Design and Implementation of Load Balancing for Quality of Service Improvement

Indrastanti Ratna Widiasari*[1], Rissal Efendi[2]
[1]Program Studi Teknik Informatika, [2]Program Studi Pendidikan Teknik Informatika dan Komputer
Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana
Jl. O Notohamidjojo 1-10 Salatiga 50715, Jawa Tengah, Indonesia
Email: [1]indrastanti@uksw.edu, [2]rissal.efendi@uksw.edu

Abstrak. Desain dan Implementasi Load Balancing untuk Peningkatan Kualitas Layanan*. Di Fakultas Teknologi Informasi UKSW diterapkan sistem load balancing dimana web server melayani maksimal 500 user dalam waktu yang sama. Hal ini untuk mencegah kelebihan beban atau downtime server selama akses simultan ke server web. Hasil pengujian menunjukkan perbedaan yang signifikan dalam penggunaan CPU, request time, dan throughput. Penggunaan load balancing lebih efektif dibandingkan hanya mengandalkan satu server, terbukti dari hasil pengujian. Penggunaan CPU dengan penyeimbangan beban jauh lebih rendah, dengan perbedaan hingga 45% dibandingkan dengan server tunggal. Waktu permintaan dengan penyeimbangan beban juga sedikit lebih baik, hanya 21,5 md dibandingkan dengan 42 md untuk satu server, menunjukkan perbedaan sebesar 20,5 md. Namun perbedaan bandwidth antara load balancing dan satu server tidak terlalu signifikan. Throughout tertinggi yang tercatat pada satu server adalah 182kb/s, sedangkan dengan load balancing mencapai 165kb/s, dengan selisih hanya 17kb/s di antara keduanya.*
Kata Kunci: *load balancing, throughput, quality of service*

Abstract. *At the Information Technology Faculty, Satya Wacana Christian University, load balancing systems are implemented where the web server serves 500 users. This is to prevent server overload or downtime during simultaneous access to the web server. Test results indicate significant differences in CPU usage, request time, and bandwidth between load balancing and single servers. The use of load balancing is more effective than relying on a single server, as evidenced by test results. The CPU usage with load balancing is significantly lower, with a difference of up to 45% compared to a single server. The request time with load balancing is also slightly better, with only 21.5ms compared to 42ms for a single server. However, the difference in bandwidth between load balancing and a single server is not very significant. The highest bandwidth recorded on a single server is 182kb/s, while with load balancing it reaches 165kb/s.*
Keywords: *load balancing, throughput, quality of service*

1. Introduction

The internet has become the backbone of digital society. It is a packet-switched distributed network that connects nearly all digital devices and is accessible worldwide [1]. As computer network technology advances, good performance is required, so a good network management system is needed. Network management can monitor existing conditions on the network so that it can avoid and minimize errors that occur [2] [3]. In the ever-evolving landscape of network infrastructure, ensuring optimal performance and service reliability is paramount. For organizations relying increasingly on digital resources and services, the demand for server resources continues to grow. However, the traditional approach of relying on a single server to handle all incoming requests is often insufficient in meeting the dynamically changing demands of users. The expansion of networks, the rise in user numbers, and the emergence of new technologies like cloud computing and big data have made managing traditional networks challenging. Load balancing is particularly crucial for meeting quality of service requirements, as it helps control and regulate data traffic across multiple resources, thereby enhancing network responsiveness, reliability, and capacity [4].

A computer network connects two or more physically and logically connected devices, enabling them to exchange data [5] [6]. If the devices in the network can exchange data and share their resources, then the network is considered to be connected. A computer network is the "interconnection" of two or more autonomous computers, linked by transmission media either through cables or wirelessly. Autonomous means that a computer does not have full control over another computer, such as making the other computer restart, shut down, lose files, or suffer system damage [7]. Certainly, one of the significant factors influencing the present configuration of the worldwide network and how information traverses it is the matter of traffic distribution, particularly within server infrastructures associated with World Wide Web (WWW) services. These infrastructures ensure users with efficient and dependable web browsing capabilities [8]. Internet connectivity issues can arise from either device malfunctions or problems with links. Even the gateway router, serving as the sole pathway to the internet service provider network, may encounter downtimes [9].

Quality of Service (QoS) is a term used to characterize the attributes of a network service, assessing the level of quality provided by that service [10]. Through the implementation of Quality of Service (QoS), bandwidth can be efficiently utilized, leading to an improvement in the quality of internet services experienced by users. The phenomenon of increasing traffic, coupled with varying resource demands, presents challenges in maintaining consistent Quality of Service (QoS). Issues such as high CPU usage, prolonged request times, and potential bottlenecks in bandwidth allocation can hinder the overall user experience. Recognizing these challenges, there arises a need for a more robust solution to distribute the workload efficiently across multiple servers while maintaining or enhancing QoS. The utilization of computer network technology as a means of data communication has been increasing steadily up to the present. Computer networks are employed to share resources efficiently in terms of both time and distance [11] [12]. The necessity for shared usage of resources within the network, whether software or hardware, has led to various advancements in network technology [13] [14]. Alongside the rising demand and the increasing number of network users seeking a network infrastructure that can provide maximum results in terms of efficiency and network security enhancement. Quality of Service (QoS) is a technology that allows network administrators to handle various congestion effects on the packet flow of different services to optimize network resource utilization instead of increasing the network's physical capacity [15] [16]. The objective of QoS mechanisms is to influence at least one of the four basic QoS parameters that have been specified [17].

