e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:09/September-2022 Impact Factor- 6.752 www.irjmets.com

IMPROVING AND DELIVERING RELEVANT WEB CONTENT USING WEB CACHING

Parul Kumari[*1]

[*1]Student, Dept. Of Information Technology, B.K. Birla College Of Arts, Science And Commerce (Autonomous), Kalyan, Maharashtra, India.

https://www.doi.org/10.56726/IRJMETS29654

ABSTRACT

Web caching is as important as it has ever been. Web caching technology has been widely used to enhance the performance of the web infrastructure and reduce the user-perceived network latencies. Proxy caching is a Web caching technique that aims to serve users. Web requests from one or a network of proxies located between the client and the servers hosting the original copies of the requested objects. Web caching was developed for two primary reasons: to cut back the load on web servers and to enhance the net user's experience while browsing the web. Prefetching techniques are used to boost processor's execution performance and faster fetch operations. This paper presents different approaches for improving the techniques to deliver relevant web content and an analysed study of prefetching techniques and various databases employed by caching servers and different browsers and discusses the techniques to reinforce web caching, discusses caching databases also the intended impacts of web caching and related performance measures, and focuses on server caching approaches that use a single cache to serve multiple users which is the Content Delivery Network (CDN) caching, object caching, and opcode caching and analysing the behaviour of various browsers associated with caching.

Keywords: Web Caching, Proxy Server, Reverse Caching, Prefetching, Redis And Memcached, Browser Behaviour, Etc.

I. INTRODUCTION

Caching has been considered to boost the delivery of content in wireless networks. Web caching is the activity of storing data for reuse, such as a replica of a web page served by an online web server. It is cached or stored the very first time a user visits the page and the next time a user requests the identical page, a cache will serve the replica, which helps keep the origin server from getting overloaded. The Internet bandwidth capacity expansion, on the alternative hand, is lagging making the web a serious performance bottleneck. The gap between the web infrastructure capacity and demand will still exist, if not expand, as information search and business transactions are being increasingly conducted over the web.

Web caching solutions and techniques improve page delivery speed significantly and reduce the work needed to be done by the backend server. Caching servers can be set to refresh at specific intervals or in response to certain events to make sure that the freshest or the foremost relevant content is cached (useful for rapidly changing information, for instance, breaking news or change in price). Web caching is done by a Proxy Server which is an intermediate entity between the original server and the client. Effective caching benefits both content consumers and content providers.

Some of the advantages that caching bring to content delivery are: Decreased costs in the network, improved responsiveness, increased performance on the same hardware, and availability of content during network interruptions. To make effective use of caching, an informative decision has to be made on which documents are to be evicted from the cache in case of cache saturation. This is particularly important during a wireless network, where the size of the client cache at the mobile terminal (MT) is small. Many medium-to-large enterprises are employing various caching products and services to boost network performance and reduce networking connection costs. Many end-user programs, including Web browsers, also maintain their local caches to scale back user-perceived network latencies.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:09/September-2022 Impact Factor- 6.752 www.irjmets.com

II. BACKGROUND STUDIES

Measuring Performance and Aids of Caching

The potential benefits of Web caching are multifold. From the end user's perspective, caching can significantly reduce the user perceived network latency and improve their web experience. From the viewpoint of the Internet infrastructure, caching can scale back the proportion of the online web traffic. As a result, the overall performance of the Internet can be enhanced and network crowding is minimized

Web caching is the most suitable and most sustainable solution to reduce internet traffic and bandwidth consumption. The cache exists to bridge the speed gap so it's important to measure its performance in designing and choosing several parameters like cache size, replacement policy, etc.

