available: https://www.monitis.com/blog/most-important-metrics-for-your-website-performance/.

[14] E. T. Loiacono, R. T. Watson, and D. L. Goodhue, WebQual: An instrument for consumer evaluation of web sites, *Int. J. Electron. Commer.*, vol. 11, no. 3, pp. 5187, 2007, doi: 10.2753/JEC1086-4415110302.

[15] D. Ayuba, A. Ismail, and M. Isa, Evaluation of page response time between partial and full rendering in a web-based catalog system, in *Procedia Technology*, vol. 11, pp. 807814, 2013, doi: 10.1016/j.protcy.2013.12.262.

[16] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, Edge computing: Vision and challenges, *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637646, 2016, doi: 10.1109/JIOT.2016.2579198.

[17] G. Karnitis and G. Arnicans, Migration of relational database to document-oriented database: Structure denormalization and data transformation, in *Proceedings - 7th International Conference on Computational Intelligence, Communication Systems and Networks, CICSyN 2015*, 2015, pp. 113118, doi: 10.1109/CICSyN.2015.30.

[18] Y. K. A. E. Alami, M. Bahaj, Supply of a key value database redis in-memory by data from a relational database, in *IEEE Mediterranean Electrotechnical Conference*, 2018, pp. 4651.

[19] D. Wang, X. An, X. Zhou, and X. L, Data cache optimization model based on cyclic genetic ant colony algorithm in edge computing environment, *Int. J. Distrib. Sens. Networks*, vol. 15, no. 8, 2019, doi: 10.1177/1550147719867864.

[20] D. Prerna, R. Tekchandani, and N. Kumar, Device-to-device content caching techniques in 5G: A taxonomy, solutions, and challenges, *Comput. Commun.*, vol. 153, no. November 2019, pp. 4884, 2020, doi: 10.1016/j.comcom.2020.01.057.

[21] T. M. Kroeger and D. D. E. Long, Exploring the bounds of web latency reduction from caching and prefetching, in *Symposium on Internet Technologies and Systems on USENIX, 1997*, [Online]. Available: https://dl.acm.org/citation.cfm?id=1267281.

[22] W. Teng, C. Chang, and M. Chen, Integrating web caching and web prefetching in client-side proxies, *IEEE Transactions On Parallel And Distributed Systems*, vol. 16, no. 5, 2005, pp. 444455.

[23] W. Ali, S. M. Shamsuddin, and A. S. Ismail, A survey of web caching and prefetching, *Int. J. Adv. Soft Comput. Appl.*, vol. 3, no. 1, pp. 127, 2011.

[24] M. Luthfi, M. Data, and W. Yahya, Perbandingan performa reverse proxy caching nginx dan varnish pada web server apache, *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 4, pp. 14571463, 2018.

[25] M. Kusuma, Evaluasi performa web server menggunakan varnish HTTP eeserve proxy dan redis database cache, in *Prosiding SENIATI*, 2016, no. Book-2, pp. 260264.

[26] I. A. Elgendy, W. Zhang, Y.-C. Tian, and K. Li, Resource allocation and computation offloading with data security for mobile edge computing, *Futur. Gener. Comput. Syst.*, vol. 100, pp. 531541, 2019, doi: 10.1016/j.future.2019.05.037.

[27] N. Pande, A. Somani, S. P. Samal, and V. Kakkirala, Enhanced web application and browsing performance through service-worker infusion framework, in *Proceedings - 2018 IEEE International Conference on Web Services, ICWS 2018 - Part of the 2018 IEEE World Congress on Services*, 2018, pp. 195202, doi: 10.1109/ICWS.2018.00032.

[28] K. Kim, S. Hong, S. Kim, and T. Kim, How to improve the performance of browsers with NVRAM, in *NVMSA 2017 - 6th IEEE Non-Volatile Memory Systems and Applications Symposium*, 2017, no. 10041608, doi: 10.1109/NVMSA.2017.8064470.

[29] X. S. Li, S. K. Yoon, J. G. Kim, and S. D. Kim, A self-learning pattern adaptive prefetching method for big data applications, *Sustain. Comput. Informatics Syst.*, vol. 20, pp. 6675, 2018, doi: 10.1016/j.suscom.2017.12.003.

[30] W. Chang, D. Goswami, S. Chakraborty, L. Ju, C. J. Xue, and S. Andalam, Memory-aware embedded control systems design, *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 36, no. 4, pp. 586599, 2017, doi: 10.1109/TCAD.2016.2613933.

[31] T. Wang, Y. Wang, X. Wang, and Y. Cao, A detailed review of D2D cache in helper selection, World Wide Web, no. October 2019, 2020, doi: 10.1007/s11280-019-00756-z.

[32] K. Dutta and D. Vandermeer, Caching to Reduce Mobile App Energy Consumption, *ACM Trans. Web*, vol. 12, no. 1, pp. 130, 2017, doi: https://doi.org/10.1145/3125778.