Currently, numerous organizations are adopting cloud-based applications and platforms due to their on-demand services and rapid responsiveness. The primary concern with cloud computing is the potential for system overload for individuals, groups, or organizations. As a result, load balancing is becoming increasingly popular, with its algorithms and solutions continually improving [18]. Load balancing is a beneficial procedure that redistributes the workload across nodes, ensuring that no single node bears an excessive burden. Its primary goal is to achieve equilibrium among virtual machines, ensuring they are neither underloaded nor overloaded [19]. Load balancing emerges as a promising solution to address these challenges. By distributing incoming network traffic across multiple servers, load balancing optimizes resource utilization, minimizes response times, and enhances system reliability. However, the successful implementation of load balancing requires careful design and consideration of factors such as server capacity, network architecture, and traffic patterns.

Load Balancing involves evenly distributing resources and shifting heavily loaded tasks from overloaded nodes to those with lighter loads. Each node independently manages load balancing at the lowest level, utilizing its unique processing speed and capabilities to address tasks. To expedite task processing, workloads must be evenly distributed across all nodes within the grid computing system, each with varying processing capacities. Therefore, an imperative is to implement a uniform load-balancing algorithm capable of dynamically distributing workloads among nodes. There are two components: the front end and the back end. The front end is on the user side and is accessible via internet connections [20], and the back end is accessible via the local network. Several algorithms have been proposed to attain efficient load balancing, each aiming to effectively distribute data and enhance related performance metrics [21].

The Round-Robin (RR) Algorithm [22] operates sequentially and cyclically, where each process is allocated a fixed time slot with no assigned priority. This algorithm is widely used due to its straightforward implementation. To meet the constraints of Quality of Service (QoS) despite limited network availability, load balancing has been identified as a crucial factor. Consequently, multiple servers can be utilized with load balancers acting as the front end [23]. Load balancing software like HAProxy (High Availability Proxy) plays a crucial role in modern web service delivery. It distributes incoming requests among web servers to ensure efficient handling of traffic [24], so it will make the flexible architectural approach, easy to handle, cost-efficient, and easily adaptable, making it highly suitable for high-bandwidth and dynamic applications in today's context [25]. HAproxy has been utilized within the cloud environment, effectively handling all six million requests, each with four connections simultaneously. Instead of directing HTTP server traffic to a single server, it can be evenly distributed among a pool of servers. Consequently, HAproxy delivers prompt response times without dropping any requests, even under the load of six million simultaneous requests. This implementation aids in enhancing availability and reducing latency for HTTP requests through the utilization of an HTTP load balancer [26].

The case study at FTI UKSW provides a valuable opportunity to explore the design and implementation of load balancing as a means to improve QoS within an academic environment. By analyzing the existing infrastructure, assessing performance metrics, and deploying appropriate load-balancing techniques, insights can be gained into the effectiveness of this approach in enhancing the overall reliability and efficiency of network services. Therefore, this study aims to investigate the design and implementation of load-balancing strategies tailored to the specific requirements of FTI UKSW. Through rigorous testing and evaluation, the effectiveness of load balancing in improving QoS parameters such as CPU utilization, request times, and bandwidth allocation will be assessed. Additionally, considerations for scalability and future expansion, including the potential integration of database synchronization, will be explored to provide comprehensive insights into optimizing network performance within educational institutions.

2. Method

This type of research is development-oriented. The design method employed in the study utilizes the PPDIOO method, as shown in Figure 1. PPDIOO is a method developed by CISCO, capable of providing key steps in the success of network planning, whether in the stages of design, implementation, or operation. The phases in this PPDIOO method are prepare, plan, design, implement, operate, and optimize.

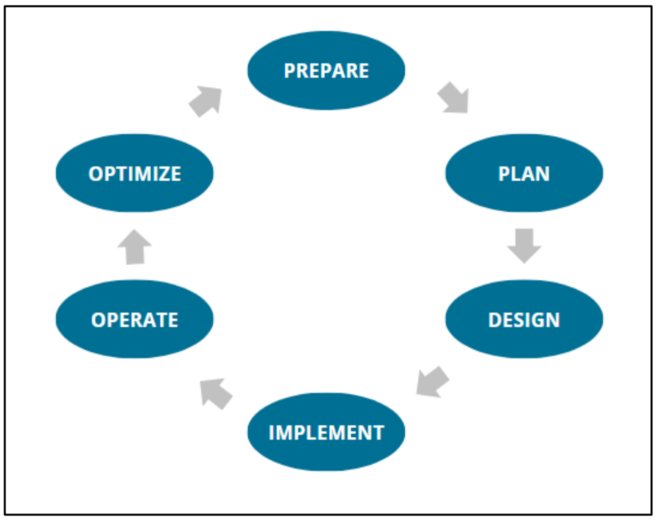

Figure 1. PPDIOO Method

The preparation stage involves defining organizational and business needs, developing network strategies, and proposing architectural concepts. During this stage, observations and interviews are conducted with the Information Technology Faculty UKSW to understand server issues during learning and teaching activities. In the planning stage, the analysis of existing problems, planning network requirements, conducting analysis, and project scheduling are carried out.

