UDC 004.056.55 (519.8) https://doi.org/10.20535/2786-8729.7.2025.338564

APPROACH TO HYBRID LOAD MANAGEMENT IN FAT-TREE WEB CLUSTERS

Kostiantyn Radchenko*
https://orcid.org/0000-0002-1282-6307

Artem Chernenkyi
https://orcid.org/0009-0005-6925-4361

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukrainee

*Corresponding author: radchenko.kostiantyn@lll.kpi.ua

The paper presents an approach to hybrid load management in a web cluster that is capable of providing adaptive request balancing based on load prediction and resilience to random web server failures. The proposed architecture is built upon the Fat-Tree topology, which ensures high scalability, structural redundancy, and efficient routing within the cluster network. The developed system performs load forecasting using moving average methods and Erlang-based queueing models, enabling the estimation of overload probabilities and proactive redistribution of computational resources. Four representative simulation scenarios were analyzed: baseline load, peak load, dynamic traffic variations, and random server failures. The obtained results demonstrate enhanced system reliability, reduced average response time, and more balanced utilization of cluster resources. In the context of rapidly growing web services and user traffic volumes, the issue of maintaining high reliability and efficiency of clustered infrastructures becomes increasingly significant. Even with robust topologies such as Fat-Tree, irregular traffic patterns and sudden surges in client requests can cause local overloads and performance degradation. Random node failures further complicate cluster management, necessitating the use of adaptive and predictive control mechanisms. The proposed model integrates Fat-Tree network simulation with statistical forecasting algorithms, forming the basis for proactive load management. This integration allows for minimizing service degradation risks, dynamically responding to workload changes, and maintaining stable operation of web infrastructures under partial node failures. The architecture shows strong potential for real-time implementation in large-scale distributed web systems. It can be further enhanced by incorporating machine learning or wavelet-based forecasting methods to improve the accuracy of load estimation and system adaptability.

Keywords: load forecasting, web cluster, Fat-Tree topology, fault tolerance, traffic balancing, stochastic failures.

1. Introduction

The rapid expansion of online services and user-generated traffic has created significant challenges for maintaining the stability, responsiveness, and reliability of web infrastructures. As digital ecosystems become increasingly distributed, ensuring efficient load distribution and fault tolerance across large-scale server clusters has become a key area of research in network systems engineering.

Modern web clusters rely on intelligent load balancing mechanisms and resilient topologies to maintain service availability under dynamic and unpredictable workloads. The Fat-Tree topology has emerged as one of the most efficient architectural solutions due to its hierarchical structure, high throughput, and redundancy. However, even with such robust topology, the performance of cluster systems can degrade under traffic bursts, uneven request distribution, or random node failures.

Traditional reactive balancing approaches respond to overloads only after they occur, which can lead to latency spikes and temporary service degradation. This motivates the development of predictive control mechanisms capable of anticipating load variations before they cause instability. Analytical models based on Erlang queuing theory, combined with statistical and machine learning forecasting techniques such as ARIMA (AutoRegressive Integrated Moving Average) or LSTM (Long Short-Term Memory), offer new opportunities for intelligent traffic prediction and adaptive load management.

Therefore, the relevance of this research lies in designing a hybrid predictive load balancing architecture that integrates the Fat-Tree topology with proactive traffic forecasting and fault tolerance. This approach ensures stable performance and optimal resource utilization under variable load and failure conditions.

2. Literature review and problem statement

The efficiency of cluster systems largely depends on the chosen network topology, load management strategy, and the ability to recover from failures.

In this context, a significant body of research has focused on the analysis of Fat-Tree topology, server load forecasting, and the development of fault-tolerant computing infrastructures.

The Fat-Tree topology, first formalized in [1], has become widely adopted in the construction of data centers and cluster systems. This is due to its property of providing equal bandwidth at all levels and supporting multipath routing. Studies [2, 3] demonstrate that Fat-Tree enables effective infrastructure scaling without reducing overall bandwidth as the number of nodes increases. In [4], the ability of this topology to reduce network traffic collisions through the use of ECMP (Equal-Cost Multi-Path) routing is emphasized.

However, most of the mentioned studies consider static load scenarios without accounting for dynamic changes or unpredictable failures.

Studies [5, 6] highlight that web application traffic is highly variable and unevenly distributed across nodes. To ensure high-quality user service, load forecasting and adaptive balancing are required. Algorithms for dynamic request distribution that consider response time, current load, and historical data were proposed in [7]. However, most of them rely on the system's instantaneous state and do not provide proactive planning.

According to [8], modern distributed systems must be capable not only of automatically detecting failures but also of adapting the topology or redirecting traffic without significant performance loss. Stochastic modeling of failures, in particular using Poisson processes and the Erlang distribution, allows for the realistic simulation of unpredictable server or network component failures [9], providing more accurate testing and recovery planning. The use of the Erlang distribution is appropriate because it effectively represents the time between events in queuing systems and enables probabilistic estimates of load, response time, and failure probability [10].

The application of the Erlang distribution in modern cluster and cloud systems is becoming increasingly relevant. Studies [11, 12] have shown that modeling inter-request times or failure intervals using Erlang allows for predicting peak loads, assessing “flash crowd” probabilities, and optimizing resource balancing. In particular, these approaches are applied to plan backup servers, estimate recovery time, and forecast the likelihood of overload on individual nodes in real time. This combination of stochastic modeling and predictive algorithms enhances the reliability of cluster systems and provides more precise resource management.

The development of recovery systems, particularly through replication or backup communication channels, is covered in [13], but such approaches are rarely integrated with predictive methods. Over the past decade, considerable attention has been given to the use of machine learning methods for load forecasting in IT systems. In [14, 15], the effectiveness of ARIMA, LSTM, and Facebook Prophet models for predicting CPU load, RPS (Requests Per Second), and other parameters is demonstrated. In particular, [16] shows that LSTM models exhibit higher accuracy over short time horizons, which is especially important for real-time cluster resource management. Despite this, a limited number of studies combine load forecasting with modeling of specific network topologies.

In [17] investigates the use of DWT (Discrete Wavelet Transformations) for forecasting web server load in general-purpose computer networks. The study highlights the advantage of wavelet-based decomposition in capturing both local and global patterns of traffic variation, which is critical for real-time load balancing. The approach enables more accurate prediction by filtering noise and isolating significant trends within web request time series.

