UDC 632:654.672
DOI https://doi.org/10.32782/tnv-tech.2024.4.2

APACHE WEB SERVER PERFORMANCE OPTIMIZATION

Antonenko A. V. – *PhD in Technical Sciences, Associate Professor, Associate Professor at the Department of Standardization and Certification of Agricultural Products of the National University of Life and Environmental Sciences of Ukraine*
ORCID ID: 0000-0001-9397-1209

Mishkur Yu. V. – *Postgraduate Student at the Department of Computer Engineering of the State University of Information and Communication Technologies*
ORCID ID: 00009-0004-6807-6914

Solskyi D. Ya. – *Postgraduate Student at the Department of Computer Engineering of the State University of Information and Communication Technologies*
ORCID ID: 0009-0005-0351-5987

Solobaiev S. H. – *Postgraduate Student at the Department of Computer Engineering of the State University of Information and Communication Technologies*
ORCID ID: 0009-0008-6298-4777

Poduran D. V. – *Master at the Department of Computer Engineering of the State University of Information and Communication Technologies*
ORCID ID: 0009-0005-4177-5456

Sarafaniuk R. O. – *Master at the Department of Computer Engineering of the State University of Information and Communication Technologies*
ORCID ID: 0009-0000-5284-2910

This article explores approaches to online optimization of the Apache web server, focusing on the MaxClients parameter. Using empirical and analytical methods, the researchers prove that MaxClients has a large impact on response time, and recommend hill-climbing strategies to determine the optimal value of MaxClients. The study includes the analysis of two optimizers using different approaches, such as Newton's method and fuzzy control, as well as heuristics based on the relationship between resource utilization and response time. In general, online optimization techniques can reduce the response time by a factor of 10 or more compared to the static default, although this may require some trade-offs between different approaches. Investigating opportunities to improve the speed and response time of the Apache web server through various techniques and settings such as optimizing server settings, using caching, data compression, optimizing request routing, and others is really important in today's Internet environment. The purpose of the study is to improve the performance and response speed of the Apache web server, which can be useful for developers and administrators of web applications and services. The speed and response time of servers are important factors in meeting user needs and achieving business goals of web applications and services. Since Apache is one of the most widely used web servers in the world, optimizing Apache server response time is an important task for many web development and administration professionals. This study examines various approaches and techniques for optimizing the response time of the Apache web server, including configuring server parameters, using caching, data compression, optimizing request routing, and others. The results of the study can be useful for developers and administrators of web applications and services that work with the Apache web server. Optimizing Apache server response time can significantly improve the performance and efficiency of web applications and services, which in turn can lead to user satisfaction and business goals.

Key words: *Apache web server, Apache MaxClients, Apache architecture, optimization, fuzzy control, heuristics.*

Антоненко А. В., Мішкур Ю. В., Сольський Д. Я., Солобаєв С. Г., Подуран Д. В., Сарафанюк Р. О. Оптимізація продуктивності веб-сервера Apache

У цій статті досліджуються підходи до онлайн-оптимізації веб-сервер Apache, зосереджуючись на параметрі MaxClients. З використанням емпіричних та аналітичних методів дослідники доводять, що MaxClients має великий вплив на час відгуку, і рекомендують використовувати стратегії для підняття на гору для визначення оптимального значення MaxClients. Дослідження включає аналіз двох оптимізаторів, що використовують різні підходи, такі як метод Ньютона і нечітке керування, а також евристику, яка базується на зв'язку між використанням ресурсів та часом відгуку. Загалом, методи онлайн-оптимізації дозволяють скоротити час відповіді у 10 або більше разів порівняно зі статичним значенням за замовчуванням, хоча це може вимагати деяких компромісів між різними підходами. Дослідження можливостей покращення швидкості та часу реакції веб-сервера Apache за допомогою різноманітних технік і налаштувань, таких як оптимізація налаштувань сервера, використання кешування, стиснення даних, оптимізація маршрутизації запитів та інших, дійсно має велике значення у сучасному Інтернет-середовищі. Мета дослідження полягає в поліпшенні продуктивності та швидкості відповіді веб-сервера Apache, що може бути корисним для розробників та адміністраторів веб-додатків та сервісів. Швидкість та час відповіді серверів є важливими факторами для задоволення потреб користувачів та досягнення бізнес-цілей веб-додатків та сервісів. Оскільки Apache є одним з найпоширеніших веб-серверів у світі, оптимізація часу відповіді сервера Apache є важливим завданням для багатьох фахівців у галузі веб-розробки та адміністрування. В даному дослідженні розглянуті різні підходи та техніки для оптимізації часу відповіді Apache веб-сервера, включаючи налаштування параметрів сервера, використання кешування, стиснення даних, оптимізацію маршрутизації запитів та інші. Результати дослідження можуть бути корисними для розробників та адміністраторів веб-додатків та сервісів, які працюють з Apache веб-сервером. Оптимізація часу відповіді Apache серверу може суттєво покращити продуктивність та ефективність веб-додатків та сервісів, що, в свою чергу, може призвести до задоволення користувачів та досягнення бізнес-цілей.

Ключові слова: *веб-сервер Apache, Apache MaxClients, архітектура Apache, оптимізація, нечіткий контроль, евристика.*

Introduction. The problem is that the response time of the Apache Web server can be quite high depending on the load and the number of requests coming to the server. This can lead to a poor user experience and adversely affect the performance of the website. Optimizing the response time of the Apache Web server can include various strategies such as caching static files, adjusting the logging level, improving the processing of requests using different algorithms, adjusting server configuration options, and optimizing the database if it is used. The result of optimization can be a decrease in server response time and an increase in site performance, which can positively affect the user experience and site ranking in search engines.