The planning stage involves identifying network requirements based on objectives, facilities, and user needs. After interviewing the network administrator in the Information Technology Faculty UKSW, it was concluded that there were issues with server overload during learning and teaching sessions. Therefore, load balancing implementation will be carried out on the server, focusing primarily on the web server using the HAProxy. HAProxy was chosen because it is easy to configure, capable of load balancing across multiple servers, and can implement a master-slave model, where if the main server fails, it will be automatically replaced by another slave/server.

The Design stage is the initial phase of designing the system based on the method to be used in building the load-balancing system, in the form of a network topology that will facilitate the construction of the network infrastructure. The general architecture design of the system installed at Information Technology Faculty UKSW, obtained after observation and interviews, explains the topology and overview of the network system used, as shown in Figure 2.

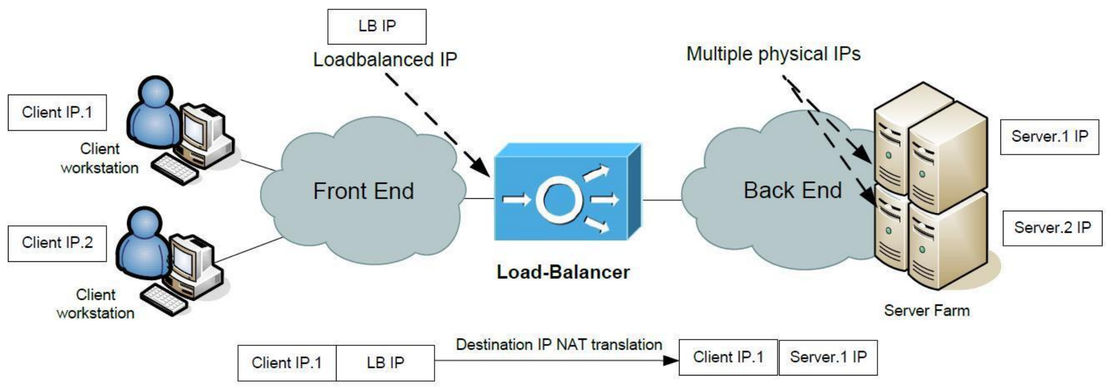

Figure 2. Load Balancing Topology with Multiple Servers Design

Figure 2 represents a general topology design, consisting of four servers: the web server, which contains the web content to be used during learning and teaching. Server 1 has the IP address of 192.168.33.1/24, server 2 has the IP address of 192.168.34.1/24, and server 3 has the IP address of 192.168.35.1/24. HAProxy itself has the IP address 192.168.15.15/24. In this topology, users will access the available web content, which will be received by HAProxy as an intermediary between the user and the server. HAProxy then checks all servers. If all servers are in an inadequate condition, HAProxy will direct the user's request to one of the servers. With 500 users, HAProxy will evenly distribute the requests to the servers to avoid overloading any single server. Once the server receives the request, it responds and displays the result in the user's web browser.

The implementation phase stands as the culmination of preceding preparatory steps and doubles as a crucial testing phase before transitioning into full operation. During this stage, load balancing is integrated into Ubuntu Server 16.04. Before this integration, essential packages like the Linux, Apache, MySQL, and PHP (LAMP) Server are installed on server 1, server 2, and server 3 these servers will undergo load balancing. Additionally, a separate PC is designated to function as the load balancer, employing HAProxy for this purpose. Once the implementation phase concludes, rigorous testing ensues, particularly during teaching and learning sessions. This testing phase ensures the efficacy and reliability of the load balancing setup, validating its ability to distribute workload efficiently and maintain system stability under real-world conditions.

Through meticulous testing and refinement, any potential issues or bottlenecks can be identified and addressed, paving the way for a seamless transition to the operational stage with enhanced performance and reliability.

Operating involves maintaining server resilience on a day-to-day basis. It includes managing and monitoring components, managing upgrade activities, managing performance, and identifying and correcting errors. During operation, the stability and performance of servers must be continuously monitored, errors detected, configurations corrected, and performance monitoring activities conducted. In this stage, monitoring is carried out on the previously tested servers.

Optimization is the stage of fine-tuning. In this stage, evaluation is conducted based on the design and testing carried out in the previous stages. If any issues arise, optimization is performed to improve the system and achieve good final results.