Building on this, [18] introduces a conceptual model aimed at improving the efficiency of web server load forecasting. The paper emphasizes a layered structure that integrates data preprocessing, feature extraction, and predictive analytics, offering a systematic framework for implementing intelligent decision-making in distributed environments. Furthermore, [19] explores the peculiarities of web traffic forecasting, noting the irregular and bursty nature of web loads. This research reinforces the need for hybrid approaches that can adapt to the stochastic and non-stationary characteristics of traffic, particularly in clustered infrastructures. Collectively, these studies contribute to the foundation for developing resilient and adaptive web cluster systems with embedded load prediction mechanisms.

The analysis of previous works indicates significant scientific interest in scalable and reliable network architectures. However, there remains insufficient integration of realistic infrastructure modeling with Fat-Tree topology, consideration of dynamic load variations, stochastic failure modeling, and predictive algorithms for resource management within web clusters.

Most existing studies focus either on improving routing efficiency or on reactive load balancing mechanisms, neglecting the predictive aspect of control. Moreover, classical statistical models often fail to capture the nonlinear and non-stationary behavior of web traffic, which is characterized by abrupt fluctuations and short-term periodicity. The lack of models capable of multiscale traffic analysis and proactive load control leads to suboptimal use of computational resources and reduced system reliability.

Therefore, the unresolved scientific problem lies in the absence of an integrated approach that combines scalable cluster topology, stochastic fault modeling, and hybrid predictive load control. Such an approach is needed to ensure reliable and adaptive operation of web clusters under dynamic traffic conditions.

3. The aim and objectives of the study

The purpose of this study is to develop and investigate a hybrid fault-tolerant web cluster model with predictive load balancing, based on Fat-Tree topology, analytical traffic models, and statistical forecasting methods.

To achieve the goal, the following tasks were set:

– to develop a Fat-Tree topology model for web cluster deployment, assessing its scalability, redundancy, and fault-tolerance capabilities;

– to design and implement a hybrid predictive load balancing mechanism that combines analytical (Erlang-based) traffic modeling and statistical forecasting (neural-based prediction) for adaptive request distribution;

– to perform simulation experiments based on the proposed cluster model with the implemented predictive load balancing mechanism to evaluate the impact on response time, load variance, and request distribution.;

– to simulate random node failures within the proposed cluster model to evaluate its reliability, fault-tolerance, and self-recovery capabilities under partial degradation.

4. The study materials and methods of load forecasting

4.1. The object, subject and hypothesis of the study

The object of this research is the operation of web server clusters organized according to the Fat-Tree topology under conditions of dynamically varying load and partial failures of individual nodes. The subject of the study includes the processes of intelligent load management and fault tolerance in distributed computing systems, based on predictive modeling and analytical methods for system performance evaluation.

The hypothesis of this research assumes that the integration of Fat-Tree network modeling with analytical and statistical load forecasting techniques can significantly improve the reliability, adaptability, and performance of large-scale web clusters. This integration enables proactive load balancing and self-recovery under fault conditions.

Such integration provides:

– enhanced efficiency of request distribution within the cluster;

– mitigation of the negative effects caused by node failures;

– implementation of proactive load management mechanisms in scalable web services.

This research focuses on the structural characteristics of the Fat-Tree topology, mathematical formalization of load in web clusters, probabilistic modeling of individual node failures, and theoretical approaches to server load forecasting.

4.2. The Fat-Tree topology model for web cluster deployment

The Fat-Tree architecture represents a modified form of the classical tree-based network topology, designed to provide uniform bandwidth distribution across all hierarchical levels. Its fundamental principle is the gradual increase in the number of connections toward the root layer, which ensures effective load balancing and scalability. Due to these properties, the Fat-Tree topology has found widespread application in modern data centers and high-performance computing systems. It serves as a robust foundation for research on predictive load management and fault-tolerant operation.

Formally, for a Fat-Tree with parameter k, where k is an even number, the total number of servers is $k^3/4$, and each of the k subnets (pods) contains $k/2$ edge and aggregation switches. The core switches provide interconnections between all subnets, forming a complete hierarchical structure with multiple alternative routes.

The characteristic properties of this topology include scalability, support for multipath routing, physical symmetry of the network, and high resilience to individual node or link failures. This makes Fat-Tree suitable for environments with intensive and variable network load.

Load in web clusters is typically described using metrics such as RPS, CPU utilization, system response time, and the number of concurrently active connections. The formalization of load allows for a quantitative assessment of each node's state at any given moment.

The mathematical model of load on the i-th node at time t can be represented as a function of several variables:

$$L_i(t) = f(R_i(t), C_i(t), M_i(t)), \tag{1}$$

where $R_i(t)$ is the request intensity, $C_i(t)$ is CPU usage, and $M_i(t)$ is memory usage. The variability of these indicators over time exhibits stochastic and seasonal characteristics, necessitating the use of time series analysis and forecasting methods.

From a reliability standpoint, each node or processor in a cluster system has a non-zero probability of failure. To model such events, both binomial and exponential distributions can be used. In particular, the exponential distribution of time to failure is described by the function

$$P(T > t) = e^{-\lambda t}, \tag{2}$$

where λ is the failure rate.

For the purposes of system resilience analysis, a binary model can be applied, where the state of a node is described by the variable $S_i(t) \in \{0, 1\}$. If $S_i(t) = 1$, the node is functioning properly and processing load; otherwise, it is considered failed, and its load $L'_i(t)$ is assigned a value of zero. This approach allows effective modeling of scenarios involving partial or prolonged component failures in the network.

One of the key directions for optimizing the operation of web clusters is forecasting node load to enable preventive load balancing. This task is formalized as a time series problem, where the predicted load value $\hat{L}_i(t+\tau)$ is determined based on the values at previous time points. The general model can be expressed as:

$$\hat{L}_i(t+\tau) = \mathcal{F}\left(L_i(t), L_i(t-1), \ldots, L_i(t-n)\right), \tag{3}$$

where $\mathcal{F}$ is the forecasting model (which may be regression-based, statistical, or neural).

Among modern approaches, RNN (Recurrent Neural Networks), particularly the LSTM architecture, occupy a special place. Thanks to its memory blocks, LSTM can retain dependencies over long time intervals. This is especially important when the load exhibits complex dynamics, fluctuations, or depends on cyclical events.

To comprehensively evaluate the effectiveness of the proposed model, both precise forecasting metrics and system-level performance indicators will be used. These include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Squared Error (MSE) violation rate, overall throughput, and the load balancing index. Fault tolerance indicators, such as the number of critical failures over a given period, are also considered.

For the study, a virtualized network with a Fat-Tree topology was implemented. All nodes were conditionally divided into servers (computing nodes) and routers. Each server in the model processes incoming requests, with the workload simulated according to selected profiles. The state of each server changes depending on the modeled load and the probability of failure.

