Journal of Theoretical and Applied Information Technology
15th February 2025. Vol.103. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PRONGHORN SWIFT OPTIMIZATION FRAMEWORK (PSOF) A DYNAMIC AND SCALABLE APPROACH FOR ENHANCING WEB SERVICE PERFORMANCE

J. GNANABHARATHI[1], K. VADIVAZHAGAN[2]

[1] Research Scholar, Department of Computer and Information Science, Annamalai University,Chidambaram,Tamilandu India.
[2]Assistant Professor, Department of Computer and Information Science, Annamalai University,Chidambaram,Tamilnadu,India.
E-mail: [1]jbharathi.jb@gmail.com, [2]vadivazhagan@gmail.com

ABSTRACT

The Pronghorn Swift Optimization Framework (PSOF) is proposed to address critical challenges in web services, including high latency, inefficient resource utilization, and poor scalability. The growing demand for fast, reliable, and scalable web services has highlighted the need for frameworks capable of optimizing performance while maintaining low overhead. PSOF is designed to enhance the efficiency of data delivery, reduce response times, and improve system throughput under dynamic traffic conditions. The framework leverages adaptive mechanisms to handle fluctuating network loads, ensuring seamless performance across distributed environments. Simulation results reveal that PSOF significantly improves key metrics such as latency, throughput, and load balancing. PSOF creates a robust infrastructure for modern web applications by dynamically allocating resources and optimizing service interactions.PSOF bridges gaps in web service optimization by introducing a unified framework that enhances reliability and scalability, meeting the demands of evolving digital environments.

Keywords: *Pronghorn Swift Optimization, Web Service Optimization, Latency Reduction Scalability in Web Services, Throughput Enhancement, Load Balancing Techniques.*

1. INTRODUCTION

Web services facilitate communication between applications across diverse platforms and languages using standard protocols and formats. The purpose centres on interoperability, making web services a fundamental technology for integrating applications, especially in distributed environments [1]. Key components include XML or JSON for data representation, HTTP or HTTPS for communication, and protocols like SOAP and REST for standardizing request and response structures. A web service operates through a request-response model. Clients send requests to access specific services, and servers respond by fulfilling these requests [2]. This interaction often uses SOAP (Simple Object Access Protocol), which leverages XML for defining messages, and REST (Representational State Transfer), which operates through standard HTTP methods (GET, POST, PUT, DELETE) to manage resources. SOAP offers a structured way to interact with complex systems, supporting security and transaction controls, making it suitable for enterprise applications [3]. REST is lightweight and flexible, making it ideal for web-based applications and APIs that demand quick responses and high flexibility [4].

The purpose of web services is multifaceted. Primarily, they enable interoperability, which allows different software systems, possibly developed in other programming languages and running on various platforms, to communicate seamlessly. Organizations can use web services to build modular applications where different components function independently but communicate through standard interfaces [5]. This modularity allows developers to create, update, or replace individual services without impacting the entire system, promoting scalability and reducing maintenance complexity [6].

Web services also play a significant role in enabling cloud computing and distributed applications. They allow applications to interact with cloud-based resources, retrieving data, processing information, or store records on remote servers [7]. As a result, web services facilitate a wide range of cloud-based services, such as data storage, computational resources, and machine learning models, which

organizations can access over the Internet without requiring extensive infrastructure investments[8].

Security and reliability are also critical aspects of web services. Web service protocols support secure communication channels, typically HTTPS, and standards such as WS-Security for SOAP or OAuth for REST[9]. This emphasis on security ensures that sensitive data remains protected during transmission, addressing concerns surrounding data breaches and unauthorized access. Furthermore, web services enable reliable transactions across different systems by providing fault tolerance mechanisms, session management, and transactional support[10].

Web services have transformed e-commerce, social networking, and enterprise resource planning (ERP) by enabling seamless data flow across different applications. In e-commerce, for example, web services connect payment gateways, customer management systems, and inventory tracking, ensuring smooth transaction processing and inventory updates. Social networks utilize web services to enable third-party applications to securely access user profiles, posts, and other data, enhancing social connectivity[11]. ERP systems integrate various business processes using web services, allowing real-time data synchronization across departments like sales, accounting, and supply chain. Web services encounter performance and scalability challenges due to factors inherent in their architecture and communication protocols. Performance bottlenecks often arise from the overhead introduced by XML or JSON serialization, as both formats tend to be verbose and require additional processing time for parsing and validation[12]. When handling high data volumes, this overhead can lead to latency issues, as the system expends resources encoding, decoding, and transmitting extensive payloads. Network latency, compounded by frequent requests or large response sizes, further impacts performance, particularly in applications demanding real-time responses[13].