3. Result and Discussion

This research utilizes HAProxy version 1.6.3 running on Ubuntu Server 16.04. The HAProxy configuration is located in the haproxy.cfg file in the directory /etc/haproxy/haproxy.cfg. The configuration in /etc/haproxy/haproxy.cfg consists of two parts: the default section that already exists in the file and does not need to be changed. Manual configuration involves adding a front end and a back end. The frontend determines how a request should be forwarded to the backend, while the backend is a collection of servers that receive requests. The backend can contain one or multiple servers, and adding more servers to the backend increases the potential load capacity by distributing the load across several servers [7]. The configuration for the front end and back end can be seen in Figure 6. In the frontend section, identity is given as a label to forward requests to the backend with the name web frontend. Inside the web front end, there is a bind *:80, which connects web server 1, web server 2, and web server 3. HTTP mode is the primary focus for load balancing, as in this research, load balancing is performed on web servers using HTTP mode. Default_backend web endpoint serves as a connector between the front end and the backend to inform the backend about what is inside the front end.

The backend section is given the identity web endpoint. The round-robin balance algorithm is used, and the forward for option instructs the load balancer to forward the client's IP to the server. The server ubuntu1 192.168.35.1:80 check part checks the IP 192.168.35.1:80 using HAProxy and then assigns it as the IP to be load balanced with other server IPs. ubuntu1, ubuntu2, and ubuntu3 are the hostnames of the servers to be load balanced. A simple test was conducted to verify that the configuration was successful by accessing the HAProxy IP (192.168.15.15), and the previously configured web content appeared. The data in Table 1, Table 2, and Table 3 are the results of tests using the web server tool conducted for five minutes (300 seconds), and then the data is presented in graphical form to compare load balancing and single server performance. The graphs can be seen in Figure 3, Figure 4, and Figure 5.

Table 1. Percentage of CPU Usage

Time (second)	Number of Users	Load Balancing (%)	Single Server (%)
0	0	0	0
25	100	11,5	15
50	150	13	18
100	200	19	32
125	250	17	30
150	300	23	40
200	350	22	49
225	400	22	67
250	450	21	43
300	500	30	48

Table 1 serves as a visual representation of CPU usage dynamics in both load balancing and single server configurations across different time intervals. At the 0th second, a period marked by system inactivity, neither load balancing nor the single server experiences any CPU usage. This initial observation establishes a baseline of idle CPU activity, where no computational tasks are being executed due to the absence of user interactions or data processing requirements. At the 25th second, the scenario changes as 100 users begin accessing the system. With this increase in user activity, CPU usage begins to register, albeit at varying levels for load balancing and the single server. Load balancing demonstrates a CPU usage of 11.5%, while the single server exhibits a slightly higher usage of 15%. This discrepancy reflects the differing efficiencies in resource utilization between the two configurations, influenced by factors such as load distribution algorithms and hardware capabilities.

By the 50th second, the user count escalates to 150, triggering further adjustments in CPU usage. Load balancing maintains its efficiency, with CPU usage increasing marginally to 13%, indicative of its ability to scale resources in response to growing demands. Conversely, the single server configuration shows a more pronounced increase in CPU usage, reaching 18%. This disparity underscores the challenges inherent in managing resource allocation within a singular infrastructure, where spikes in user activity can exert greater strain on available resources. This pattern persists throughout the observation period until the 300th second, when the system experiences peak activity with 500 users accessing simultaneously. At this point, load balancing and the single server exhibit distinct CPU usage levels, reflecting their respective capacities to handle the heightened workload efficiently. Table 1 shows how CPU usage varies across different user load scenarios in both load balancing and single server setups. These insights enable administrators to make informed decisions regarding resource allocation, capacity planning, and system optimization strategies, ultimately enhancing overall system performance and user experience.

Table 2. Request Time

Time (second)	Number of Users	Load Balancing (ms)	Single Server (ms)
0	0	0	0
25	100	27,5	40
50	150	26,5	37
100	200	29	27
125	250	21,5	28
150	300	22,5	30
200	350	24	37
225	400	21,5	39
250	450	25,5	42
300	500	23	38

Table 2 presents a sequential depiction of events as follows: initially, at the 0th second, the absence of activity is evident, with no users accessing the system. Consequently, both load balancing and single server setups exhibit idle states in terms of request time, reflecting the lack of ongoing processing or user interaction. However, as the timeline progresses, user engagement initiates, with 100 users accessing the system by the 25th second. This influx of user activity prompts a corresponding increase in request time, with load balancing and the single server experiencing request times of 27.5ms and 40ms, respectively. This disparity in request time suggests that load balancing is more efficient in handling user requests, exhibiting a quicker response compared to the single server configuration.

Subsequently, at the 50th second, the user count escalates to 150, resulting in further adjustments to request time. Load balancing demonstrates resilience in managing the increased workload, maintaining a request time of 26.5ms, while the single server experiences a slight increase to 37ms. This trend underscores the comparative efficiency of load balancing in distributing requests across multiple servers, mitigating the impact of increased user loads on response times. The subsequent intervals continue to reflect this pattern, with user count fluctuations corresponding to request time variations. This dynamic is exemplified at the 300th second when the system experiences peak activity with 500 users accessing simultaneously. At this critical juncture, load balancing and the single server exhibit distinct request times, reflecting their respective capabilities in handling the heightened demand. In essence, Table 2 offers insight into the responsiveness of load balancing and single server setups to varying user loads. By presenting a granular view of request time dynamics, it facilitates a nuanced understanding of system performance under different conditions, informing decision-making processes related to resource allocation and optimization strategies.