We consider five aspects of web caching benefits which are: hit ratio, byte hit ratio, latency reduction, hop reduction, and weighted-hop reduction. For the byte hit ratio, we can say the number of bytes that hit the proxy cache as the percentage of the total number of bytes requested which measures the amount of bandwidth the cache has saved. The Hit ratio is the probability of getting hits out of some number of memory references made by the CPU. Cache Hit eliminates the needs to contact the originated server. Considering the limited storage space of surrogate servers, the proper replacement of contents decreases the miss rate of content requests as a crucial performance metric in CDNs.

There are many techniques to improve hit rates such as using the small and simple cache, trace caches, and pipelined cache access and retaining time loss in the address translation. Caching reduces bandwidth consumption; hence, it reduces network traffic and decreases network congestion Caching strives to scale back the latency of the user related to obtaining the documents. Latency can be reduced as the cache is generally nearer to the client than the source of the content. Caching tries to reduce the network traffic from the web servers. Network load can be reduced for the reason that the pages that are served from the cache have to traverse less of the network than when they are served by the source of the content. This can scale back the number of requests on the content provider. It may also lower the transit costs for access providers and is being consummate by temporarily storing static assets (such as images, HTML, CSS, and JavaScript) from the webpage so they can be easily accessed later. Correctly using a cache can scale up the website's performance, and a faster website can lead to higher conversion rates.

Reverse Proxy Caching

A reverse proxy can reduce the load on its original servers by caching both static and dynamic content, referred to as web acceleration. A conventional forward proxy server allows multiple clients to route traffic to an external network. For example, a business may have a proxy that routes and filters employee traffic to the public Internet. A reverse proxy, on the alternative hand, routes traffic on behalf of multiple servers. A server cache is a kind of cache that's associated with site caching, except rather than temporarily saving content on the client side, it is stored on a site's server. Server caching is additionally fully handled and administered on the server with no involvement of the end-user or a browser. This includes Content Delivery Network (CDN), Object Caching, and Opcode Caching.

For caching, a Content Delivery Network (CDN) narrows the load on an application origin and enhances the experience of the requestor by delivering a local copy of the content from a near cache edge, or Point of Presence. CDNs are geographically distributed networks of proxy servers and they aim to serve content to users more quickly. Object caching involves storing database query results so that when the following time a result's needed, it can be served from the cache without having to repeatedly query the database. OPcache is a kind of caching system that saves precompiled script bytecode during a cache, so every time a user visits an online web page, it loads faster.

A reverse proxy is a server that sits ahead of web servers and forwards client (e.g. web browser) requests to those web servers. They're generally implemented to assist the increase in security, performance, and reliability. Caching servers are accountable for the storage and delivery of cached files. Their main function is to accelerate website load times and cut back bandwidth consumption. Each CDN caching server generally holds multiple storage drives and high amounts of RAM resources. In proxy caching, the cache server receives the request for an item/object from a client. If the item is present in its cache, it responds with the item. Else, it

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:09/September-2022 Impact Factor- 6.752 www.irjmets.com

requests the source of the item and ensures the client has the requested item. The proxy either serves these requests using previously cached responses or obtains the required documents from the original Web servers on behalf of the clients[5]. The benefits of Proxy caching are that it reduces latency and network traffic which makes the experience of the web better and higher availability of the websites. However, the disadvantage is that cache could be a single point of failure.

Conceptually, proxy caches can be viewed as middleware connecting to end-user web programs and web servers through protocols. Such caches act as servers to client programs and as clients to Web servers. As a proxy server has its own IP address, it acts as a go-between for a computer and the Internet. The computer knows this address, and when one sends a request on the internet, it's routed to the proxy, which then gets the response from the web server and forwards the information from the page to the computer's browser.

In-Memory Databases: Redis and Memcached

An in-memory cache acts as a data storage layer that sits between applications and databases to deliver responses with high speeds by storing data from earlier requests or copied directly from databases Compared with the conventional disk database, the in-memory database has faster data-storage speed, higher throughput, and stronger concurrent access capability, which meets the fast response requirements of many applications.