4.3. Design and implementation of a hybrid predictive load balancing mechanism

The proposed approach integrates analytical Erlang-based traffic modeling with neural-based statistical forecasting, forming a dual-layer predictive control loop capable of proactively redistributing client requests within the Fat-Tree cluster architecture.

Traditional load balancing methods, such as Round Robin or Least Connections, rely solely on instantaneous system states and are unable to anticipate future load surges or node degradation. To address these limitations, the hybrid mechanism developed in this study employs a two-tier predictive framework:

- analytical layer – models the current traffic intensity and service capacity using Erlang queuing theory;
- forecasting layer – predicts short-term load variations using a neural network trained on historical web traffic metrics.

The integration of these two models provides both stability (from the analytical baseline) and adaptivity (from the neural forecasting module), which together form a self-adjusting decision engine for proactive load management.

The analytical layer estimates the expected utilization and blocking probability for each node in the cluster.

Using an $M/M/c$ queueing model, where λ denotes the average request arrival rate and μ the service rate per server, the offered load A is defined as:

$$A = {}^{\lambda}/_{\mu}. \tag{4}$$

The probability of overload (blocking) for a node with c service channels is computed via the Erlang B formula:

$$B(c, A) = \frac{{}^{A^c}/_{c!}}{\sum_{k=0}^{c} {}^{A^k}/_{k!}}. \tag{5}$$

This expression allows the system to evaluate real-time saturation levels and to estimate the probability that a new request will be queued or rejected. These analytical parameters form the baseline input for the higher-level forecasting layer.

The forecasting layer employs a feed-forward neural network designed to predict the short-term workload on each node. The input variables include recent request arrival rates $\lambda\left(t-n\dots t\right)$, service response times, queue lengths and prior forecast errors. The network output $\hat{L}_i\left(t+1\right)$ represents the predicted load for the next time interval. The model is trained using the MSE criterion.

To simulate realistic load conditions, synthetic request generation was applied, incorporating fluctuations within daily and weekly cycles. The incoming traffic was formed based on a combination of harmonic functions and stochastic oscillations using the formula:

$$L_i\left(t\right) = Asin\left(2\pi ft+\phi\right)+\eta\left(t\right), \tag{6}$$

where A is the load amplitude, f is the frequency, φ is the phase, and $\eta\left(t\right)$ represents the random noise component (white noise or a Gaussian noise function).

The neural component captures nonlinear temporal dependencies and stochastic fluctuations that the Erlang model cannot represent. At each scheduling interval, the hybrid controller computes a weighted decision metric for each node:

$$W_j = \alpha\cdot B_j\left(c,A\right)+\beta\cdot\hat{L}_j\left(t+1\right), \tag{7}$$

where α and β are adaptive coefficients balancing the analytical and predictive contributions.

The load balancer then selects the server S_k with the minimum W_j value:

$$S_k = arg\min_j W_j. \tag{8}$$

This mechanism enables predictive request routing, ensuring that servers likely to become overloaded are pre-emptively relieved before congestion occurs.

Additionally, during the simulation, periods of peak load with high request density were introduced to emulate critical situations such as “flash crowd” events or DoS (Denial of Service) attacks.

As shown in Fig. 1 the models also incorporated random changes in server states, reflecting real hardware failures or overloads. For this, pseudo-random modeling was applied according to the exponential distribution of time between failures. At the moments of failure, the load on the corresponding server was reset, and routing was redistributed through neighboring nodes according to the minimum depth algorithm or ECMP.

For forecasting the load on individual nodes, three fundamentally different approaches were employed. These include the classical autoregressive model ARIMA, a deep learning model based on LSTM networks, and a stochastic approach grounded in queuing theory using the Erlang distribution. The models were trained on time series of past load values within a 60-minute window, and the forecasts were generated for the next 15, 30, and 60 minutes.

The LSTM model was implemented using the `Keras` library with the TensorFlow backend. For training, the MSE loss function, the Adam optimizer, and early stopping (in case of no improvement in metrics) were employed. In parallel, an ARIMA model was built with optimal parameters (p,d,q), selected using the Bayesian Information Criterion.

Due to its flexibility, the Erlang distribution enables probabilistic estimates of system load levels, the likelihood of queue blocking, and the prediction of peak loads. In the context of web servers, this approach allows for the assessment of QoS (Quality of Service) metrics. These metrics include average response time, probability of access denial, and the distribution of waiting times in the queue. The Erlang distribution is characterized by two key parameters. The first is the number of stages k, which determines the shape of the distribution and the degree of variability of intervals between events, and the second is the rate parameter λ, which defines the average arrival rate of requests. Additionally, modeling with the Erlang distribution provides a basis for planning system scaling and optimizing

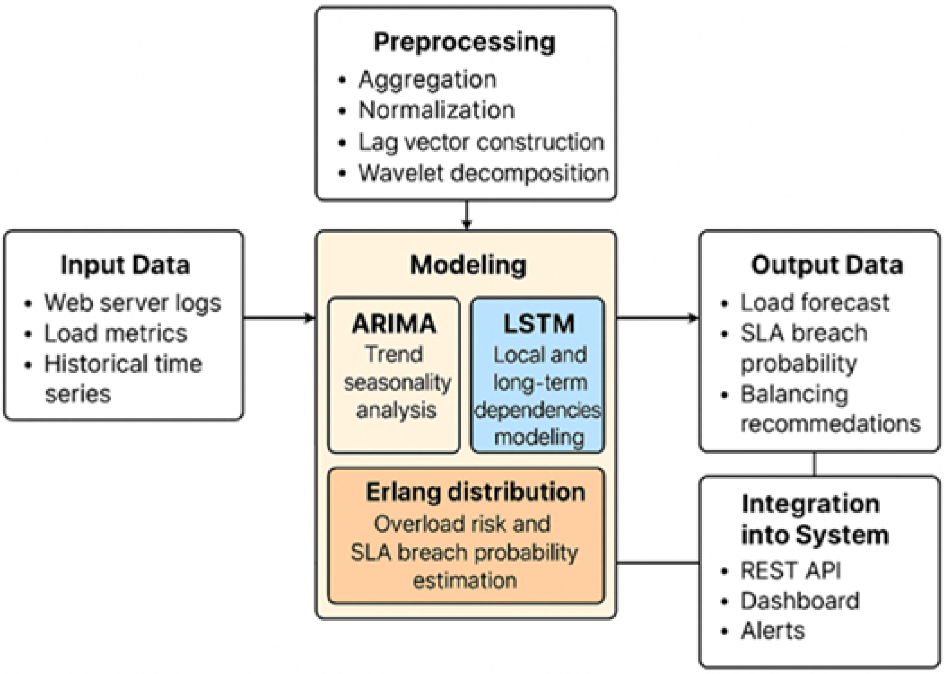

Fig. 1. Web server load model forecasting flowchart

resource allocation. This is critical for maintaining high availability and reliability of computing clusters under variable request intensity.