Table 3. Throughput

Time (second)	Number of Users	Load Balancing (ms)	Single Server (ms)
0	0	0	0
25	100	126	127
50	150	118	120
100	200	87	154
125	250	143	180
150	300	140	182
200	350	98	120
225	400	110	130
250	450	160	182
300	500	166	177

In Table 3, the initial observation at the 0th second indicates a lack of activity, with no users accessing the system. Consequently, both load balancing and single server setups exhibit negligible bandwidth usage, reflecting the absence of data transmission. However, as time progresses, user activity initiates, with 100 users accessing the system by the 25th second. This uptick in user engagement prompts a corresponding increase in bandwidth utilization, with load balancing and the single server recording 126kb/s and 127kb/s, respectively.

Subsequently, at the 50th second, the user count escalates to 150, resulting in adjustments to bandwidth consumption. Load balancing registers a bandwidth usage of 118kb/s, while the single server shows a slight increase to 120kb/s. This pattern continues as the user count fluctuates, reaching its zenith at the 300th second when 500 users are actively engaging with the system. At this juncture, load balancing and the single server exhibit corresponding bandwidth usage, reflecting the culmination of user activity and the consequent strain on network resources. The progression depicted in Table 3 underscores the dynamic nature of bandwidth utilization in response to varying user loads. As user activity intensifies, both load balancing and single server setups adapt by allocating and managing network resources to maintain optimal performance and user experience.

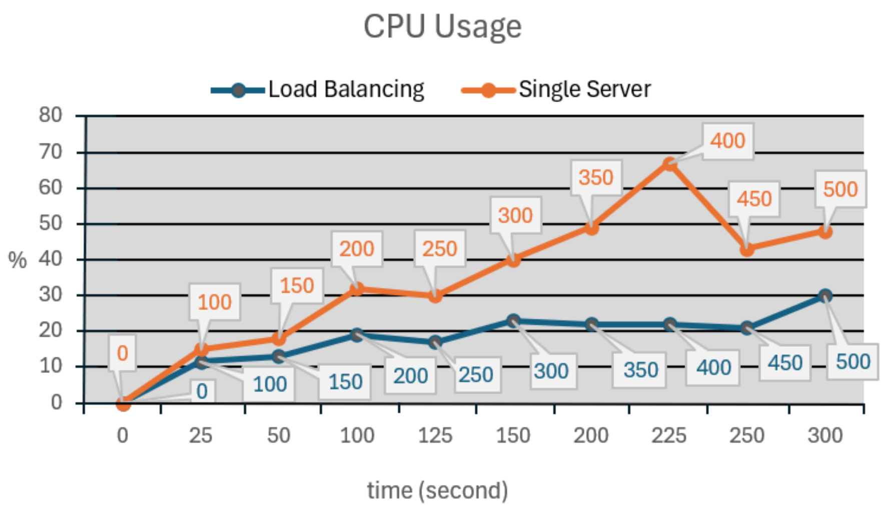

Figure 3. CPU Usage

Figure 3 provides a stark comparison between the utilization of load balancing and a single server. In the context of load balancing, the utilization of CPU resources maintains a relatively stable pattern over time, fluctuating within the range of 10% to 30%. This consistency suggests an efficient distribution of workload across multiple servers, ensuring that no single server bears an excessive burden. Conversely, the utilization of CPU resources in the single server setup exhibits notable instability, with fluctuations spanning from 15% to 70%. This erratic behavior indicates resource allocation and management challenges within a single-server environment. The most pronounced surge in CPU usage within the single server setup occurs at the 225th second, coinciding with a load of 400 users, nearly peaking at 70%. This surge highlights the strain experienced by the solitary server when confronted with a sudden increase in user demand. Such spikes in CPU usage can lead to performance degradation, as the server struggles to cope with the heightened workload.

The contrasting patterns of CPU utilization between load balancing and a single server underscore the advantages of employing load balancing techniques. Load balancing facilitates a more even distribution of computational tasks, thereby preventing individual servers from becoming overwhelmed. This balanced distribution not only promotes stability in CPU usage but also enhances overall system resilience and responsiveness. In contrast, the fluctuations and occasional spikes in CPU usage observed in the single server setup underscore the inherent limitations of relying on a solitary resource for handling variable workloads. Without the ability to distribute tasks across multiple servers, the single server is susceptible to performance bottlenecks and resource exhaustion during periods of high demand. Overall, Figure 3 illustrates the performance disparities between load balancing and single server usage. By maintaining consistent CPU usage levels and effectively managing workload distribution, load balancing emerges as a more robust solution for accommodating fluctuating user demands and ensuring optimal system performance.