In-memory data stores don't require a visit to disk, reducing engine latency to microseconds. Due to this, in-memory data stores can support an order of magnitude more operations and faster response times. This results in a blazing-fast performance with average read and write operations taking less than a millisecond and support for millions of operations per second. The foremost popular in-memory caches are redis, memchaed, and ehcache.

Redis delivers sub-millisecond response times, enabling ample requests per second for real-time applications in industries like gaming, ad-tech, financial services, healthcare, and IoT. Due to its fast performance, Redis may be a popular choice for caching, session management, gaming, leaderboards, real-time analytics, geospatial, ride-hailing, chat/messaging, media streaming, and pub/sub-apps. Memcached has distributed memory caching system accustomed to speeding up the dynamic database-oriented websites by caching data and objects in RAM to chop back the number of times the external is read. It's a free and open-source database that permits one to effectively scale higher loads. This is popular for database query results caching, session caching, web content caching, API caching, and caching of objects like images, files, and metadata.

Many known companies use redis as in-memory storage (example: AWS, Twitter, Snapchat, StackOverflow)

III. EXPERIMENT OVERVIEW AND ANALYSIS

Prefetching Techniques

The two main techniques for tolerating read latency as well as write latency are prefetching and multithreading. The key to tolerating read latency is to split apart the request for data and the use of that data while finding enough parallelism to keep the processor busy in between. The distinction between prefetching and multithreading is that prefetching finds parallelism within the one single thread of execution, while multithreading exploits parallelism across multiple threads. Prefetching algorithms are based on clustering of users, probability of access and popularity of web objects. With the help of prefetching, latency is reduced, and bandwidth under-consumption is avoided

To hide the latency within one single thread, the request for the data (i.e. the prefetch request) must be moved back sufficiently far beforehand of the utilization of the data within the execution stream. There are different prefetching techniques discussed in this paper such as: dns-prefetch, prefetch, pre-connect, and pre-render.

a) DNS-Prefetch

A dns-prefetch tells the browser that we'd need a few resources from a specific URL, and therefore the browser can start the DNS resolution as quickly as possible.

<link rel="dns-prefetch" href="https://example.com" />

Once the browser completes parsing the document, it starts with the DNS resolution for the searched item (In this case example.com). Thus the further requests to the searched item for resources become slightly faster. It can be used when one knows that they have to connect to that domain soon, and want to speed up the initial

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:09/September-2022 Impact Factor- 6.752 www.irjmets.com

connection. It is a way to fix cache validation for the URL of the requested page. The resource is loaded later. It has better browser support than pre-connect.

b) Prefetch

Pre-connect is analogous to a DNS-prefetch, but it also carries out the TCP handshake and TLS negotiations alternatively. This minimizes the user's time by eliminating round-trip latency.

<link rel="prefetch" href="input.html">

This asks the browser to download and cache a resource. The download of the resource (script.js or stylesheet) happens with low priority. This saves time and avoids the repeated query request process each time the webpage is requested. It's necessary to specify the 'as' attribute because it helps the browser to prioritize and schedule the download properly.

In prefetch, the browser downloads the mentioned resources when it has free time or is idle. It gives a low priority to these resources. This prefetch is used when the resource has to be used later or in the future. For example, any navigation page like about us or contact us. It is used when the resource is not necessary to load at an initial phase which minimizes the load of fetching the data from the server and hence improves the performance of the web.

C) Pre-render

Pre-render is the epitome of all prefetching techniques and must be used with caution. Pre-rendering tells the browser to load all the assets from a particular web page.

<link rel="prerender" href="https://example.com/about.html" />

In effect, the page at https://example.com/about.html is invisibly loaded and kept by the browser. When the navigation happens from the current page to this page, the browser can readily load the pre-rendered page, thus enhancing the user experience. It is useful when one is sure that a user will visit a specific page next, then one has to render it faster. It is used to create static documents before we serve the user. We are sending the data and processing it to static documents when one can serve the information. It is a very good way of creating a very performant website to survive on basic static assets but has enough information. It is useful for a formally large website but mostly uses static information.