The aim of the study. The goal of optimizing Apache web server response time is to improve the performance and efficiency of web applications and services that use this web server. To achieve this goal, various approaches and methods for optimizing the response time of Apache web servers should be considered, such as server configuration, use of caching, data compression, optimization of request routing, etc. Optimizing the response time of the Apache server can significantly improve the performance and efficiency of web applications and services, resulting in the satisfaction of user needs and the achievement of business goals.

The objects of research for optimizing the response time of the Apache web server are the web server itself, its configuration parameters and interaction between clients and server applications. The study should cover web server settings such as network settings, operating system configuration, server application settings, request processing, and response sending.

The process by which the server responds to client requests and its parameters are the subject of research in the work devoted to optimizing the response time of the

Apache web server. Research tasks are request processing speed, response time, number of requests per unit of time, number of simultaneous connections and other parameters that can affect server response time.

Analysis of recent research and publications. With the proliferation of e-commerce systems, service quality, particularly response time, is attracting increasing attention. One of the challenges in this context is adapting systems to changing workloads by optimizing them through online configuration. This article reviews the following approaches to online optimization on Apache web servers, with an emphasis on methods that are less invasive and can be applied to a wide range of configurations and systems.

Consider the Apache MaxClients parameter, which determines the number of requests processed by the web server in parallel. Table 1 shows the average response times measured with different configurations of MaxClients for different workloads. The test environment used to collect this data will be described later in this section.

Table 1

Response time for various loads, sec.

MaxClients	Workload	
	Dynamic	Dynamic + Static
150	50	
650	1	15
900	30	2

It is often found that the optimal value of MaxClients varies depending on the type of pages visited by the site. Because real-world workloads can change rapidly, optimizing these important parameters online can significantly improve them [1, 2].

This article describes a general approach to online optimization of the response time of the widely used Apache web server. A related area of research is differentiated services, which are aimed at achieving response time goals for different classes of tasks. The authors use an analog-integral controller to adjust and differentiate the response time. A multi-input multi-output controller scheme is used to regulate server CPU and memory usage within a given QoS value, and an approach that combines queuing theory and control theory to regulate response time is described. Unfortunately, the control problem that this approach solves is very different from the optimization problem [3, 4]. Essentially, regulation (eg, providing response time targets for gold and silver services) defines how to «cut the pie», and optimization (eg, minimizing response time for a class of service) is what it does. Several studies have been conducted in the field of optimization of online resources in computer systems. This study describes an Apache implementation that manages web server resources based on profit maximization (e.g., response within 8 seconds to avoid pushing users away). Although the results are interesting, this approach requires significant changes to Apache's resource management scheme. This approach considers benefit maximization in service level agreements for web server farms, but in a way that is based on an accurate analytical model of the managed system.

Presentation of the main research material. Recently, a fuzzy control approach was proposed, which allows to minimize the response time by combining a feedback control scheme with a qualitative analysis of the influence of configuration parameters on QoS. Unfortunately, this approach has a long convergence time [6].

Figure 1 shows our proposed architecture. The target system (e.g., Apache) provides one or more configuration parameters (e.g., MaxClients) that are dynamically changed by the optimizer to optimize the measured variable (e.g., response time). First, MaxClients is shown to have an ascending effect on response time, and therefore a hill-climbing method can be used to determine the optimal value of MaxClients. Two hill-climbing optimizers are studied, one based on Newton's method and the other based on fuzzy control.

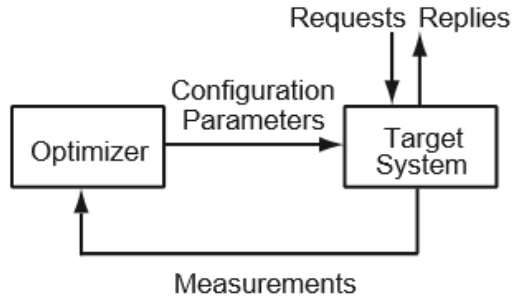

Fig. 1. General architecture for online optimization. The target system is controlled by configuration parameters that are dynamically changed by the optimizer in response to changing workloads

The third method is a heuristic method that uses the observed relationship between loading bottlenecks and minimizing response time. Newton's method performs better than Apache's standard method, but gives inconsistent results due to different response times. Fuzzy control is more reliable but converges slowly. Heuristics work well in our prototype system, but are difficult to generalize because they require knowledge of bottlenecks and the ability to measure resource usage [2].

Apache is typically structured as a collection of workers that process HTTP requests. Our study uses version 1.3.19 where workers are processes, but we believe the basic idea can be broadly applied.

The request flow in Apache is shown in Figure 2. The request enters the TCP accept queue, where it waits for a worker. Workers process the request until it is completed, after which they accept a new request. The number of worker processes is limited by the MaxClients parameter [5].

Many conclusions in this article are based on the results of experiments. All experiments were performed on a Pentium III 600 MHz server with 256 MB RAM and Apache 1.3.19 server software with Linux 2.4.7 on the same machine, connected to a 100 Mbps LAN; used a synthetic workload generator running on the same machine. File size distribution is the same as in Webstone 2.5; static and dynamic loads were used. Requests to dynamic pages were handled by CGI scripts in Webstone 2.5. More details are needed to explain how the queries are generated.

Our workload model is presented in WAGON [9].

More details are needed to explain how to submit a request. Our workload model is based on the WAGON model, which has been proven to be applicable to a wide range of web requests. This model organizes the workload into sessions (which are a series of user interactions). As shown in Figure 2, a session consists of several page requests. The page contains a number of built-in objects whose parameters are determined by the length of the packet. Thus, the load parameters are session arrival rate, session duration

(number of clicks or page requests in a session), packet length (number of objects in a packet), and reasoning time (time between successive clicks). Table 2 summarizes the parameters used in this work and is based on data provided by a public website based on a synthetic blog created using the WAGON model and using httperf to make HTTP/1.1 requests.[10, 11].

