INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 08 ISSUE: 05 | MAY - 2024 SJIF RATING: 8.448 ISSN: 2582-3930

Decoding Efficiency: A Technical Exploration of Apache, Nginx and Varnish Cache Server through Comprehensive Performance Metrics

Sakshi S. Sawant (Student)
D Y Patil Institute of Master Of Computer Applications and Management Akurdi,Pune
sakshisawantj@gmail.com

Manjiri M. Chavan, Asst.Professor
D Y Patil Institute of Master Of Computer Applications and Management Akurdi,Pune
Manjiri.chavan@dypimca.ac.in

Abstract- Web servers play a crucial role in delivering web content efficiently to users. When a web server receives a request, it processes the request for the requested resources. One method to optimize this process is through the use of cache servers. Cache servers store frequently accessed data in memory, reducing the need to retrieve data from the original source each time a request is made. By leveraging cache servers, web servers can significantly enhance their performance by reducing response times. When a cache server successfully serves a request with a cache hit, it eliminates the need for the web server to process the request and retrieve the data, thereby speeding up the response time. This efficiency is crucial for improving user experience, as faster response times lead to quicker loading of web pages and reduced latency. Through this technical exploration of Apache, Nginx, and Varnish cache servers, we aim to analyze and compare their performance metrics to determine the most effective solution for reducing response times and optimizing web server efficiency. Understanding how cache servers impact performance in terms of cache hit ratio, cache miss ratio, client connections, CPU usage, memory usage, error rates, requests per second, and bandwidth will provide valuable insights into selecting the best cache server solution for improved web server performance.

Keywords— Cache Server, Cache, Response time, Apache, Nginx, Varnish.

I. INTRODUCTION

A web server is a software application that processes requests from clients over the Hypertext Transfer Protocol (HTTP) to deliver web content. When a client, such as a web browser, sends a request for a specific webpage or resource, the web server retrieves and responds with the requested data. To optimize this process, cache servers act as intermediaries between the client and the web server.

Cache servers store copies of frequently accessed data in memory. When a client makes a request, the cache server checks if the data is available in its cache. If the data is found (cache hit), the cache server can directly respond to the client, reducing the need for the web server to retrieve the data. In cases where the requested data is not in the cache (cache miss), the cache server fetches the data from the web server and stores it for future requests.

The most important factors affecting cache performance include accessing frequency, last access time, cache block size, and cache object size. Accessing frequency refers to how often data is accessed from the cache, impacting the

efficiency of the cache system. The last access time is crucial as it determines how quickly data can be retrieved from the cache. Cache block size plays a significant role in performance, with larger blocks benefiting from spatial locality but potentially increasing miss penalties. Additionally, the size of cache objects influences performance, affecting how efficiently data is stored and retrieved within the cache system.

Efficiency in delivering web content is influenced by various factors affecting response time. These factors include the performance of cache servers, the speed of data retrieval, network latency, and server processing time. Additionally, the use of HTTP response codes plays a critical role in communicating the status of requests between clients and servers. By understanding and optimizing these factors, web servers can deliver content swiftly and improve user experience.

In today's digital landscape, efficient content delivery is paramount for enhancing user experience. Prominent solutions like Apache, Nginx, and Varnish Cache Server are utilized to reduce response time and improve user experience. These cache servers employ diverse caching strategies to store and deliver frequently accessed content, thereby boosting the speed and performance of web applications. Each server provides distinct features and functionalities tailored to specific requirements, making them indispensable components in the realm of web caching and content delivery.

II. RELATED WORK

This study proposes the innovative concept of HTTP timed redirection, which enables the consolidation of GET requests within a hybrid web server. This approach allows for the Master server to enter energy-saving sleep intervals, thereby reducing its energy consumption. The experimental findings demonstrate substantial energy savings for the Master server, particularly when incorporating delays for specific requests. This method is well-suited for applications that can tolerate delays. Overall, the research offers a promising strategy for minimizing energy consumption in hybrid web servers through the development of novel network protocols.[1]