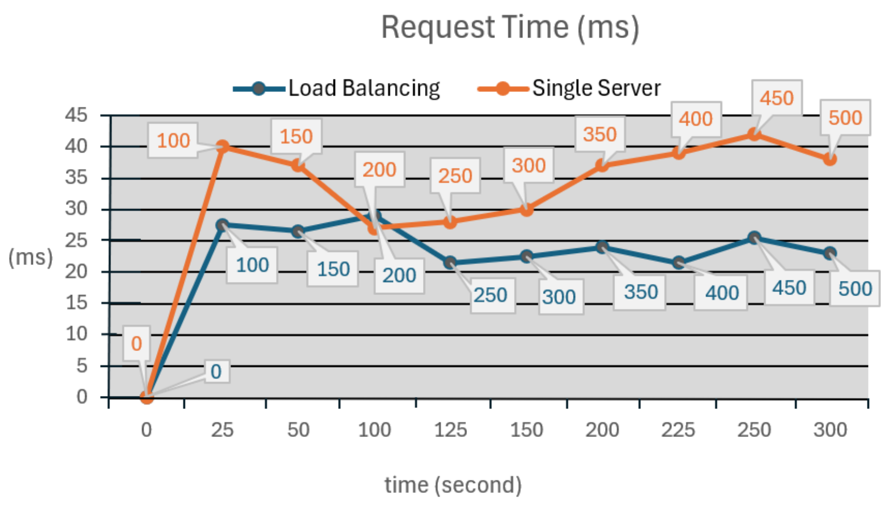

Figure 4. Request Time

The analysis of request time occurrence in load balancing displays variations, with a noticeable surge identified around the 100th second, reaching nearly 30 ms. Request duration under load balancing varies between 20 ms and 30 ms. Likewise, within a single-server arrangement, fluctuations in request time are present, albeit generally higher compared to load balancing. The lowest point arises at the 100th second with 200 users, approximately at 27 ms, whereas the apex is noted at the 250th second with 450 users. As request time escalates, the server's handling of requests prolongs. Fluctuations in request time are typical in server settings and can be affected by diverse factors such as the number of active users, server burden, and resource availability. In the context of load balancing, these fluctuations may be influenced by the system's methodology in distributing traffic across available servers. At specific intervals, one server may endure a heavier load than others, resulting in spikes in request time for that server.

Nevertheless, a comparison between load balancing and a solitary server configuration reveals that while fluctuations persist in both scenarios, request time tends to exhibit greater stability in load balancing. This is because load balancing actively oversees and evens out traffic distribution among available servers, mitigating the likelihood of significant spikes in request time on any particular server. It's essential to acknowledge that heightened request times can detrimentally affect application performance and user experience. With increasing request times, users may encounter extended response durations or even encounter difficulty accessing services. Hence, monitoring and managing request time represent pivotal components of server infrastructure management to ensure operational efficiency and optimal performance. Through comprehension of the patterns of request time fluctuations in both load balancing and single-server setups, system administrators can implement measures to enhance system performance and elevate overall user experience. This could involve adjustments to server configurations, augmentation of resource capacities, or adoption of more efficient load-balancing strategies. Thus, an in-depth understanding of request time serves as a cornerstone in constructing resilient and dependable server infrastructure.