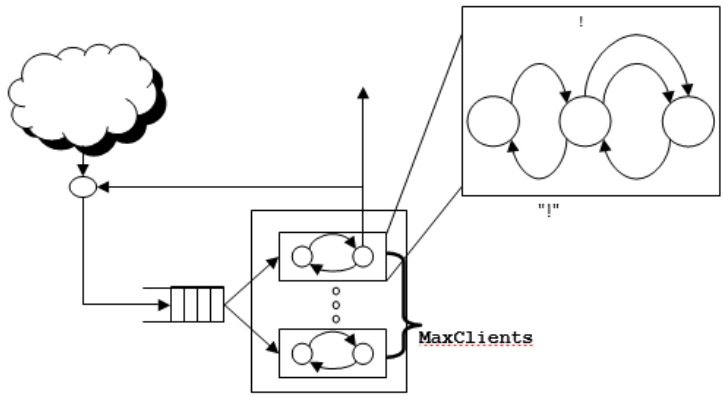

Fig. 2. Apache architecture and session flow

Table 2

Workload parameters

Parameter Nome	Distribution	Parameters
Session Rate	Exponential	Mean = 0.1
Session Length	Log Normal	Mean = 8, Ơ = 3
Burst Length	Gaussian	Mean = 7, Ơ = 3
Think Time (s)	Log Normal	Mean = 30, Ơ = 30

The metric we decided to minimize is server-side response time. Measuring response time on the client side is the most meaningful metric for users, but this information is usually not available in real time on the web server. Also, while client-side response times can be approximated from server-side measurements and TCP models, server-side response times can only be verified [12].

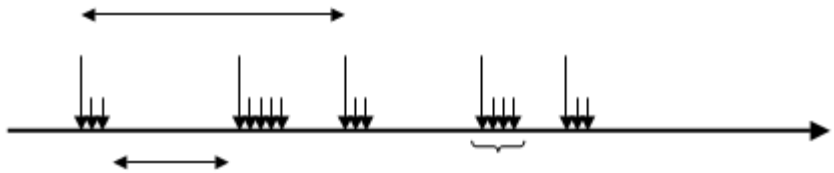

Fig. 3. Elements of the WAGON workload model
(The solid and dashed lines indicate two sessions. The long arrow indicates the HTML text of the Web page, while the short arrow indicates the request to the Web page object)

This article looks at server-side response time, namely page response time (RT). This metric needs to be evaluated because page delivery may involve multiple requests. The following formula is used for this:

$$RT = AQT + BL \times ST \quad (1)$$

Accept Queue Time (AQT) comes from the Linux kernel and adds a tool to measure the average waiting time of connections entering the accept queue over a period of time. Service Time (ST) is measured by measuring the first and last steps of Apache's request processing cycle (that is, when the request is received and when the response is sent). The average number of embedded requests per page, also known as the packet length, is denoted BL. It can be calculated as the number of requests handled by all workers divided by the number of connections handled in the TCP host queue, because HTTP/1.1 connection persistence means that an existing TCP connection remains open between successive HTTP requests in a packet, only the first request must establish a TCP connection and enter the TCP host queue. This leads to the above equation.

Figure 6 shows Apache experimental results with different MaxClients settings. The circle shows the average response time, and the vertical line shows the root mean square deviation. Notice that the circular line appears concave at the top of this curve. Also, this curve shows that MaxClients has more than a 10x impact on response time for this workload.

This concavity can be explained in terms of the Apache architecture: if the value of MaxClients is too low, there will be a large delay due to waiting in the TCP receive queue. In fact, the queue may overflow and requests may be rejected. On the other hand, if the value of MaxClients is too large, resources will be overloaded, which also reduces performance. In extreme cases, exceeding the process limit can lead to internal server errors. The overall result of these factors is that the response time is an upward-concave function of MaxClients. Essentially, a workflow can be thought of as the logical resources required to process a request. However, this requires physical resources such as CPU, memory, and I/O bandwidth.

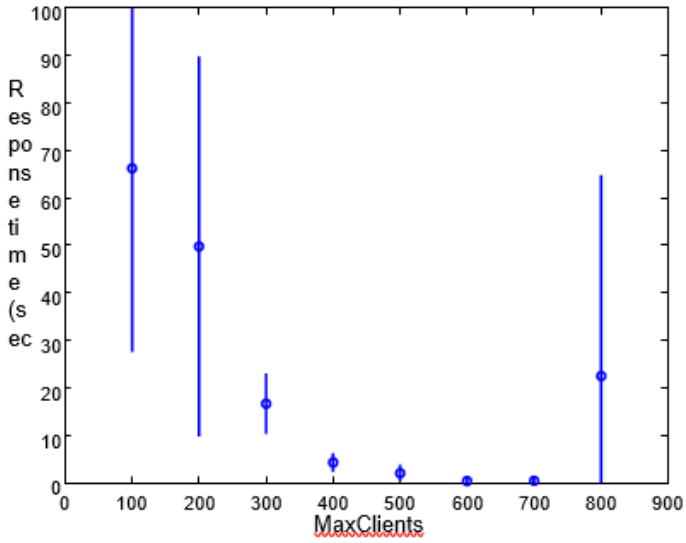

Fig. 4. Impact of MaxClients on Response Time

Since we are interested in online optimization, we need to change MaxClients dynamically. For this, a mechanism similar to Graceful Restart was implemented. This mechanism reportedly allows changing MaxClients without stopping Apache. This

requires an agent in the Apache system. This agent also provides tools such as CPU usage, memory usage, and server-side response time.

Although this article focuses on MaxClients, we believe that the approach used is more widely applicable. For example, we're currently exploring KeepAliveTimeOut, another Apache parameter that specifies the amount of time a session can remain inactive when using a persistent connection. Other systems have configuration options similar to MaxClients, such as the number of servlet threads or EJB threads on the application server [13].