The paper advocates for the integration of browser and proxy caches to enhance cache management architecture, aiming to alleviate internet traffic. It explores the benefits of hierarchical and distributed caching, proposing a hybrid model for superior performance. With the exponential growth of web traffic impacting system speed and data generation rates, monitoring becomes imperative. The study underscores the importance of efficient cache protocols, validation techniques, and security measures to elevate caching systems. Overall, the research targets the escalating challenges of web traffic by optimizing cache management through innovative strategies and hybrid cache architectures.[2]

The proposed distributed cache management architecture seeks to diminish internet traffic and enrich user experience by merging browser and proxy caches. By distributing proxy server loads and enhancing cache availability, the architecture can refine network performance and diminish latency for users. The system's design entails proxy servers locally verifying cache availability, establishing peer-to-peer connections, and validating cache freshness using E-Tags or modification dates.[2]

The research conducted a performance comparison of caching strategies on a Wordpress Multisite, examining two scenarios: utilizing an http accelerator (varnish) full cache without database cache and employing an http accelerator

(varnish) full cache with database cache (redis). The findings indicate that segregating object cache types can notably diminish MySQL disk read operations, reduce CPU utilization by 25%, and slash network traffic by up to 94%, all while maintaining a flawless 0% error rate for clients. However, the study did not investigate the impact of variables like network latency, server location, or user demographics on caching strategy performance, which could influence the overall efficacy of the caching approach.[3]

The research paper introduces WSCRP, a novel web cache replacement algorithm that prioritizes weight and cost factors to optimize hit rate in constrained cache memory environments. By incorporating frequency, time, and cost metrics, WSCRP outperforms traditional methods like LRU, LFU, and WRP, leading to enhanced performance indicators such as hit rate and byte rate across diverse datasets. The study underscores the significance of effective cache replacement strategies in web caching. Experimental findings validate the superiority of the proposed algorithm over conventional techniques, demonstrating improved hit rate and byte rate for a range of datasets.[4]

The paper introduces an active learning-driven caching framework tailored for edge caching to mitigate delivery latency and network congestion by proximity to end-users. It presents an active learning approach for content popularity prediction and suggests two caching policies to optimize queries and ensure minimum cache hit ratios. This model-free framework is adaptable to static and dynamic content popularity models. Through Monte Carlo simulations, the performance of the proposed caching policies surpasses existing methods, showcasing their effectiveness. The research highlights key contributions such as the active learning method for content popularity prediction and an adaptive caching framework designed to enhance caching efficiency.[5]

The research paper formulates an integer linear program to address a content caching problem that considers content downloading costs and deadline constraints. Three algorithms are developed for problem-solving: repeated column generation, popularity-based caching, and random-based caching, with the repeated column generation algorithm outperforming the others. Simulation results demonstrate that the repeated column generation algorithm provides nearly optimal solutions within a 1.6% gap of the global optimum. The study suggests that significant cost reductions (up to one-third) can be achieved when content requests have longer deadlines. Different caching algorithms are suitable for various scenarios based on cache capacity and content popularity, offering insights for optimizing content caching strategies in time-slotted systems.[6]

The research compares Apache and Nginx web server performance on Docker, with Apache demonstrating superiority in requests per second. Both servers exhibit comparable performance in availability, failed transactions, response time, and transaction rate within the Docker environment. The study evaluates containerization, specifically Docker, for Apache and Nginx web servers' performance, highlighting similar results in various metrics. Apache outperforms Nginx in requests per second with default configurations on Docker. Docker is recommended for deploying Apache and Nginx web servers due to their high performance and availability.[7]

© W3Techs.com	usage	change since 1 October 2021
1. Nginx	33.4%	-0.6%
2. Apache	31.4%	+0.4%
3. Cloudflare Server	20.8%	+0.7%
4. LiteSpeed	11.6%	+0.3%
5. Microsoft-IIS	6.4%	-0.1%

percentages of sites

Fig. 1 Most Popular Web Server[7]