Graphical representation of the Erlang distribution for different k values at a fixed λ shows that increasing k reduces the probability of extremely short or long intervals between requests. This property is useful for predicting peak loads and managing queues. Similarly, varying λ allows modeling different levels of average request intensity, providing the ability to assess system resource margins and QoS metrics such as average response time and probability of failure. The dependence of the distribution on parameters k is illustrated in Fig. 2.

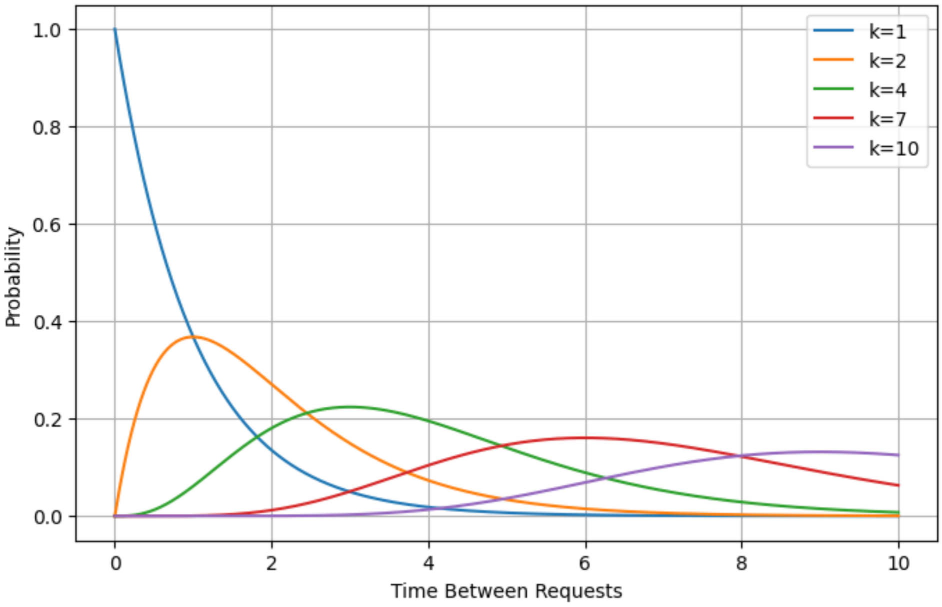

Fig. 2. Erlang distribution for different values of k $(\lambda = 1)$

The forecasting results were applied for predictive load balancing, which reduced the likelihood of overloading critical nodes.

The effectiveness was evaluated according to the following key indicators:

– forecasting accuracy. The predictive performance of the models was measured using the MAE, the RMSE, and the R^2 (coefficient of determination);

– system performance. Operational efficiency was evaluated by monitoring the average request processing latency, the throughput (requests per second), and the number of successfully processed requests;

– load balancing. The uniformity of workload distribution across nodes was quantified using the LIF (Load Imbalance Factor);

– fault tolerance. System robustness was analyzed in terms of the share of unserved requests, the average server downtime, and the response speed of the system under failure conditions.

All indicators were analyzed under conditions of normal operation, peak loads, and large-scale failure scenarios. Early detection of peak loads enables proactive request balancing, prevents overload of individual nodes, and reduces the number of lost requests.

5. Results of hybrid predictive load balancing experiments

5.1. Simulation experiments for evaluating predictive load balancing efficiency

To validate the effectiveness of the proposed hybrid predictive load balancing mechanism, a series of simulation experiments were conducted. These experiments used a synthetic Fat-Tree network model representing a scalable web server cluster.

The Fat-Tree topology was implemented, configured within a Python-based simulation environment using the `NetworkX` library. The chosen topology configuration was a Fat-Tree with parameter $k = 4$, which makes it possible to model a cluster consisting of 16 servers, 20 switches, and a fully functional three-tier network.

The input data for the forecasting models consisted of time series describing the number of requests processed per unit of time (e.g., requests per second) for each server. The data were first normalized to the $[0, 1]$ range for the LSTM model, and stationarity checks (Dickey-Fuller test) were performed to properly configure the ARIMA models.

For training, 80% of the simulation data were used, while the remaining 20% were reserved for testing. The sliding window length for the LSTM model was 60 minutes (720 records with a 5-second simulation interval), and forecasts were built for horizons of 15, 30, and 60 minutes ahead.

ARIMA was modeled separately for each server with individual parameter selection (p, d, q) through a grid search minimizing. The most frequently optimal combinations were $p = 3, d = 1, q = 2$.

The LSTM model was implemented using the `Keras` deep learning framework. Its architecture consisted of a single LSTM layer with 128 hidden units, followed by a Dropout layer with a rate of 0.2 to mitigate overfitting. The output was produced by a Dense layer with a single neuron. The model was trained using the `Adam` optimizer and the MSE as the loss function. Training was carried out for 50 epochs, with an early stopping mechanism employed to prevent overfitting and improve generalization.

Erlang-based model was applied as a stochastic alternative to estimate the probability of incoming requests. For each server, the inter-arrival times were modeled as Erlang(k, λ), where the parameters k (shape) and λ (rate) were determined based on historical data. This approach allowed: estimating the probabilistic intensity of requests at any given time; forecasting peak and low load periods; accounting for the natural variability of request streams in the system.

The model performance was evaluated using three metrics: MAE, RMSE, and R^2. The comparative results for different forecasting horizons are summarized in Table 1.

As shown in Table 1, the LSTM model consistently demonstrates superior accuracy compared to ARIMA, particularly over longer forecasting horizons. This advantage is attributed to its ability to capture complex nonlinear dependencies and variations in load behavior. The Erlang-based model,

Table 1. Forecasting accuracy of ARIMA, LSTM, and Erlang-based models

Forecast Horizon	**Model**	**MAE**	**RMSE**	R^2
15 min	ARIMA	0.048	0.066	0.91
	LSTM	0.031	0.045	0.95
	Erlang	0.036	0.050	0.93
30 min	ARIMA	0.074	0.092	0.85
	LSTM	0.055	0.068	0.90
	Erlang	0.060	0.075	0.88
60 min	ARIMA	0.120	0.152	0.71
	LSTM	0.087	0.110	0.82
	Erlang	0.095	0.120	0.80

in turn, proves effective for short-term load forecasting under high variability and complements both ARIMA and LSTM, especially in assessing the probabilistic characteristics of the system.