Online optimization describes how the response time can be minimized by dynamically adapting MaxClients, assuming that the response time is concave to MaxClients. Several methods are considered, including Newton's method, fuzzy control, and heuristic methods. These methods are compared in terms of minimizing the response time and speed of convergence to a fixed value. The first is desirable from the point of view of improving the quality of service. The second is important for adaptation to load changes. Both of the above approaches involve feedback. Therefore, MaxClients is adjusted based on the observed impact on response time or other metrics. Alternatively, you can use the direct feedback method, when the optimal value of MaxClients is calculated based on the analytical model. This direct feedback method is attractive because it avoids problems related to stability and speed of convergence. However, this method requires an analytical model with input data that a) tracks the measured response time and b) can be easily estimated. The model developed satisfies point (a) but not (b). For example, the service speed μm depends on the number of servers, i.e. μm = f(m), but we do not know this function. It is possible to develop a model that satisfies (a) and (b), but in the absence of such a model we turn to a feedback approach [16, 17].

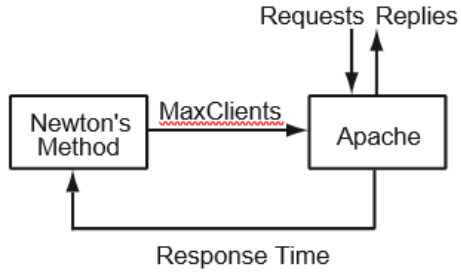

Fig. 5. Apache online optimization architecture using Newton's method to dynamically adjust MaxClients based on response time measurements

Figure 5 shows an architecture where Newton's method is used to optimize when browsing Apache online using dynamically adapting MaxClients [14]. Newton's method is a widely used optimization technique that uses the slope of minimizable functions (eg, Figure 4) to estimate the value of MaxClients that minimizes the response time. For example, if y is response time and x is MaxClients, you can use the approximation y = f(x) ≈ a(x-x*)2+b. Newton's method is represented by the next level.

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k) \quad (2)$$

where xk is the value of x at a discrete moment in time k. Equation (3) starts with the initial value x0 at k = 0.

Its slope (the second partial derivative f(x)) is calculated at the point xk; its negative value indicates a steeper descent direction). The value (of the second partial derivative

of f(x)) indicates the size of the update step; introducing second-order partial derivatives eliminates the local linear search and can lead to faster convergence. However, this algorithm is more sensitive to measurement noise.

Fuzzy control is another approach to online optimization. This study examines exactly this approach. Figure 6 shows an architecture that uses fuzzy control to optimize when browsing Apache online by dynamically adapting MaxClients. The fuzzy controller uses changes in MaxClients and view time to dynamically optimize MaxClients.

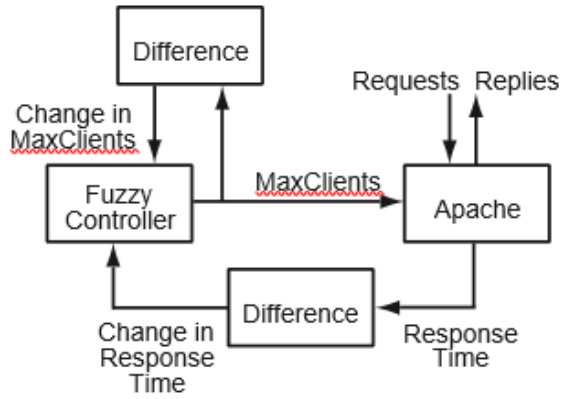

Fig. 6. Apache online optimization architecture using Fuzzy Control to dynamically adjust MaxClients based on changes in MaxClients and response time

The behavior of the fuzzy controller is governed by a set of IF-THEN rules. For example, «IF the change in MaxClients is small and the change in response time is small, THEN the next change in MaxClients is small.» The terms changein-MaxClients and changein-response-time are linguistic variables and neglarge are linguistic values. Linguistic changes are a natural way to deal with the uncertainty that creates probability theory in the context of computer systems. Country linguistic variables in a form that corresponds to the number of variables. Fuzzy control systems provide a method of matching between numerical and linguistic changes (so-called fuzzy fuzzification and unfolding). For more information on fuzzy control, see Unclear control [15, 18].

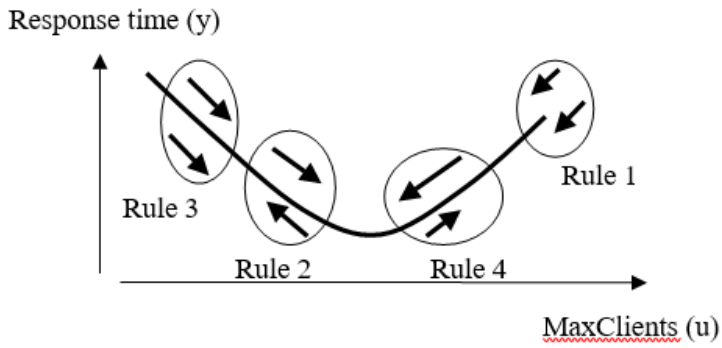

Fig. 7. Illustration of fuzzy rules

Table 3

Unclear rules

Rule	IF	THEN
	change in AND change in Max Clients AND Response Time	change in next Max Cleints
1	neglarge AND neglarge	neglarge
2	neglarge AND poslarge	poslarge
3	poslarge AND neglarge	poslarge
4	poslarge AND poslarge	poslarge

The considered optimization problem can be easily expressed in the form of fuzzy rules. The rules in Table 3 are organized as follows (also shown in Figure 7, where the dashed arrow indicates the precondition IF and the solid arrow indicates the subsequent THEN part): the IF part shifts the position on the lookup time curve for the optimal value of MaxClients. For example, Rule 4 considers the case where MaxClients increases, which increases the browsing time. This means that you are to the right of the optimal MaxClients value; the THEN part shows how you change MaxClients, where neglarge is the decrease and poslarge is the increase. Rules 1 and 3 indicate the position where the response time decreases as a result of the last MaxClients change in the right direction. In contrast, rules 2 and 4 deal with situations where the response time has increased as a result of the last operation, i.e. a «wrong operation»: the number of MaxClients changes varies with the rate of convergence and the degree of steady-state oscillation.