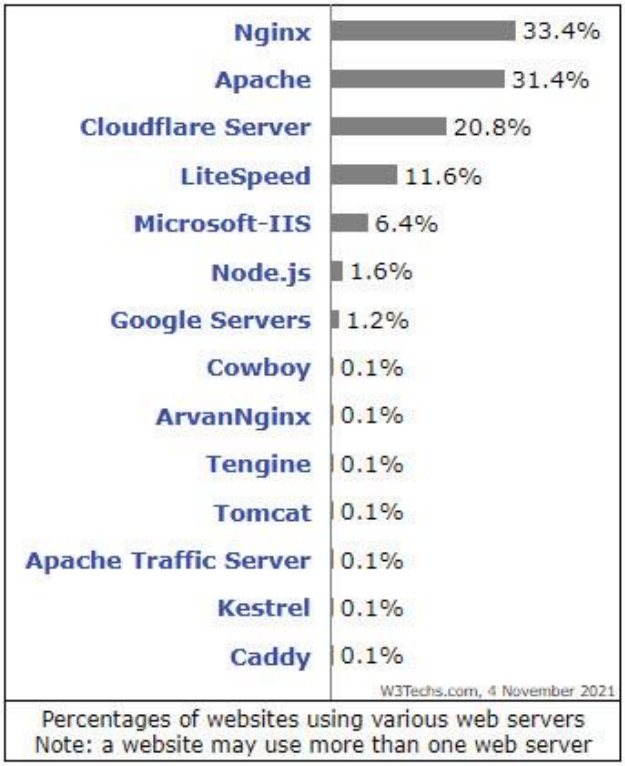


Fig. 2 Usage of Web Server [7]

III. RESEARCH METHODOLOGY

A cache is a high-speed memory that stores frequently accessed data to minimize the time required to retrieve information from the main memory or other slower storage devices. It serves as an intermediary between the CPU and main memory, enabling expedited access to data and instructions. Caches can be hardware or software components, finding application in diverse domains such as web browsers, servers, and databases. Their operation involves storing copies of data in a designated location, which may be local or on a server, facilitating faster retrieval and alleviating the burden on the main memory or storage devices. Judicious utilization of caches can substantially enhance system performance by diminishing latency and augmenting throughput.