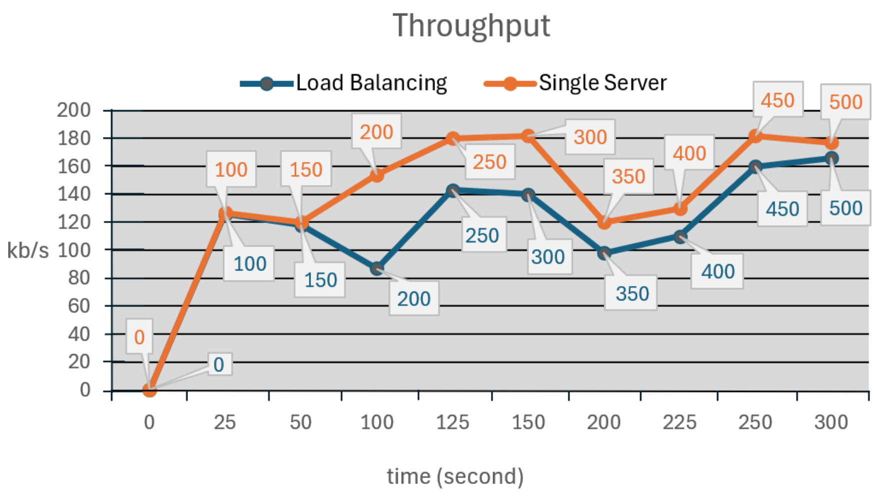

Figure 5. Throughput

Figure 5 presents a detailed comparison of throughput utilization between single server and load balancing configurations, revealing subtle distinctions in their performance. In the case of the single server setup, bandwidth usage fluctuates within a relatively narrow range, spanning from 120kb/s to 185kb/s. The highest bandwidth consumption occurs at the 150th second, registering 182kb/s while accommodating 300 users. This peak indicates a period of heightened data transfer demands, potentially straining the server's network resources. Conversely, in the load balancing scenario, bandwidth usage also demonstrates variability but within a slightly different range, ranging from 80kb/s to 165kb/s. The peak bandwidth utilization in the load balancing setup is observed at the 300th second, coinciding with 500 users, reaching 165kb/s. This peak underscores a moment of intensified data transmission requirements across the load-balanced servers, possibly due to a surge in user activity or data-intensive operations. While the disparities in bandwidth utilization between the single server and load balancing setups are relatively modest, they offer insights into resource allocation and workload distribution efficiency. The narrower bandwidth fluctuations in the single server setup suggest a more consistent usage pattern, possibly indicating optimized network management strategies or less dynamic user activity. On the other hand, the broader range of bandwidth utilization in load balancing reflects the dynamic nature of distributing network traffic across multiple servers, adapting to fluctuating user demands, and ensuring optimal resource utilization. Overall, Figure

1 illustrates the nuanced differences in bandwidth utilization between a single server and load-balancing configurations. By evaluating these patterns, administrators can gain valuable insights into network performance and make informed decisions regarding resource allocation and system optimization strategies.

4. Conclusion

Based on the research conducted, namely the design and implementation web server load balancing using HAProxy, it can be concluded that using load balancing on the Information Technology Faculty UKSW server can reduce the load on the server used for CBT which previously only used a single server, where if only using a single server could result in overload if accessed by 500 students. The use of load balancing is more effective than just using a single server, it can be seen in the test results that CPU usage in load balancing is less, and the difference in CPU usage between load balancing and a single server can reach 45%. Request time on load balancing is slightly better than a single server, which reaches 42ms, while load balancing is only 21.5ms; the difference is 20.5ms. The bandwidth on load balancing and single server is not that significant; on a single server, the highest bandwidth is 182kb/s, and on load balancing, the highest is 165kb/s; the difference between the two is only 17kb/s. From the results of implementing load balancing on the Information Technology Faculty Satya Wacana Christian University server, it is necessary to add servers both in software and hardware so that their performance is even better, and perhaps we can add a database server which in the future can be developed to include database synchronization.

References

[1] S. K. Keshari, V. Kansal, and S. Kumar, "A Systematic Review of Quality of Services (QoS) in Software Defined Networking (SDN)," *Wireless Personal Communications*, vol. 116, no. 3, pp. 2593–2614, 2021. doi: 10.1007/s11277-020-07812-2.

[2] J. Moedjahedy, "Implementasi Point to Point Jaringan Internet Nirkabel di SMA Universitas Klabat," *CogITo Smart Journal*, vol. 2, no. 2, pp. 240–249, 2016. doi: 10.31154/cogito.v2i2.33.240-249.

[3] F. G. J. Rupilele and A. Palilu, "Rancang Bangun Sistem Informasi Manajemen Pengaduan Masyarakat dan Monitoring Knerja Akademik Perguruan Tinggi," *Jurnal Sistem Informasi dan Komputer*, vol. 8, no. 2, pp. 141–148, 2019. doi: 10.32736/sisfokom.v8i2.672.

[4] K. A. Jadhav, M. M. Mulla, and D. G. Narayan, "An Efficient Load Balancing Mechanism in Software Defined Networks," in *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*, Sep. 2020, pp. 116–122. doi: 10.1109/CICN49253.2020.9242601.

[5] K. Nugroho and A. Y. Kurniawan, "Uji Performansi Jaringan menggunakan Kabel UTP dan STP," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, dan Teknik Elektronika*, vol. 5, no. 1, p. 48, 2018. doi: 10.26760/elkomika.v5i1.48.

[6] Q. A. H. H. Rohman and N. S. Salahuddin, "Rancang Bangun Prototipe Mobil Penjelajah Dengan Kendali Jarak Jauh Melalui Jaringan Wi-Fi Berbasis Antarmuka Web," *Teknika*, vol. 7, no. 1, pp. 1–7, 2018. doi: 10.34148/teknika.v7i1.79.

[7] A. S. Dina and D. Manivannan, "Intrusion Detection Based on Machine Learning Techniques in Computer Networks," *Internet of Things*, vol. 16, 100462, 2021. doi: 10.1016/j.iot.2021.100462.

[8] P. Dymora, M. Mazurek, and B. Sudek, "Comparative Analysis of Selected Open-Source Solutions for Traffic Balancing in Server Infrastructures Providing WWW Service," *Energies*, vol. 14, no. 22, p. 7719, 2021. doi: 10.3390/en14227719.

[9] F. U. Raharjo, F. Khair, J. G. A. Ginting, and E. F. Cahyadi, "TCP and UDP Traffic Performance Trade-off on VRRP-BGP Routing Protocol in VyOS," in *2023 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, Nov. 2023, pp. 536–541. doi: 10.1109/COMNETSAT59769.2023.10420818.