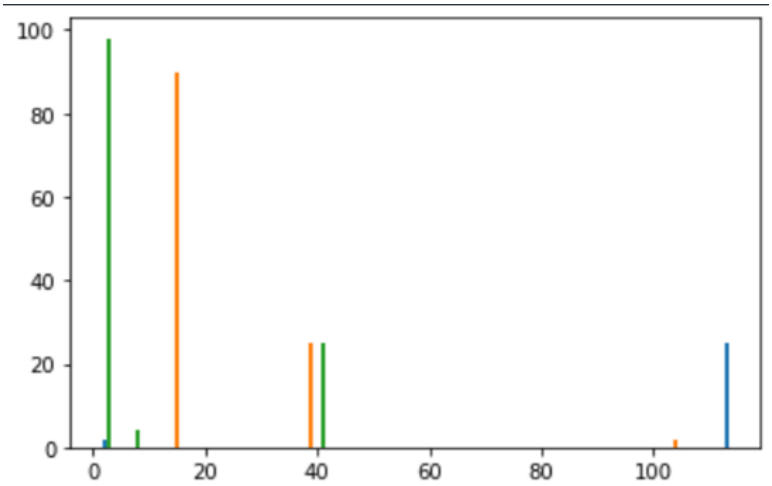

Figure 1: Performance vs Time Graph of different prefetching techniques.

The most widely used prefetching technique is link prefetching or simply prefetch as the prefetch link relation type is used to identify a resource that might be required by the next navigation, and that the user agent should fetch, such that the user agent can deliver a faster response once the resource is requested in the future. When prefetching is used in channel search, the contents of a channel can be received before users actually requests that channel, thereby reducing the delay time that occurs when switching to the corresponding channel. One can any prefetching technique based on the requirement. This research has experimented the above prefetching techniques. As a result, the mentioned prefetching techniques are used on the basis of one's need as this prefetching helps in faster delivery of web content.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:09/September-2022 Impact Factor- 6.752 www.irjmets.com

Comparative Study of Redis and Memcached

By experimenting and analyzing both Redis and Memcached we can say that both Memcached and Redis offer sub-millisecond response times by keeping data in memory. Also, both in-memory databases allow the distribution of data across multiple nodes. Similarly, both support all major programming languages including Java, Python, JavaScript, C, and Ruby. Furthermore, there are a few Java clients available for both in-memory databases. For example, Xmemcached and Memcached-java-client are available for Memcache, while Jedis, Lettuce, and Redisson are available for redis. Memcached allows clearing or cleaning the cache using the 'flush_all' command whereas Redis allows us to do everything from a cache by using commands like 'FLUSHDB' and 'FLUSHALL'.Memcached is a transient memory.

Table 1. Comparison of In-Memory databases: Redis and Memcached.

Redis	Memcached
Redis comes with an avid command-line interface (redis-cli) allowing us to execute commands.	Memcached allows one to run commands by connecting to the server using telnet.
Redis provides us functionality to multiply clusters by duplicating the primary storage for better functionality and high obtainability	Memcached supports duplication with third-party forks like rep-cached.
Redis provides the out-of-the-box feature for transactions to execute commands. (One can start the transaction using the 'MULTI' command, 'EXEC' command for the execution of the subsequent commands and provides the 'WATCH' command for the conditional execution of the transaction.)	Memcached doesn't support transactions, whilst its operations are atomic.
Redis uses a single core and displays higher performance than Memcached in storing small datasets	Memcached implements a multi-threaded architecture by employing multiple cores.
It can scale horizontally via clustering, which is relatively more complex to line up and operate. Also, one can use Jedis or Lettuce to enable a Redis cluster employing a Java application	Memcached's multi-threaded architecture is highly extensible, accomplished by employing multiple computational resources.