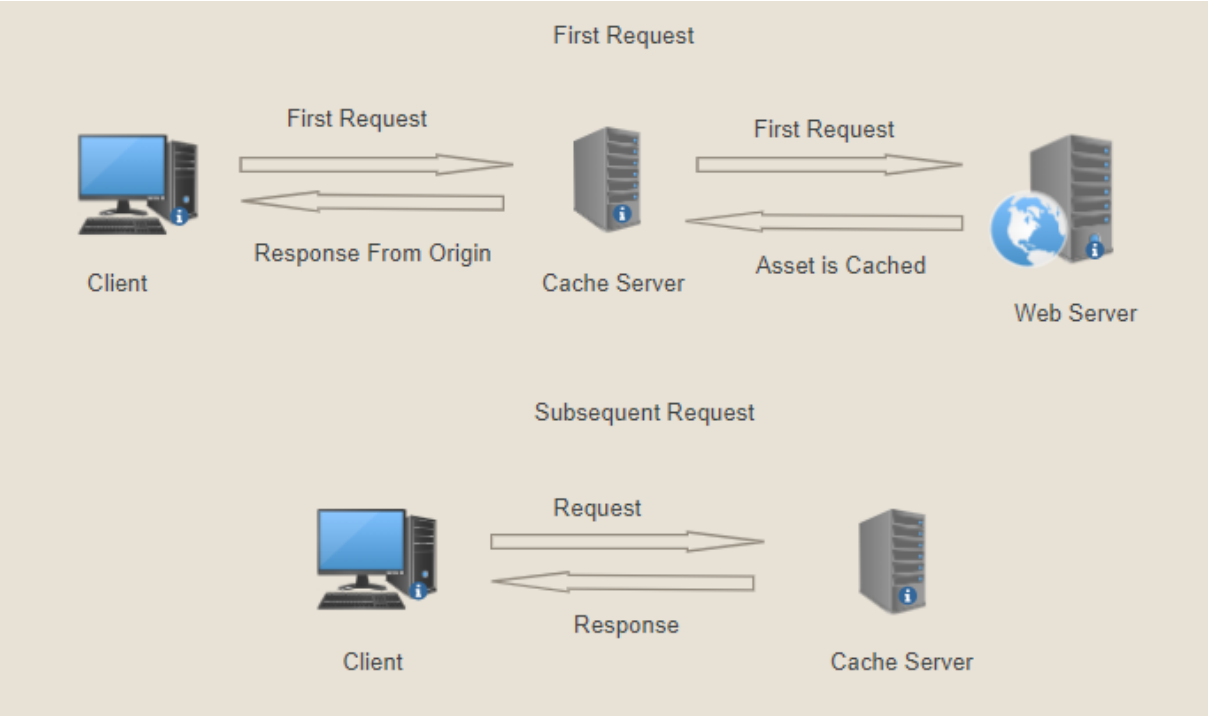


Fig. 3 Working of Cache Server

When a client requests a website, the query is directed to the cache server. If the cache contains the requested data, the cache server promptly fulfills the request; otherwise, the query is forwarded to the original web server, which then responds to the client. The data is stored in the cache server for future access. Subsequently, when the client requests the same website or data, the cache server efficiently handles the request, either providing the cached content directly or retrieving it from the original server if necessary. This process optimizes response times by serving frequently accessed data swiftly from the cache, reducing the load on the original server and enhancing overall system performance for seamless user experiences.

A. Cache Hit

In cache memory, a cache hit signifies successful data retrieval from the cache, providing expedited access compared to primary storage. It indicates the presence of requested data in the cache, reducing retrieval time and latency. Cache hits enhance system performance by accelerating access to frequently accessed data, refining response times and operational efficiency.

B. Cache Miss

A cache miss occurs when the desired data is absent from the cache, requiring retrieval from primary storage. This heightens latency as the system resorts to the original data source. Cache misses impede performance by delaying data retrieval, underscoring the importance of minimizing them to ensure streamlined system functionality and prompt data access in operational contexts.

Table 1 : Cache Hit Ratio and Cache Miss Ratio

Server Name	Cache Hit Ratio	Cache Miss Ratio
Nginx	84%	16%
Varnish	82%	18%
Apache	83%	17%

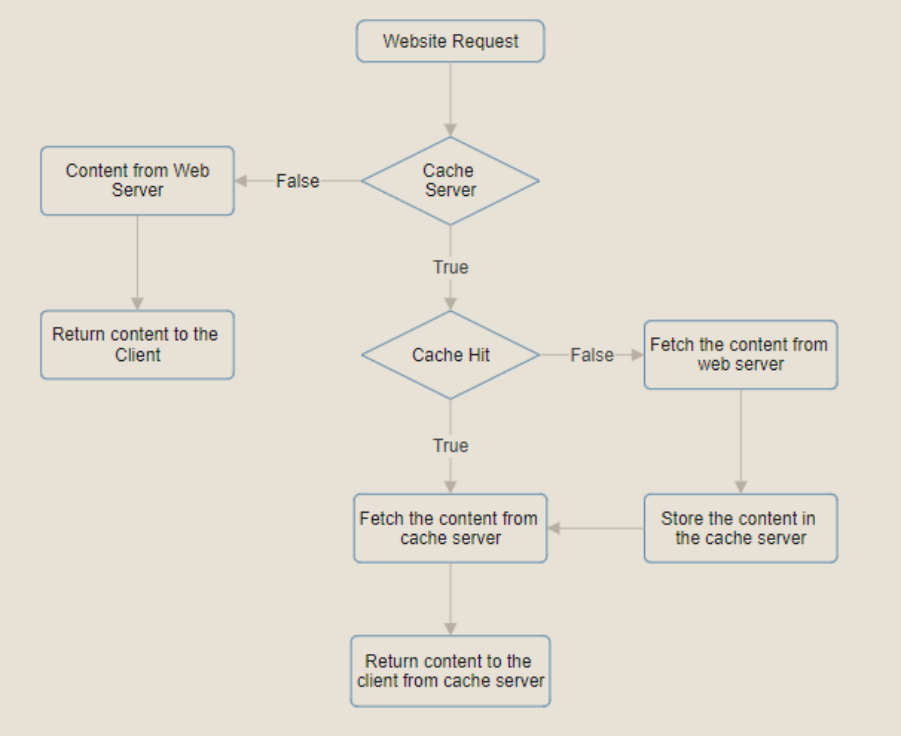


Fig.4 Working flow of Cache Server

C. CPU Utilization

CPU utilization is a measure of the CPU's workload, expressed as the percentage of time the CPU spends processing instructions. High utilization can negatively impact system performance, causing slower response times and potential bottlenecks. Conversely, low utilization indicates available CPU capacity that can be leveraged for additional tasks, optimizing system efficiency.