[10] V. Y. P. Ardhana and M. D. Mulyodiputro, "Analisis Quality of Service (QoS) Jaringan Internet Universitas Menggunakan Metode Hierarchical Token Bucket (HTB)," *Journal of Informatics Management and Information Technology*, vol. 3, no. 2, pp. 70–76, 2023. doi: 10.47065/jimat.v3i2.257.

[11] S. K. Shandilya, S. Upadhyay, A. Kumar, and A. K. Nagar, "AI-assisted Computer Network Operations testbed for Nature-Inspired Cyber Security based adaptive defense simulation and analysis," *Future Generation Computer Systems*, vol. 127, pp. 297–308, 2022. doi: 10.1016/j.future.2021.09.018.

[12] X. Zhu and S. Luo, "The influence of computer network technology on national income distribution under the background of social economy," *Computer Communications*, vol. 177, pp. 166–175, 2021. doi: 10.1016/j.comcom.2021.06.025.

[13] A. Burik, "Using Technology to Help Students Set, Achieve, and Publicize Goals," *Adult Literacy Education: The International Journal of Literacy, Language, and Numeracy*, vol. 3, no. 1, pp. 83–89, 2021. doi: 10.35847/ABurik.3.1.83.

[14] H. G. Ogelman, H. Güngör, Ö. Körükçü, and H. Erten Sarkaya, "Examination of the relationship between technology use of 5–6 year-old children and their social skills and social status," *Early Child Development and Care*, vol. 188, no. 2, pp. 168–182, 2018. doi: 10.1080/03004430.2016.1208190.

[15] Y. Ben Slimen, J. Balcerzak, A. Pagès, F. Agraz, S. Spadaro, K. Koutsopoulos, M. Al-Bado, T. Truong, P. G. Giardina, and G. Bernini, "Quality of perception prediction in 5G slices for e-Health services using user-perceived QoS," *Computer Communications*, vol. 178, pp. 1–13, 2021. doi: 10.1016/j.comcom.2021.07.002.

[16] B. Vijay Kumar, S. Musthak Ahmed, and M. N. Giri Prasad, "Efficient method to identify hidden node collision and improving Quality-of-Service (QoS) in wireless sensor networks," *Materials Today: Proceedings*, vol. 80, pp. 1747–1750, 2023. doi: 10.1016/j.matpr.2021.05.498.

[17] D. H. H. Hailu, G. G. Lema, B. G. Gebrehaweria, and S. H. Kebede, "Quality of Service (QoS) improving schemes in optical networks," *Heliyon*, vol. 6, no. 4, e03772, 2020. doi: 10.1016/j.heliyon.2020.e03772.

[18] S. Khare, U. Chourasia, and A. J. Deen, "Load Balancing in Cloud Computing," in *Proceedings of the International Conference on Cognitive and Intelligent Computing*, 2022, pp. 601–608. doi: 10.1007/978-981-19-2350-0_58.

[19] K. D. Patel and T. M. Bhalodia, "An Efficient Dynamic Load Balancing Algorithm for Virtual Machine in Cloud Computing," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, May 2019, pp. 145–150. doi: 10.1109/ICCS45141.2019.9065292.

[20] I. Odun-Ayo, M. Ananya, F. Agono, and R. Goddy-Worlu, "Cloud Computing Architecture: A Critical Analysis," in *2018 18th International Conference on Computational Science and Applications (ICCSA)*, Jul. 2018, pp. 1–7. doi: 10.1109/ICCSA.2018.8439638.

[21] A. Jangra and N. Mangla, "Cloud Load Balancing Using Optimization Techniques," in *Proceedings of the International Conference on Cognitive and Intelligent Computing*, 2021, pp. 735–744. doi: 10.1007/978-981-15-7130-5_60.

[22] N. S. Kaurav and P. Yadav, "A genetic algorithm-based load balancing approach for resource optimization for cloud computing environment," *International Journal of Information and Computing Science*, vol. 6, no. 3, pp. 175–184, 2019.

[23] H. Babbar, S. Parthiban, G. Radhakrishnan, and S. Rani, "A genetic load balancing algorithm to improve the QoS metrics for software defined networking for multimedia applications," *Multimedia Tools and Applications*, vol. 81, no. 7, pp. 9111–9129, 2022. doi: 10.1007/s11042-021-11467-x.

[24] A. Kumar, G. Somani, and M. Agarwal, "Comparing HAProxy Scheduling Algorithms During the DDoS Attacks," *IEEE Networking Letters*, vol. 1, pp. 1–1, 2024. doi: 10.1109/LNET.2024.3383601.

[25] A. Khaliq, M. A. Tahir, G. Nadeem, S. H. Adil, J. Jamshid, and J. A. Memon, "Performance comparison of Webservers load balancing using HAProxy in SDN," in *2023 4th International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, Mar. 2023, pp. 1–5. doi: 10.1109/iCoMET57998.2023.10099326.
[26] Y. A. H. Omer, M. A. Mohammedel-Amin, and A. B. A. Mustafa, "Load Balance in Cloud Computing using Software Defined Networking," in *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, Feb. 2021, pp. 1–6. doi: 10.1109/ICCCEEE49695.2021.9429607.