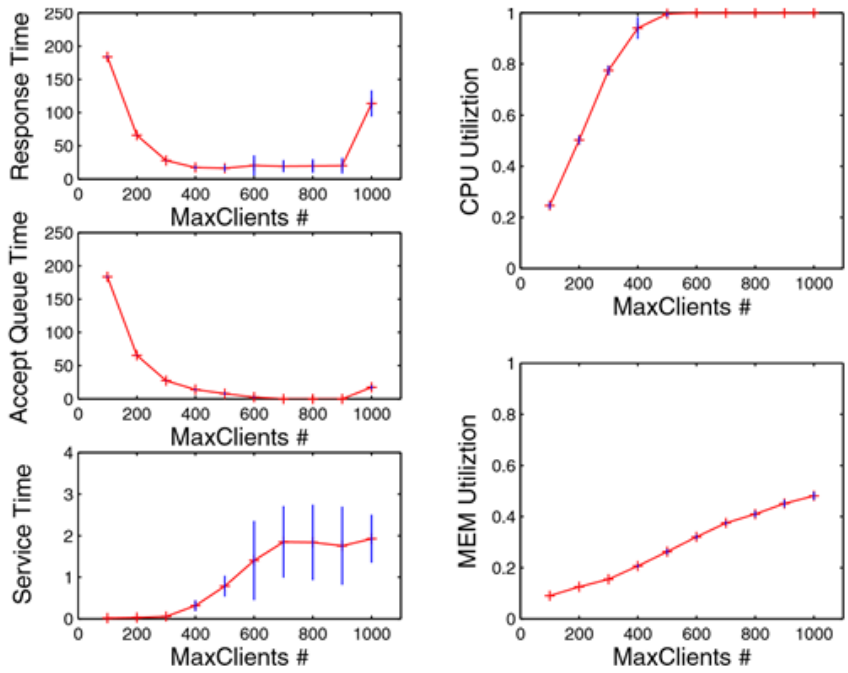

Fig. 8. Measuring Apache for dynamic load

Obviously, if the curve is steeper, then small changes in MaxClients are optimal. For more gradual gradients, it is better to use large changes. Saturation-based optimization heuristics are robust to noise and the specific functions being optimized, and are driven by the pursuit of fast convergence. As MaxClients increase and CPU load is 100%, the response time is minimized. This can be seen from the static and dynamic load measurements in Figures 10 and 11, where the average queue time and service time are measured for different values of MaxClients, and the browsing time is calculated using level (1). CPU and memory usage were also measured to monitor system resource usage. In Fig. 10 hours of viewing decreases when increasing MaxClients from 200 to 480, after which CPU usage is around 100%. In Fig. 11 this saturation state occurs when MaxClients reaches approximately 800 [19, 20].

Our intuition as to why this happens is this: MaxClients is looking for a set of logical resources called Apache workers. These logical resources are divided into physical resources such as CPU, memory, and I/O bandwidth. By increasing MaxClients until the physical resources are saturated, more logical resources can run in parallel.

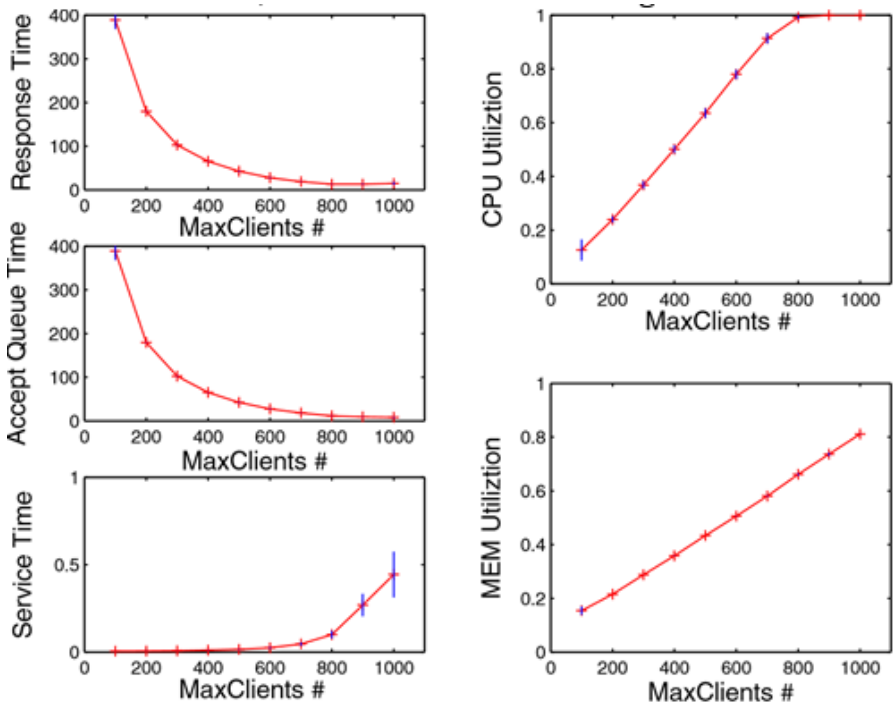

Fig. 9. Measuring Apache for static load

However, as soon as the physical resource is saturated, the further increase of the logical resource does not increase parallelism. Instead, this increase adds overhead (for example, due to switching processes).