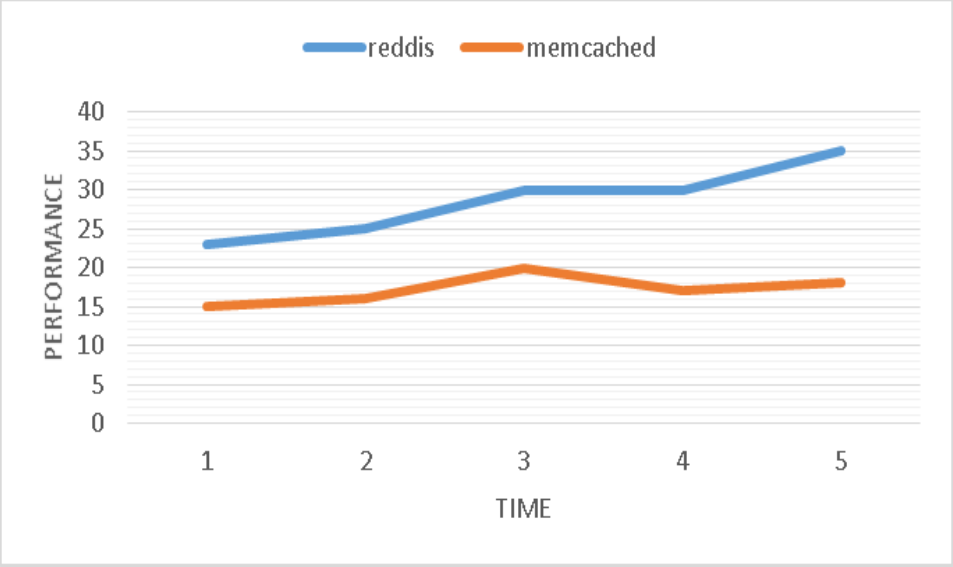

Figure 2: Redis and Memcached Performance Graph

One can say redis is much better in terms of efficiency as it's faster and has much more workloads that redis can do which memcached can't. The very tiny performance differences (almost in microseconds) seem minor in

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:09/September-2022 Impact Factor- 6.752 www.irjmets.com

the face of the enormous gulf in functionality and therefore the proven fact that both tools are so fast and efficient that they may very well be the last piece of your infrastructure one ever needs to worry about scaling.

Redis is helpful when the project is multilingual because redis has shared data structures. Redis needs to apply and release memory frequently to the system, which leads to increasing memory occupied by Redis and increasing fragmentation rate. One can put pre-defined data structures in one language and retrieve them in another language.

Caching Behavior of Web browsers

When a user visits an online web page, the contents of that page are stored within the browser's cache so it doesn't need to be re-requested and re-downloaded. Mostly, browsers use local caches to improve the latency of requests that can hit in the cache and to reduce the network bandwidth consumed. Effectively and efficiently using the browser cache can boost end-user response times and reduce bandwidth utilization. The cache-ability of an object on the browser is decided by:

- The response headers returned from the original web server. If the headers display that the content must not be cached then it won't be.
- A validator like an ETag or Last-Modified header must be present within the response.

If an item is considered cacheable, the browser will retrieve the item from the cache on repeat visits if it's considered "fresh" (Freshness determined by a legitimate expiration time that's still within the fresh period).

a) Chrome

When a user revisits the identical site or visits another website where the identical files are used, Chrome will load them from its internal cache, rather than re-downloading each file again. Chrome 86 which was released in 2020, Google has changed its mechanism which is known as "cache partitioning", this feature works by changing how resources are saved within the HTTP cache supported by two additional factors. Due to this, a resource's storage key will contain three items, instead of one: the top-level site domain (http://a.example), the resource's current frame (http://c.example), the resource's URL (https://b.example/cat.png). Chrome has blocked all the past attacks against its cache mechanism, as most website components will only have access to their own resources and won't be able to check resources that they haven't created themselves.