Around each server, a load generator with random fluctuations was modeled according to the previously described scenarios. In addition, a mechanism for random server failures was introduced, with parameters defined by an exponential distribution of time between failures, with a mean time of 4 hours.

The following main scenarios were designed and implemented:

– scenario 1: baseline load. Uniformly distributed load with moderate intensity and no failures;

– scenario 2: peak load. Simulation of flash crowd periods with increased request volume;

– scenario 3: predictive load balancing. Utilization of an Erlang-based model for load forecasting with automatic traffic redirection.

Each scenario was tested over 24 hours of simulated time, while statistics on key metrics were collected.

In the experimental *Scenario 1*, a stable baseline load on the web cluster was simulated, corresponding to the normal operating mode without sudden fluctuations in the number of requests. The purpose of this scenario was to determine the baseline performance indicators of the system under the condition of uniform load distribution across all servers. A synthetic request generator with constant intensity was used to produce the load, corresponding to the average traffic level of a typical web service. The inter-arrival times of requests were modeled using an exponential distribution with parameters tuned to reflect realistic user behavioral patterns.

The network structure is illustrated in Fig. 3.

The main metrics recorded during the simulation were:

– average system response time (latency);

– cluster throughput (number of processed requests per unit time);

– resource utilization of each server;

– number of rejected or lost requests.

The results of this scenario served as the baseline reference point for further comparison with other, more demanding or failure-prone scenarios. It was recorded that under baseline load, the average system response time was about 120 milliseconds, and the cluster throughput was 980 requests per second. No failures or rejected requests were observed, indicating stable system operation under normal conditions.

In the *Scenario 2*, peak load typical for periods of increased user activity – such as during advertising campaigns or large online events – was simulated.

For modeling peak load, a synthetic generator was used that increased the request intensity by 2–3 times compared to the stable baseline level during specific time intervals. Peak periods lasted for 10–15% of the total simulation time and were distributed unevenly to imitate random flash crowd effects.

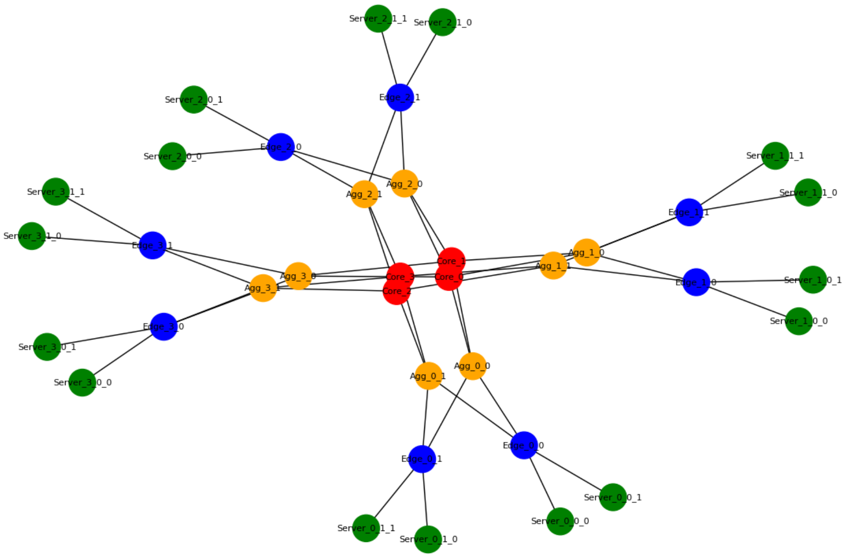

Fig. 3. Fat-Tree topology $(k = 4)$ with uniform server load

The key metrics measured in this scenario included:
– maximum system response time;
– average throughput during peak and non-peak periods;
– number of rejected or delayed requests;
– load level of individual servers in the cluster.

As shown in Fig. 4, the goal of this experiment was to assess the system's ability to cope with sudden increases in request volume. It also aimed to determine the impact of peak load on the performance and stability of the web cluster.

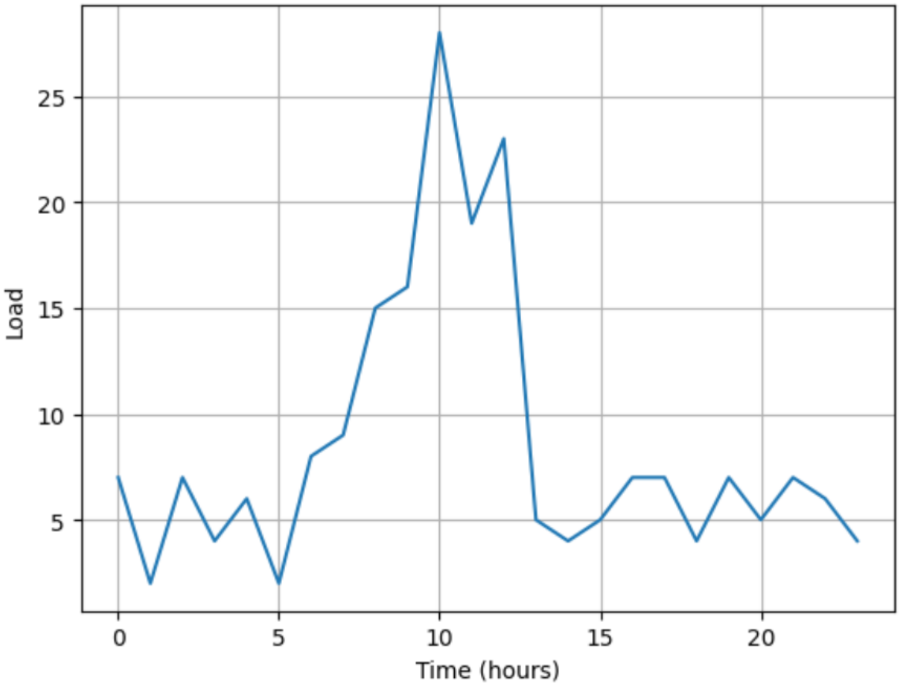

Fig. 4. Time series of web server requests demonstrating a flash crowd event

The results showed that during peak periods, the average latency increased by a factor of 2.2 compared to the baseline scenario, reaching 270 ms, while throughput decreased by 22% compared to normal operation. The increased response delays and reduced throughput indicate the presence of bottlenecks in network or server resources under peak loads.