The experimental results compare the online optimization methods in terms of the minimum value of the achieved response time, convergence speed and stability. Figure 11 compares the performance of Newton's method with Apache's default scheme. The figure contains three subfigures, each of which has two plots. In each figure, the upper graph shows the trajectory of MaxClients and the lower graph shows the corresponding

response time. Note that Newton's method does improve the response time compared to the standard Apache scheme. However, due to the variability of the response time, different runs of Newton's method can produce very different results. This is because obtaining the Hessian matrix requires three samples to calculate the second derivative at each step of the algorithm. This increases the convergence time and also makes the algorithm more sensitive to noise in the response time measurement. Unfortunately, the response time is usually quite noisy unless averaged over many samples (something that reduces the speed with which the controller can respond to changes in workload). Because of this sensitivity to noise, we do not consider Newton's method in other comparisons.

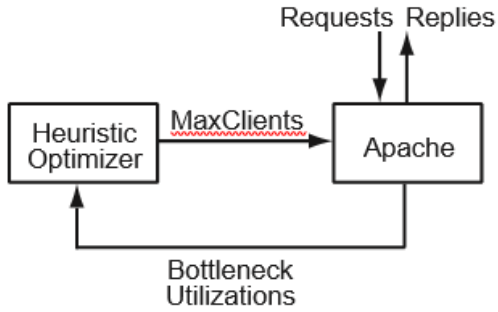

Fig. 10. Apache Online Optimization Architecture Using Saturation-Based Heuristics to Dynamically Adjust MaxClients and Changes in Bottleneck Usage

Next, we compare the typical Apache scheme with fuzzy control and the heuristic method presented earlier. Figure 13 shows the results for dynamic loading. (The results are structured similarly to Figure 11, 12, 13)

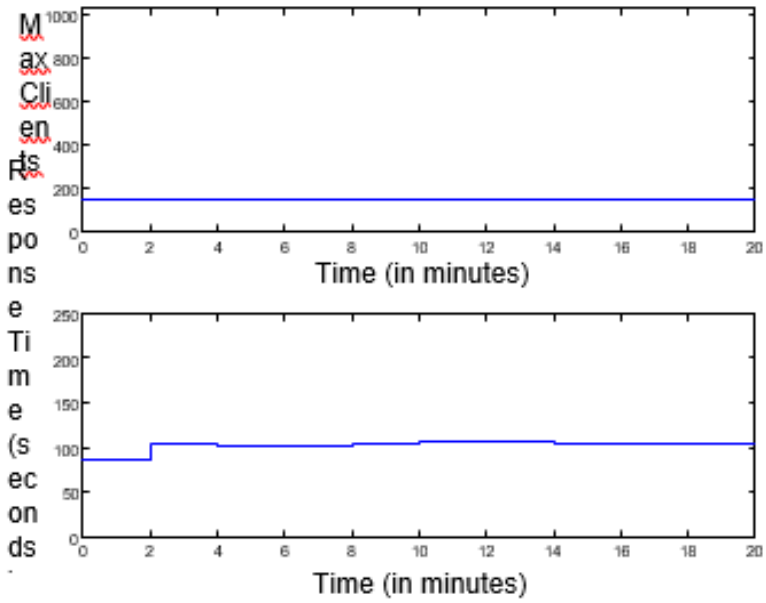

Fig. 11. A typical Apache control scheme

We see that the heuristic converges to its MaxClients value after 2 minutes. For fuzzy control, convergence takes about 14 minutes. On the other hand, fuzzy control achieves a shorter response time. Figure 14 shows the results for static loading. Again, heuristics converge faster than fuzzy control.