b) Firefox

Mozilla Firefox uses in-memory caching for entire sites, including their JavaScript states, for one single browser session. Going backward and forward between visited pages requires no page loading then the JavaScript states are preserved. This feature, remarked by some as bfcache (for "Back-Forward Cache"), makes page navigation in no time. The caching state is perpetuated until the user closes the browser. However, Mozilla has also announced similar plans to implement Chrome's cache partitioning mechanism.

c) Safari

Safari caches content from third-party origins separately for each document origin, so for example if two sites say a.com and b.com both use a common library, third-party.com/script.js, then script.js will be cached separately for both sites and if someone has an 'empty' cache and visits the first site and then the other, script.js will be downloaded twice. In theory, using common libraries, fonts, etc. from a public CDN provides several benefits: reduced hosting costs for sites on a tight budget, improved performance as the resource is hosted on a Content Delivery Network (CDN), etc.

To see the differences between the application with and without acceleration, a new browser session must be initiated; the other three ways of loading a page on repeat visits will show no differences with or without acceleration. Modern web services use in-memory caching extensively to increase throughput and reduce latency.

IV. CONCLUSION

This paper concludes with major technical approaches related to the design and operation of proxy cache networks and server caching techniques. It also discusses the databases used for caching analyzing different techniques we can apply over it. In conclusion to that redis is better and more efficient than memcahed as the difference of microseconds plays a huge role in delivery data. This paper represents a detailed and comparative

study of caching databases redis and memcached and analyzed the behavior of different web browsers for effective caching.

This also includes the study of prefetching techniques and reverse caching proxy for proper content delivery. Prefetching techniques used in different webpages helps one to save response time and redundancy of query process. Depending upon the need these techniques are used. Also, analysis of the performance of our web caching system by logging the cache hit and total time to process one hundred requests when adding and removing cache servers. The detailed study of replacement algorithms and the introduction of new techniques for faster content delivery using different sever aching techniques can also be discussed in the future.

V. REFERENCES

[1] Balamash, Abdullah, and Marwan Krunz. "An Overview of Web Caching Replacement Algorithms." IEEE Communications Surveys & Tutorials 6, no. 2 (2004): 44–56. https://doi.org/10.1109/COMST.2004.5342239

[2] Zeng, D., F.-Y. Wang, and M. Liu. "Efficient Web Content Delivery Using Proxy Caching Techniques." IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) 34, no. 3 (August 2004): 270–80. https://doi.org/10.1109/TSMCC.2004.829261.

[3] Nanda, Pranay, Shamsher Singh, and G.L. Saini. "A Review of Web Caching Techniques and Caching Algorithms for Effective and Improved Caching." International Journal of Computer Applications 128, no. 10 (October 15, 2015): 41–45. https://doi.org/10.5120/ijca2015906656.

[4] Barish, G., and K. Obraczka. "World Wide Web Caching: Trends and Techniques." IEEE Communications Magazine 38, no. 5 (May 2000): 178–84. https://doi.org/10.1109/35.841844.

[5] Ghasemi, Abdorasoul, and Amirhosein Ahmadi. "Cache Management in Content Delivery Networks Using the Metadata of Online Social Networks." Computer Communications 189 (May 2022): 11–17. https://doi.org/10.1016/j.comcom.2022.02.021.

[6] Nalajala, Anusha, T. Ragunathan, Rathnamma Gopisetty, and Vignesh Garrapally. "Rank-Based Prefetching and Multi-Level Caching Algorithms to Improve the Efficiency of Read Operations in Distributed File Systems." translated by Satish Narayana Srirama, Jerry Chun-Wei Lin, Raj Bhatnagar, Sonali Agarwal, and P. Krishna Reddy, 227–43. Big Data Analytics. Cham: Springer International Publishing, 2021.

[7] Liu, Qian. "A High Performance Memory Key-Value Database Based on Redis." Journal of Computers 14, no. 3 (2019): 170–83. https://doi.org/10.17706/jcp.14.3.170-183.