Table 2 : CPU Utilization

Server Name	Min(%)	Max(%)	Sum(%)	Average(%)
Nginx	0.71	1.02	3.24	0.81
Varnish	0.84	1.22	4.1	1.03
Apache	0.16	0.58	1.33	0.33

Table 3 : Performance and availability

Server Name	Performance	Availability
Nginx	100%	100%
Varnish	100%	100%
Apache	100%	100%

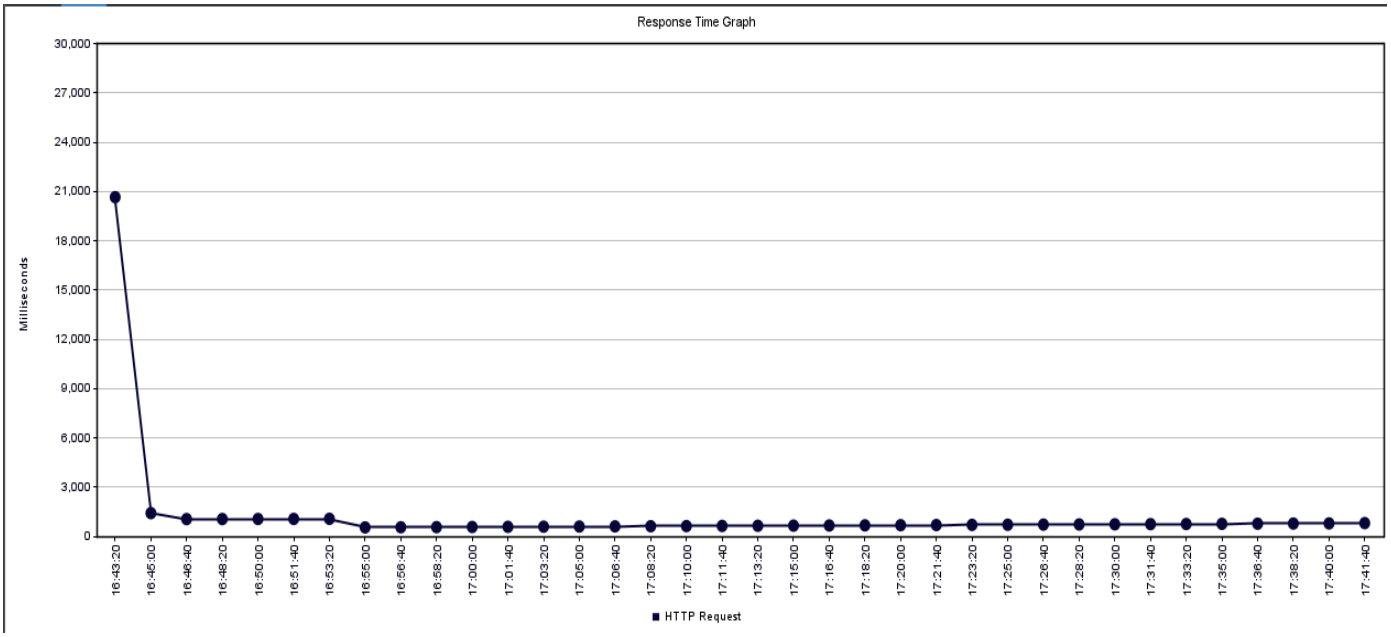
Table 4 : Memory Usage

Server Name	Memory Usage(%)
Nginx	25
Varnish	25
Apache	24

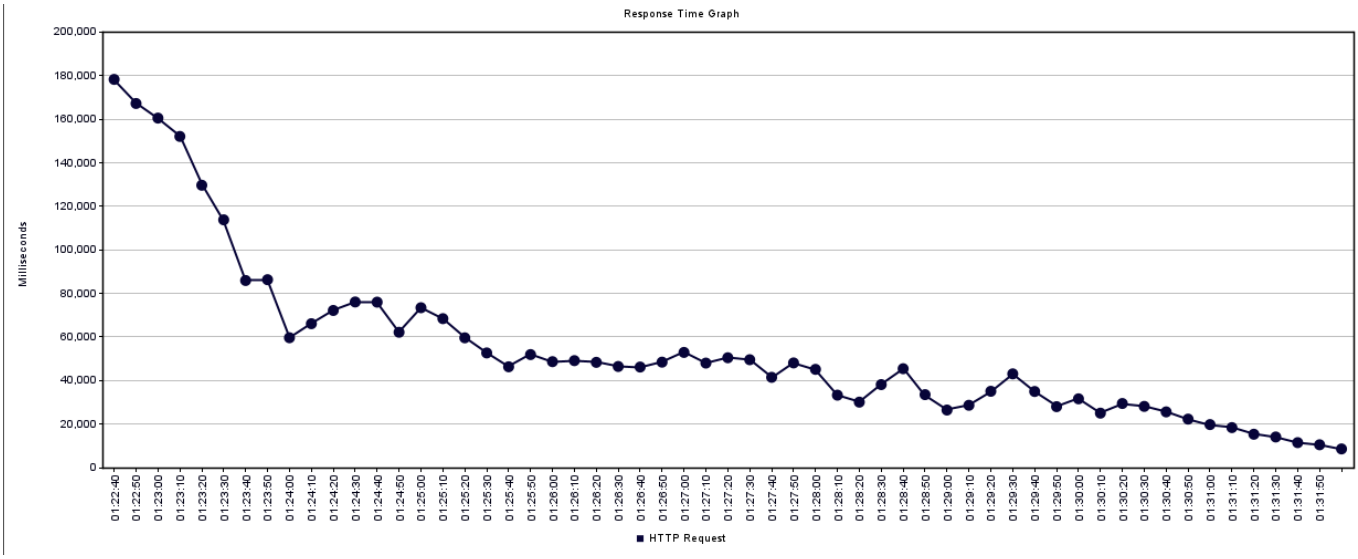
IV. RESULT AND DISCUSSION

The table 1 illustrates the cache performance of Nginx, Varnish, and Apache servers. Nginx