The main goal of next *Scenario 3* was to evaluate the benefits of predictive load balancing in a Fat-Tree web cluster. Instead of simply distributing requests evenly, the system used forecasted values of server load to optimize request allocation. This approach was intended to prevent overloads on individual servers and to maximize the overall efficiency of the cluster.

The predictive load of the Erlang-based model is shown with a discretization interval of five minutes, demonstrating expected server demand over time in Fig. 5.

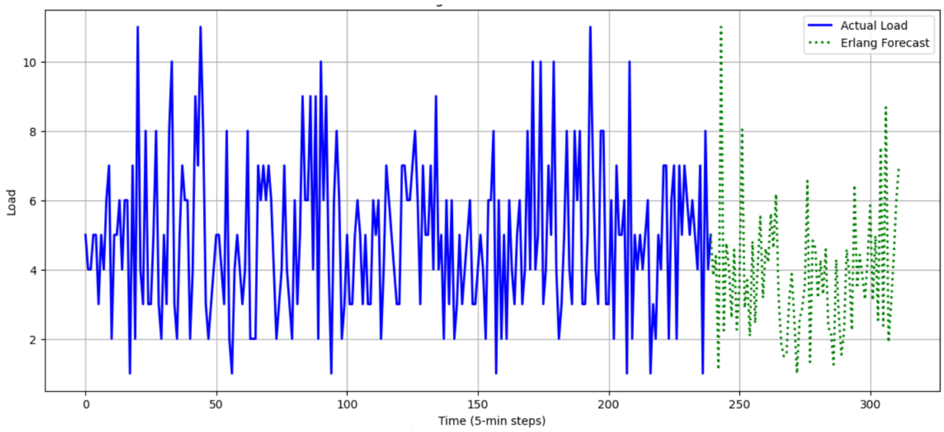

Fig. 5. Predictive load of Erlang-based model with 5-min discretization

Here, the system leveraged historical data to predict the upcoming load for each server, employing models such as moving averages or machine learning approaches. Based on these predictions, new incoming requests were dynamically routed to servers with lower expected utilization. This scenario setup assumed no server failures, allowing the focus to remain entirely on improving load distribution and system responsiveness.

The anticipated results included a more even utilization of cluster resources and a noticeable reduction of peak loads on individual servers. Additionally, an overall improvement in system response time was expected compared to a baseline with uniform request distribution.

In this scenario, an approach was implemented that combines time series–based load forecasting with dynamic request distribution among servers. The use of models makes it possible to detect increasing traffic trends in advance and proactively redirect requests. As shown in Fig. 6 the comparative chart illustrates latency changes in two cases: without forecasting (red curve) and with predictive load balancing enabled (green curve).

In the first case, a gradual increase in latency with sharp peaks can be observed, indicating server overload. In contrast, when applying the predictive model, latency remains relatively stable even during peak load periods. This confirms the effectiveness of the approach in reducing the risk of performance degradation and improving the overall reliability of the system.

5.2. Fault-tolerance analysis through random node failure simulation

Ensuring fault tolerance is a critical aspect of distributed web server infrastructures. This is especially important in large-scale systems employing the Fat-Tree topology, where component failures can lead to service degradation or partial system outages. To evaluate the resilience of the proposed hybrid predictive load balancing mechanism, a set of simulation experiments was conducted involving randomized node failure events under varying load conditions. The objective of

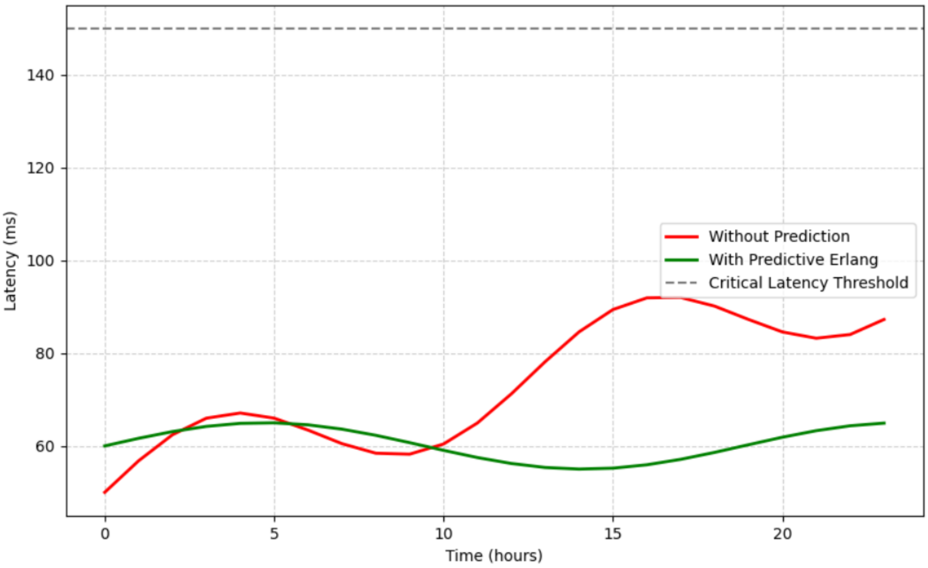

Fig. 6. Predictive load balancing: latency comparison

this stage was to assess the system's ability to maintain operational stability, minimize request loss, and ensure service continuity during partial network degradation.

The fault-tolerance analysis was based on a stochastic failure model, where the probability of node failure was defined as a time-dependent function $p_{fail}(t)$ representing hardware or software degradation in real-world clusters. Failures were simulated using a Poisson process with mean rate λ_{fail}, such that at each simulation interval Δt, the occurrence of a failure event followed:

$$p_{fail}(\Delta t) = 1 - e^{-\lambda_{fail}\Delta t}. \tag{9}$$

Once a node failed, its processing capacity was instantly set to zero, and all ongoing and pending requests were either rerouted to neighboring nodes or dropped if no available capacity remained. Recovery events followed an exponential distribution with MTTR (Mean Time to Repair) defined by system reliability parameters.

The following classes of failures were modeled:

- single-node failures – isolated faults occurring at random intervals;
- clustered failures – simultaneous faults within one switch domain (edge layer);
- transient failures – temporary outages with automatic recovery after a short interval.

The simulation involved randomly selecting 20% of the nodes for potential failure throughout the observation period, thereby ensuring realistic fault dynamics and varying recovery sequences.

A particular focus was placed on assessing the efficiency of traffic redirection and load balancing mechanisms when forecasted server utilization is taken into account.