[33] G. Barish and K. Obraczka, World wide web caching: Trends and techniques, *IEEE Commun. Mag.*, vol. 38, no. 5, pp. 178185, 2000, doi: https://doi.org/10.1109/35.841844.

[34] T. Ma, Y. Hao, W. Shen, Y. Tian, and M. Al-Rodhaan, An improved web cache replacement algorithm based on weighting and cost, *IEEE Access*, vol. 6, pp. 2701027017, 2018, doi: 10.1109/ACCESS.2018.2829142.

[35] T. Koskela, J. Heikkonen, and K. Kaski, Web cache optimization with nonlinear model using object features, *Comput. Networks*, vol. 43, no. 6, pp. 805817, 2003, doi: 10.1016/S1389-1286(03)00334-7.

[36] J. Mertz and I. Nunes, Automation of application-level caching in a seamless way, *Softw. Pract. Exp.*, vol. 48, no. 6, pp. 12181237, Jun. 2018, doi: 10.1002/spe.2571.

[37] T. Chen, Obtaining the optimal cache document replacement policy for the caching system of an EC website, *Eur. J. Oper. Res.*, vol. 181, no. 2, pp. 828841, 2007, doi: 10.1016/j.ejor.2006.05.034.

[38] S. Podlipnig and L. B. Osz, A Survey of Web Cache Replacement Strategies, *ACM Comput. Surv.*, vol. 35, no. 4, pp. 374398, 2003, doi: https://doi.org/10.1145/954339.954341.

[39] C. Cunha, A. Bestavros, and M. Crovella, Characteristics of WWW client-based traces, *Cummington St. Boston*, MA, 1995. doi: 10.5555/859844.

[40] G. Hasslinger, K. Ntougias, F. Hasslinger, and O. Hohlfeld, Performance evaluation for new web caching strategies combining LRU with score based object selection, *Comput. Networks*, vol. 125, pp. 172186, 2017, doi: 10.1016/j.comnet.2017.04.044.

[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.

[42] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, Evaluating content management techniques for Web proxy caches, *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 27, no. 4, pp. 311, Mar. 2000, doi: 10.1145/346000.346003.

[43] P. Cao and S. Irani, Cost-Aware WWW Proxy Caching Algorithms, in *1st USENIX Symp. Internet Technol. Syst. USITS 1997*, no. December, 1997.

[44] L. Cherkasova, Improving WWW proxies performance with greedy-dual-size-frequency caching policy, HP Lab. Tech. Rep., no. 9869, 1998, doi: 10.1.1.94.5958.

[45] NLANR, The National Laboratory for Applied Network Research (NLANR), 2001. http://www.nlanr.net/ (accessed Jul. 03, 2020).

[46] H. Ibrahim, W. Yasin, N. I. Udzir, and B. Process, Intelligent cooperative web caching policies for media objects based on J48 decision tree and Nave Bayes supervised machine learning algorithms in structured peer-to-peer systems, *J. Inf. Commun. Technol.*, vol. 15, no. 2, pp. 85116, 2016.

[47] X. Li, X. Wang, Z. Sheng, H. Zhou, and V. C. M. Leung, Resource allocation for cache-enabled cloud-based small cell networks, *Comput. Commun.*, vol. 127, no. April, pp. 2029, Sep. 2018, doi: 10.1016/j.comcom.2018.05.007.

[48] T. Ma, J. Qu, W. Shen, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan, Weighted greedy dual size frequency based caching replacement algorithm, *IEEE Access*, vol. 6, pp. 72147223, 2018, doi: 10.1109/ACCESS.2018.2790381.

[49] M. I. Zulfa, R. Hartanto, A. E. Permanasari, and W. Ali, Improving cached data offloading optimization based on enhanced hybrid ant colony genetic algorithm, *IEEE Access*, vol. 10, no. August, pp. 8455884568, 2022, doi: 10.1109/ACCESS.2022.3197205.

[50] W. Ali, S. Sulaiman, and N. Ahmad, Performance improvement of least-recently-used policy in web proxy cache replacement using supervised machine learning, *Int. J. Adv. Soft Comput. its Appl.*, vol. 6, no. 1, pp. 138, 2014.

[51] V. Holmqvist and J. Nilsfors, Cachematic automatic invalidation in application-level caching systems, in *International Conference on Performance Engineering*, 2019, pp. 167178.

[52] S. Chen, X. Tang, H. Wang, H. Zhao, and M. Guo, Towards scalable and reliable in-memory storage system: A case study with redis, in *2016 IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 16601667, doi: 10.1109/TrustCom.2016.0255.

[53] M. Li, H. Zhang, Y. Wu, and C. Zhao, MemSC: A scan-resistant and compact cache replacement framework for memory-based key-value cache systems, *J. Comput. Sci. Technol.*, vol. 32, no. 1, pp. 5567, Jan. 2017, doi: 10.1007/s11390-017-1705-3.