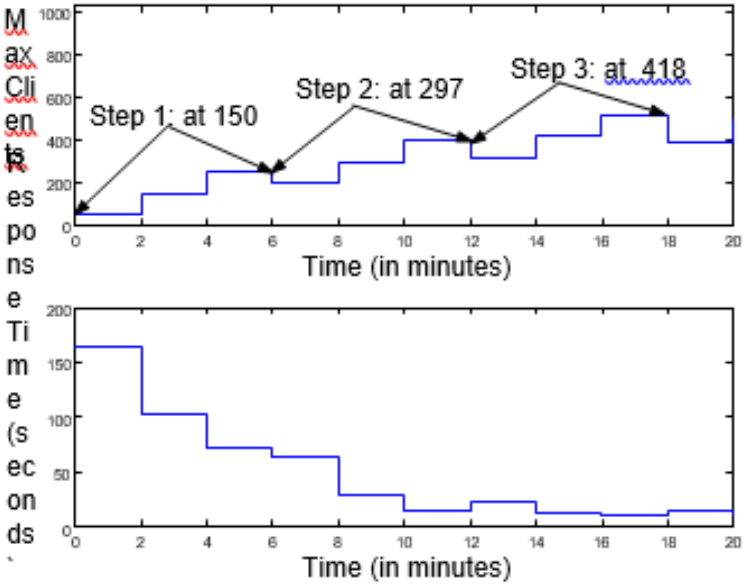

Fig. 12. First launch

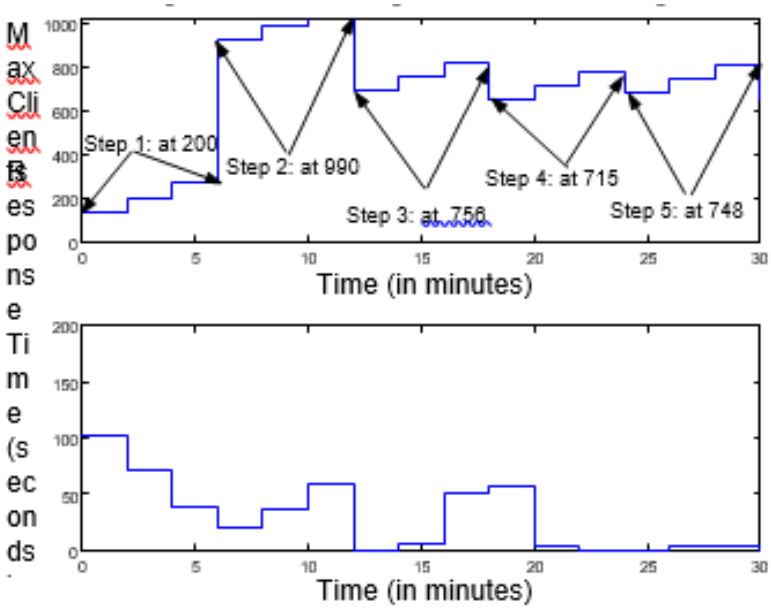

Fig. 13. Second launch

Comparison of the standard Apache method and the Newton method under dynamic loading; Newton's method certainly achieves a shorter response time, but its behavior is inconsistent due to response time variability. Here, however, the steady-state response time achieved by the heuristic method is almost equal to the response time achieved by the fuzzy control.

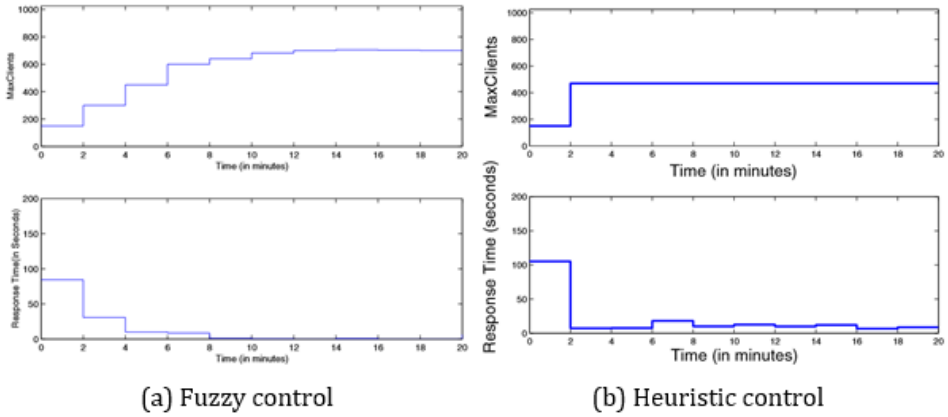

Fig. 14. Comparison of performance of online optimization schemes under dynamic load

Table 4 shows a qualitative comparison of the four formed systems. The Apache system handles minimization view points for response time mainly because it was not designed for this purpose. The Newtonian method is better in this respect, but converges completely and has poor immunity.

Table 4

Qualitative comparisons of methods

	Optimization	**Speed**	**Robustness**
Default Apache	Poor	Fast	Good
Newton's Method	Fair	Slow	Poor
Fuzzy Control	Good	Slow	Good
Heuristic	Good	Fast	Fair

Fuzzy control is very stable, after which it is slightly assumed, but coincides completely. Our heuristics provide good optimization and fast convergence, but resource overload assumptions are not always true.

Conclusions. This article explores an approach to web optimization of Apache web server configuration settings, focusing on techniques that are less invasive and can be applied to a wide range of settings and systems. We focus on the MaxClients parameter, which controls the maximum number of clients. First, we show that MaxClients has an ascending effect on response time and that descent methods can be used to determine the optimal value of MaxClients. This has been demonstrated through measurements and analytical modeling. The basic intuition is that MaxClients controls the trade-off between TCP receive queue delays and delays due to contention for operating system resources. We investigated two optimizers using the hill-climbing method (one based on Newton's method and the other based on fuzzy control); the third method is a heuristic

that exploits the relationship between loading bottlenecks and minimizing response time. In both cases, web optimization showed a response time reduction of more than 10 times compared to using static default values. The trade-offs between online systems are as follows. Newton's method is well known, but it does not provide consistent results with highly variable data such as response time. Fuzzy control is more reliable but converges slowly. The heuristic method works well in our prototype system, but may be difficult to generalize because it requires knowledge of bottlenecks and the ability to measure resource usage. Future challenges include the following. First, optimization of several parameters at the same time. This may include other parameters that can be configured dynamically in Apache, such as KeepAliveTimeOut (which determines how long a TCP connection must remain alive for a client before it is terminated). Second, although we studied performance tuning of Apache web servers, there are more complex systems, such as database servers and application servers, where online optimization can have a more dramatic effect on end-user response times.

BIBLIOGRAPHY:

1. Y. Diao, J. L. Hellerstein, and S. Parekh, "Optimizing quality of service using fuzzy control," in Proceedings of Distributed Systems Operations and Management, 2012.

2. Apache Software Foundation. http://www.apache.org.

3. Y. Diao, J. L. Hellerstein, and S. Parekh, "A business-oriented approach to the design of feedback loops for performance management," in Proceedings of Distributed Systems Operations and Management, 2011.

4. C. Lu, T. Abdelzaher, J. Stankovic, and S. Son, "A feedback control approach forguaranteeing relative delays in web servers," in Proceedings of the IEEE Real-Time Technology and Applications Symposium, 2011.

5. Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury, "Using MIMOfeedback control to enforce policies for interrelated metrics with application to the Apache web server," in Proceedings of Network Operations and Management, 2012.

6. L. Sha, X. Liu, Y. Lu, and T. Abdelzaher, "Queuing model based network serverperformance control," in Proceedings of the IEEE Real-Time Systems Symposium, 2012.

7. D. Menasce, V. Almeida, R. Fonsece, and M. Mendes, "Busines oriented resourcemanagement policies for e-commerce servers," Performance Evaluation, 2010, (42), 223–239.

8. Z. Liu, M. S. Squillante, and J. L. Wolf, "On maximizing service-level-agreementprofits," in Proceedings of the ACM Conference on Electronic Commerce (EC'11), 2011.