In this setup, server failures were modeled with a predefined probability. Once a server failed, it temporarily stopped handling requests until it recovered after a fixed recovery time. During this downtime, all traffic originally directed to the failed server was automatically redistributed among the remaining active servers. The redistribution strategy relied on short-term load forecasting in order to avoid overloading the remaining nodes.

The simulation procedure included several key steps. First, a forecasting model such as an Erlang-based model was applied to estimate near-future server load. At each time step, the operational state of every server was determined randomly as either *active* or *failed*, based on the scenario parameters for failure probability and recovery duration. Incoming requests were evenly generated, but only distributed among the servers that remained active, with adjustments guided by the load forecasts. All data on server utilization, failures, and recovery cycles was collected for further analysis.

A heat map visualizes the server load under a random failure scenario, showing how different nodes are affected and indicating areas of potential congestion in Fig. 7.

The expected outcome of this scenario was to quantify how the frequency and duration of failures affect the cluster's overall performance. It also aimed to measure the effectiveness of predictive balancing strategies under failure conditions and to highlight potential bottlenecks and overload zones caused by hardware degradation.

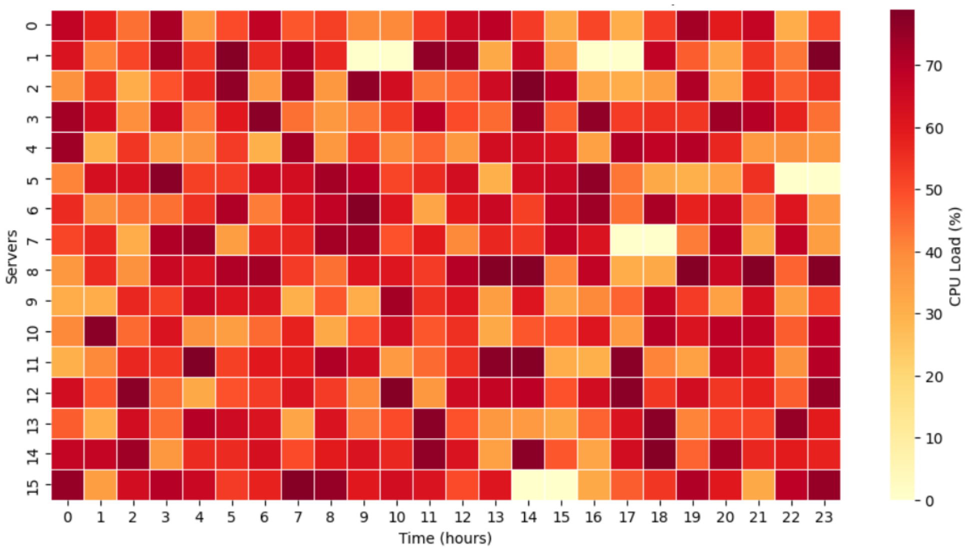

Fig. 7. Heat map of server load in a random failure scenario

The evaluation relied on metrics such as average and peak load of active servers, downtime and recovery time per server, the number of redirected requests, overall throughput, and system response time.

6. Discussion of the experimental results

The simulation experiments demonstrated the practical effectiveness of the proposed hybrid predictive load balancing mechanism within a web cluster environment based on the Fat-Tree topology. The evaluation involved three main experimental scenarios designed to test system performance under varying traffic conditions, fault events, and forecasting models.

In the baseline scenario, which represented a uniform distribution of requests without prediction, the system achieved an average response delay of 120 ms. The throughput in this scenario was approximately 980 requests per second. Under peak load conditions, the average delay increased to 270 ms, while throughput decreased by nearly 22%, indicating congestion in certain nodes due to unbalanced traffic.

To address this issue, the predictive control mechanism was activated, integrating Erlang-based analytical modeling for estimating service intensity and neural network forecasting (LSTM) for predicting short-term load variations. The hybridization of these methods allowed the system to adaptively redistribute traffic in near real-time. As a result, the average processing delay was reduced by 12% and the proportion of failed requests decreased by 18% compared to the non-predictive scenario.

When the predicted node load exceeded 80% of the allowable threshold, the system preemptively redirected new requests to less loaded servers. This proactive behavior significantly mitigated overload formation and reduced the variance of load distribution across the cluster by 18–25%. In comparison, a purely statistical forecasting method such as ARIMA showed lower adaptability under highly dynamic load conditions, confirming the advantage of deep learning-based predictors for volatile web traffic.

The fault-tolerance simulations introduced random node failures to evaluate cluster stability. The system maintained operational functionality even with up to 20% of nodes disconnected, primarily due to the redundant structure of the Fat-Tree topology. Through rapid rerouting and predictive redistribution, throughput decreased by only 15%, and the percentage of unserved requests remained below 5%. However, when failures exceeded this threshold, localized "hot spots" began to appear, suggesting the need for further optimization of load balancing in extreme fault conditions.

Overall, the discussion of experimental results confirms that the integration of hybrid forecasting methods (Erlang + LSTM) with adaptive load control enhances the robustness, responsiveness, and scalability of web cluster systems. This hybrid mechanism demonstrates the potential for practical deployment in intelligent cloud environments that require both real-time adaptability and predictive decision-making capabilities.

Future research could explore scaling up simulations to include a larger number of servers to evaluate system performance in extensive data centers. It could also involve applying deep learning techniques for load forecasting under complex scenarios such as seasonal or anomalous traffic. Additionally, analyzing the effects of real user traffic based on web server logs and developing adaptive routing protocols within the Fat-Tree topology could enhance network latency optimization.

Conclusions

1. The Fat-Tree-based web cluster model exhibited high scalability and resilience under varying load and failure conditions. Thanks to its multi-tiered structure and redundant interconnections, the topology ensured efficient traffic rerouting and sustained system functionality even during partial network degradation.

2. The proposed hybrid predictive load balancing mechanism combines analytical and intelligent forecasting methods – specifically, the Erlang model for service demand estimation and the LSTM neural network for short-term load prediction. This integration enables proactive control and reduces the dependency on reactive balancing strategies.

3. Quantitative evaluation showed that the hybrid approach reduced the average response delay by up to 12%. It also improved throughput stability by 18–25% and decreased the number of overload-related service failures by 30–40% compared to a baseline non-predictive balancing scheme. These metrics confirm a measurable gain in efficiency and system responsiveness.

4. Fault-tolerance analysis demonstrated that the cluster maintained stable operation with up to 20% node loss, while total unserved requests remained under 5% due to rapid rerouting mechanisms. This validates the reliability of the proposed architecture for critical and large-scale web systems.