achieves an 84% cache hit rate and 16% miss rate, Varnish records 82% hits and 18% misses, and Apache maintains 83% cache hits with 17% misses, reflecting the effectiveness of their caching systems.

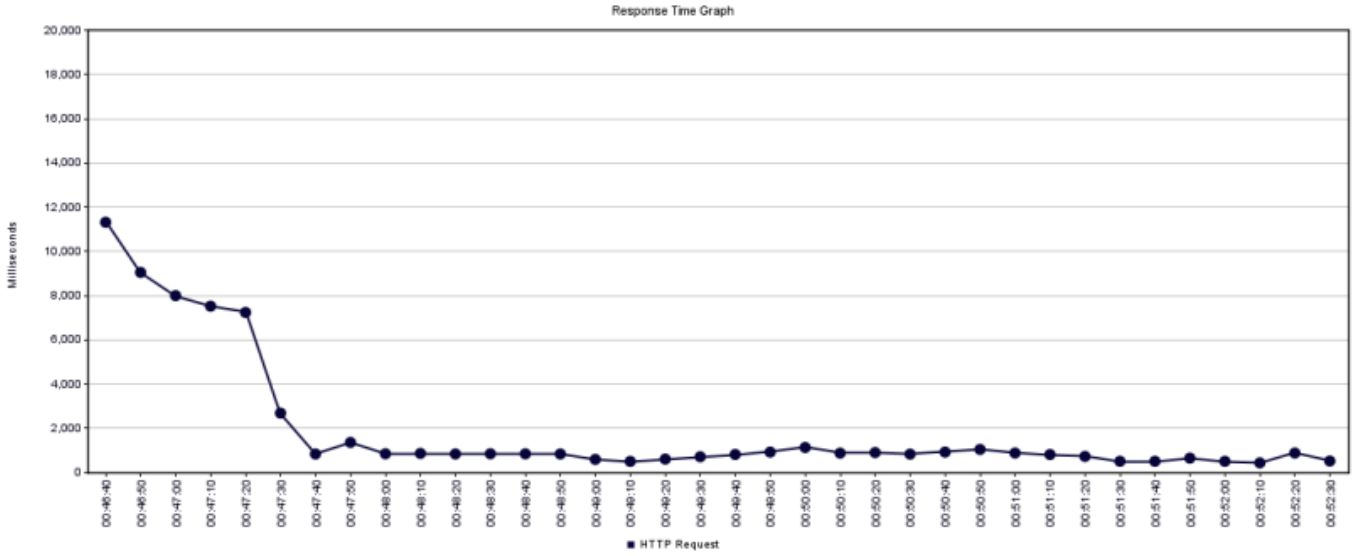
The table 2 displays CPU utilization metrics for Nginx, Varnish, and Apache servers. It includes their minimum, maximum, total, and average CPU percentages. Apache exhibited the lowest average utilization (0.33%), while Varnish had the highest (1.03%). Nginx fell in between with 0.81% average utilization.