9. Mindcraft, "Webstone 2.5 web server benchmark," 2008. http://www.mindcraft.com/ webstone/.

10. Z. Liu, N. Niclausse, C. Jalpa-Villanueva, and S. Barbier, "Traffic model and performance evaluation of web servers," Tech. Rep. INRIA, 2009, 38-40

11. D. Mosberger and T. Jin, "httperf: A tool for measuring web server performance,"in First Workshop on Internet Server Performance (WISP 2008), ACM, 2008, 59-67.

12. D. P. Olshefski, J. Nieh, and D. Agrawal, "Inferring client response time at the webserver," in Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, 2012.

13. S. S. Lavenberg, ed., Computer performance modeling handbook. Orlando, FL: Academic Press, INC, 2013.

14. L. Perssini, The Mathematics of Nonlinear Programming. Springer-Verlag, 2008.

15. K. M. Passino and S. Yurkovich, Fuzzy Control. Menlo Park, CA: Addison Wesley Longman, 2008.

16. Зайцев Є.О. Smart засоби визначення аварійних станів у розподільних електричних мережах міст. Таврійський науковий вісник. Серія: Технічні науки, 2022. (5).
17. Цвик О.С. Аналіз і особливості програмного забезпечення для контролю трафіку. Вісник Хмельницького національного університету. Серія: Технічні науки, 2023. (1).
18. Новіченко Є.О. Актуальні засади створення алгоритмів обробки інформації для логістичних центрів. Таврійський науковий вісник. Серія: Технічні науки, 2023. (1).
19 Твердохліб А.О., Коротін Д.С. Ефективність функціонування комп'ютерних систем при використанні технології блокчейн і баз данних. Таврійський науковий вісник. Серія: Технічні науки, 2022. (6).
20. Lei Song (2008) Biswanath Mukherjee. On the Study of Multiple Backups and Primary-Backup Link Sharing for Dynamic Service Provisioning in Survivable WDM Mesh Networks / IEEE Journal on selected areas in Telecommunication. 2008, (26), (6), 84-91.

REFERENCES:

1. Y. Diao, J. L. (2012). Hellerstein, and S. Parekh, "Optimizing quality of service using fuzzy control," in Proceedings of Distributed Systems Operations and Management,
2. Apache Software Foundation. http://www.apache.org.
3. Y. Diao, J. L. Hellerstein, and S. Parekh. (2011). "A business-oriented approach to the design of feedback loops for performance management," in Proceedings of Distributed Systems Operations and Management.
4. C. Lu, T. Abdelzaher, J. Stankovic, and S. Son. (2011). "A feedback control approach forguaranteeing relative delays in web servers," in Proceedings of the IEEE Real-Time Technology and Applications Symposium.
5. Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury. (2012). "Using MIMOfeedback control to enforce policies for interrelated metrics with application to the Apache web server," in Proceedings of Network Operations and Management.
6. L. Sha, X. Liu, Y. Lu, and T. Abdelzaher. (2012). "Queuing model based network serverperformance control," in Proceedings of the IEEE Real-Time Systems Symposium.
7. D. Menasce, V. Almeida, R. Fonsece, and M. Mendes. (2010). "Busines oriented resourcemanagement policies for e-commerce servers," Performance Evaluation, (42), 223–239.
8. Z. Liu, M. S. Squillante, and J. L. Wolf, (2011). "On maximizing service-level-agreementprofits," in Proceedings of the ACM Conference on Electronic Commerce (EC'11).
9. A. Mindcraft. (2008). "Webstone 2.5 web server benchmark" http://www.mindcraft.com/ webstone/.
10. Z. Liu, N. Niclausse, C. Jalpa-Villanueva, and S. Barbier. (2009). "Traffic model and performance evaluation of web servers," Tech. Rep. 3840, INRIA, Dec.
11. D. Mosberger and T. Jin. (2008). "httperf: A tool for measuring web server performance,"in First Workshop on Internet Server Performance (WISP 2008), ACM, 59-67.
12. D. P. Olshefski, J. Nieh, and D. Agrawal. (2012). "Inferring client response time at the webserver," in Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems.
13. S. S. Lavenberg, ed. (2013). Computer performance modeling handbook. Orlando, FL: Academic Press, INC.
14. L. Perssini. (2008). The Mathematics of Nonlinear Programming. Springer-Verlag.

15. K. M. Passino and S. Yurkovich. (2008). Fuzzy Control. Menlo Park, CA: Addison Wesley Longman.

16. Zaitsev Ye.O. (2022). Smart zasoby vyznachennia avariinykh staniv u rozpodilnykh elektrychnykh merezhakh mist. Tavriiskyi naukovyi visnyk. Seriia: Tekhnichni nauky, (5).

17. Tsvyk O.S. (2023). Analiz i osoblyvosti prohramnoho zabezpechennia dlia kontroliu trafiku. Visnyk Khmelnytskoho natsionalnoho universytetu. Ceriia: Tekhnichni nauky, (1).

18. Novichenko Ye.O. (2023). Aktualni zasady stvorennia alhorytmiv obrobky informatsii dlia lohistychnykh tsentriv. Tavriiskyi naukovyi visnyk. Seriia: Tekhnichni nauky, (1).

19. Tverdokhlib A.O., Korotin D.S. Efektyvnist funktsionuvannia kompiuternykh system pry vykorystanni tekhnolohii blokchein i baz dannykh. Tavriiskyi naukovyi visnyk. Seriia: Tekhnichni nauky, 2022, (6)

20. Lei Song (2008) Biswanath Mukherjee. On the Study of Multiple Backups and Primary-Backup Link Sharing for Dynamic Service Provisioning in Survivable WDM Mesh Networks / IEEE Journal on selected areas in Telecommunication. Vol. 26, No 6. pp. 84–91.