5. The scientific novelty of this work lies in the development of a hybrid predictive control framework. This framework combines analytical modeling of request flows (Erlang-based traffic analysis), neural-based forecasting of web server load, and adaptive redistribution within a Fat-Tree cluster.

References

[1] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 892–901, 1985. https://doi.org/10.1109/TC.1985.6312192.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008. https://doi.org/10.1145/1402946.1402967.

[3] A. Greenberg *et al.*, "Vl2: A scalable and flexible data center network," *Communications of the ACM*, vol. 54, no. 3, pp. 95–104, 2009. https://doi.org/10.1145/1897852.1897877.

[4] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," *Computer Communication Review*, vol. 39, pp. 63–74, 2009. https://doi.org/10.1145/1592568.1592577.

[5] Y. Lohumi, D. Gangodkar, P. Srivastava, M. Khan, A. Alahmadia, and A. Alahmadia, "Load balancing in cloud environment: A state-of-the-art review," *IEEE Access*, pp. 1–14, 2023. https://doi.org/10.1109/ACCESS.2023.3337146.

[6] L. Wang *et al.*, "Cloud computing: A perspective study," *New Generation Computing*, vol. 28, no. 2, pp. 137–146, 2010. https://doi.org/10.1007/s00354-008-0081-5.

[7] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1094–1104, 2001. https://doi.org/10.1109/71.963420.

[8] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013. https://doi.org/10.1145/2408776.2408794.

[9] V. Cristea, C. Dobre, C. Stratan, F. Pop, and A. Costan, *Architectures for Large Scale Distributed Systems*. IGI Global, 2010. https://doi.org/10.4018/978-1-61520-703-9.ch002.

[10] M. Ferriol-Galmés *et al.*, "RouteNet-Erlang: A graph neural network for network performance evaluation," in *Proc. IEEE INFOCOM 2022*, 2022. https://doi.org/10.1109/INFOCOM48880.2022.9796944.

[11] K. Vimhala and L. Gowrishankar, "Modelling dynamic traffic loads in multiserver queues using G/G/k queue," *E3S Web of Conferences*, vol. 484, p. 01023, 2024. https://doi.org/10.1051/e3sconf/202448401023.

[12] C. Y. Chew, G. Teng, and Y. S. Lai, "Simulation of erlang and negative binomial distributions using the generalized lambert W function," *Journal of Computational Mathematics and Data Science*, vol. 10, p. 100092, 2024. https://doi.org/10.1016/j.jcmds.2024.100092.

[13] A. U. Rehman, R. Aguiar, and J. Barraca, "Fault-tolerance in the scope of cloud computing," *IEEE Access*, vol. 10, pp. 1–21, 2022. https://doi.org/10.1109/ACCESS.2022.3182211.

[14] J. Kumar, A. Singh, and R. Buyya, "Self directed learning based workload forecasting model for cloud resource management," *Information Sciences*, vol. 543, 2021. https://doi.org/10.1016/j.ins.2020.07.012.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] F. Agulló, A. Gutierrez-Torre, J. Torres, and J. L. Berral, "Enhancing the output of time series forecasting algorithms for cloud resource provisioning," *Future Generation Computer Systems*, vol. 170, p. 107833, 2025. https://doi.org/10.1016/j.future.2025.107833.

[17] K. O. Radchenko, "Application of discrete wavelet transformations for forecasting the level of load on the web server of general purpose computer networks," *Computer-Integrated Technologies: Education, Science, Production*, no. 45, pp. 90–96, 2021. https://doi.org/10.36910/6775-2524-0560-2021-45-13.

[18] K. O. Radchenko, "Conceptual model for ensuring the effectiveness of web server load forecasting," *Scientific notes of Taurida National V.I. Vernadsky University. Series: Technical Sciences*, vol. 31(70), no. 6, pp. 135–141, 2020. https://doi.org/10.32838/TNU-2663-5941/2020.6-1/23.

[19] K. O. Radchenko, "Peculiarities of predicting the level of web traffic in general purpose computer networks," *Problems of Informatization and Management*, no. 3(71), pp. 41–50, 2022. https://doi.org/10.18372/2073-4751.71.17002.

УДК 004.056.55 (519.8)

ПІДХІД ДО ГІБРИДНОГО КЕРУВАННЯ НАВАНТАЖЕННЯМ У ВЕБКЛАСТЕРАХ FAT-TREE

Костянтин Радченко
https://orcid.org/0000-0002-1282-6307

Артем Черненький
https://orcid.org/0009-0005-6925-4361

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

У статті представлений підхід до гібридного керування навантаженням у вебкластері, що здатен забезпечувати адаптивне балансування запитів на основі прогнозування навантаження та врахування випадкових відмов вебсерверів. Як топологічну основу обрано архітектуру Fat-Tree, що характеризується високою масштабованістю, структурною надмірністю та ефективністю маршрутизації. Запропонована система здійснює прогнозування навантаження за допомогою ковзного середнього та моделей черг типу Ерланга, що дозволяє оцінювати ймовірність перевантаження окремих вузлів і здійснювати проактивне перерозподілення ресурсів. У процесі моделювання розглянуто чотири типові сценарії функціонування кластера: базове, пікове та динамічне навантаження, а також випадкові відмови серверів. Отримані результати свідчать про підвищення відмовостійкості системи, зменшення середнього часу відгуку та більш рівномірне використання обчислювальних ресурсів. У сучасних умовах інтенсивного зростання кількості вебсервісів і користувацького трафіку актуальним є питання забезпечення надійності та ефективності кластерних інфраструктур. Навіть за використання стійких топологій, таких як Fat-Tree, нерівномірність запитів і пікові навантаження здатні спричиняти локальні перевантаження або зниження продуктивності. Випадкові відмови вузлів додатково ускладнюють управління системою, що потребує впровадження механізмів адаптивного прогнозного керування. Розроблена модель інтегрує мережеве моделювання Fat-Tree з алгоритмами статистичного аналізу, створюючи основу для побудови проактивної системи балансування навантаження. Такий підхід дозволяє зменшити ризик деградації сервісу, своєчасно реагувати на зміни у трафіку та підтримувати стабільну роботу вебінфраструктури навіть за умов часткових відмов. Запропонована архітектура демонструє значний потенціал для практичної реалізації у масштабованих вебсервісах. Її можна додатково вдосконалити за рахунок впровадження методів машинного навчання або вейвлет-прогнозування для підвищення точності оцінки навантаження та адаптивності системи.

Ключові слова: прогнозування навантаження, вебкластер, Fat-Tree топологія, відмовостійкість, балансування трафіку, стохастичні відмови.