[8] Li, Dingding, Mianxiong Dong, Yanting Yuan, Jiaxin Chen, Kaoru Ota, and Yong Tang. "SEER-MCache: A Prefetchable Memory Object Caching System for IoT Real-Time Data Processing." IEEE Internet of Things Journal 5, no. 5 (October 2018): 3648–60. https://doi.org/10.1109/JIOT.2018.2868334.

[9] Bahn, Hyokyung. "Effect of Caching and Prefetching Techniques to Accelerate Channel Search Latency in IPTVs." International Journal of Internet, Broadcasting and Communication 14, no. 2 (May 31, 2022): 17–22. https://doi.org/10.7236/IJIBC.2022.14.2.17.

[10] Singh, T S Bhagavath, and S Chitra. "Prefetching of Web Objects For Effective Retrieval Process Through Data Mining Techniques." Preprint. In Review, April 30, 2021. https://doi.org/10.21203/rs.3.rs-266666/v1.

[11] Hosseini, S. Mohammad, Amir Hossein Jahangir, and Sina Daraby. "Session-Persistent Load Balancing for Clustered Web Servers without Acting as a Reverse-Proxy." In 2021 17th International Conference on Network and Service Management (CNSM), 360–64. Izmir, Turkey: IEEE, 2021. https://doi.org/10.23919/CNSM52442.2021.9615592.

[12] Yang, Juncheng, Yao Yue, and K. V. Rashmi. "A Large-Scale Analysis of Hundreds of In-Memory Key-Value Cache Clusters at Twitter." ACM Transactions on Storage 17, no. 3 (August 31, 2021): 1–35. https://doi.org/10.1145/3468521.

[13] Singh, T S Bhagavath, and S Chitra. "Prefetching of Web Objects For Effective Retrieval Process Through Data Mining Techniques." Preprint. In Review, April 30, 2021. https://doi.org/10.21203/rs.3.rs-266666/v1.

[14] Tiago, João, David Dias, and Luís Veiga. "Adaptive Edge Content Delivery Networks for Web-Scale File Systems." arXiv, July 13, 2022. http://arxiv.org/abs/2207.06341.

[15] M. Miyamoto. R. Kawashima and H. Matsuo, "Transparent Relational Database Caching Based on Strong Engines Using In-Memory Database", 2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW), 2021, pp. 444-448, doi: 10.1109/CANDARW53999.2021.00082.

[16] Buyuktanir, Tolga, and Mehmet S. Aktas. "Mobile Prefetching and Web Prefetching: A Systematic Literature Review." translated by Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Ana Maria A. C. Rocha, and Chiara Garau, 75–89. Computational Science and Its Applications – ICCSA 2022 Workshops. Cham: Springer International Publishing, 2022.

[17] Shivakumar, Shailesh Kumar. "General Web Performance Optimization Methods." Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed Up Digital Platforms. Berkeley, CA: Apress, 2020. https://doi.org/10.1007/978-1-4842-6528-4_2.

[18] D. Li, M. Dong, Y. Yuan, J. Chen, K. Ota and Y. Tang, "SEER-MCache: A Prefetchable Memory Object Caching System for IoT Real-Time Data Processing," in IEEE Internet of Things Journal, vol. 5, no. 5, pp. 3648-3660, Oct. 2018, doi: 10.1109/JIOT.2018.2868334.

[19] Ziv Scully and Adam Chlipala. 2017. "A program optimization for automatic database result caching". In Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL '17). Association for Computing Machinery, New York, NY, USA, 271–284. https://doi.org/10.1145/3009837.3009891

[20] Zulfa, M.I., Hartanto, R. and Permanasari, A.E. (2020), "Caching strategy for Web application – a systematic literature review", International Journal of Web Information Systems, Vol. 16 No. 5, pp. 545-569. https://doi.org/10.1108/IJWIS-06-2020-0032