Graph 1: Response Time Graph for Apache



Graph 2: Response Time Graph for Nginx



Graph 3: Response Time Graph for Varnish

The graph depicts the response times of HTTP requests. Based on the tables displaying cache hit ratios, cache miss ratios, CPU utilization, availability, and memory usage, as well as the response time graph, it can be concluded that the Nginx server outperforms Varnish and Apache servers in terms of efficiency and effectiveness. Nginx exhibits higher cache hit ratios, lower CPU utilization, and better overall performance metrics compared to the other two servers. This suggests that Nginx is better equipped to handle HTTP requests efficiently, providing faster response times and improved user experience.

Nginx excels over other servers due to its user-friendly configuration options tailored to HTTP status codes. By efficiently managing expiration times, Nginx optimizes cache storage by refreshing data promptly, ensuring up-to-date content delivery. This approach enhances server performance by effectively managing cache storage, enabling quick updates, and maintaining a responsive and dynamic web environment.

V. CONCLUSION

The data presented conclusively shows that Nginx outshines Varnish and Apache in efficiently handling HTTP requests. Nginx's superior cache hit ratios, lower CPU utilization, and overall performance make it the ideal choice for delivering fast, reliable responses to users. Nginx's user-friendly configuration, particularly its ability to manage expiration times based on HTTP status codes, enables efficient cache storage and prompt data refreshing, optimizing server performance through effective cache management and quick updates. This maintains a responsive, dynamic web environment. Nginx's technical advantages and user-friendly configuration make it the preferred server for web applications prioritizing speed, efficiency, and user experience.

REFERENCES

[1] M. Mostowfi, K. Christensen, S. Lee and J. Yun, "Timed redirection: HTTP request coalescing to reduce energy use of hybrid web servers," 37th Annual IEEE Conference on Local Computer Networks, Clearwater Beach, FL, USA, 2012, pp. 168-171

[2] A. Imtiaz and M. J. Hossain, "Distributed cache management architecture: To reduce the internet traffic by integrating browser and proxy caches," 2014 International Conference on Electrical Engineering and Information & Communication Technology, Dhaka, Bangladesh, 2014

[3] M. Kusuma, Widyawan and R. Ferdiana, "Performance comparison of caching strategy on wordpress multisite," 2017 3rd International Conference on Science and Technology - Computer (ICST), Yogyakarta, Indonesia, 2017, pp. 176-181

[4] T. Ma, Y. Hao, W. Shen, Y. Tian and M. Al-Rodhaan, "An Improved Web Cache Replacement Algorithm Based on Weighting and Cost," in IEEE Access, vol. 6, pp. 27010-27017, 2018

[5] S. Bommaraveni, T. X. Vu, S. Chatzinotas and B. Ottersten, "Active Content Popularity Learning and Caching Optimization With Hit Ratio Guarantees," in IEEE Access, vol. 8, pp. 151350-151359, 2020

[6] G. Ahani and D. Yuan, "Optimal Scheduling of Content Caching Subject to Deadline," in IEEE Open Journal of the Communications Society, vol. 1, pp. 293-307, 2020

[7] W. M. C. J. T. Kithulwatta, K. P. N. Jayasena, B. T. G. S. Kumara and R. M. K. T. Rathnayaka, "Performance Evaluation of Docker-based Apache and Nginx Web Server," 2022 3rd International Conference for Emerging Technology (INCET), Belgaum, India, 2022