Scalability issues emerge when web services face limitations in handling concurrent requests, often due to inadequate resource management. As traffic increases, server resources like CPU and memory may become overtaxed, resulting in reduced response times and service failures. Statelessness in RESTful services, while enhancing simplicity, contributes to scalability constraints by requiring repeated authentication and resource fetching with each request, which consumes additional resources[14]. Load balancing and caching techniques are frequently applied to alleviate these constraints, but they demand careful configuration. Web services require robust infrastructure, optimized message formats, and efficient load distribution mechanisms to scale effectively in high-demand environments. To address high latency and enhance performance and scalability in web services, a practical framework involves specific optimization methods that improve data processing speed, reduce bottlenecks, and enable efficient resource utilization[15].

Bio-inspired computing represents an interdisciplinary approach that draws principles from natural biological processes to solve complex computational problems[16]. This field mimics the mechanisms observed in nature, such as evolution, self-organization, and collective behaviour, to develop efficient algorithms and systems. Techniques such as genetic algorithms, ant colony optimization, particle swarm optimization, and artificial neural networks are central to this domain[17]. Bio-inspired computing finds applications in optimization, machine learning, robotics, and data mining[18]. By leveraging the adaptability and resilience of biological systems, this approach addresses challenges in scalability, efficiency, and robustness[19]. Its potential for solving real-world problems continues to expand, fostering innovation across science, engineering, and technology[20].

1.1. Problem Statement

Web services have become integral to modern digital infrastructures, enabling seamless communication and data exchange between distributed applications. As demand for web services increases, high latency, inefficient resource utilization, and limited scalability hinder their ability to meet user expectations. Frequent packet loss, uneven load distribution, and suboptimal routing paths exacerbate these challenges, leading to degraded performance and reduced service reliability. Dynamic network conditions and fluctuating traffic patterns further complicate maintaining consistent quality of service.

Current frameworks often struggle to address these challenges effectively due to their inability to dynamically adapt to evolving traffic demands or optimize resource allocation. The absence of cohesive strategies for reducing latency and managing workloads contributes to inefficient operations, particularly in large-scale distributed systems. Furthermore, achieving high throughput and minimizing packet drop ratio while maintaining scalability remains a critical concern in web service environments. This scenario highlights the need for an advanced optimization framework that dynamically adjusts to varying conditions, ensures equitable load

distribution, reduces latency, and maximizes resource efficiency. The PSOF is proposed to address these gaps by providing a comprehensive, scalable solution for optimizing web services, enabling robust performance and reliability under diverse operating conditions.

1.2. Motivation

The increasing reliance on web services for real-time applications, data-intensive tasks, and distributed systems underscores the critical need for optimized frameworks. High latency, inefficient resource allocation, and uneven load distribution limit the system's ability to meet user demands effectively. Growing network complexity and fluctuating traffic patterns further exacerbate challenges, impacting performance and scalability. The necessity to address these challenges has motivated the development of a framework that ensures low latency, efficient resource utilization, and improved scalability across diverse web service environments. By leveraging adaptive mechanisms, optimization strategies, and dynamic resource management, the proposed PSOF aims to bridge gaps in current solutions. PSOF is designed to create a robust infrastructure capable of sustaining high throughput, equitable load distribution, and reliable performance, meeting the evolving demands of modern web service architectures.

1.3. Objective

The objective is to address critical challenges in web services by enhancing performance, scalability, and reliability. The framework minimizes latency, improves throughput, and ensures efficient resource utilization in distributed environments. By leveraging adaptive optimization techniques, PSOF aims to dynamically allocate resources and balance workloads, reducing bottlenecks and achieving equitable load distribution across network nodes.

PSOF handles fluctuating traffic patterns and high-density network conditions while maintaining seamless operations. The framework seeks to optimize data delivery paths, reduce packet drop ratios, and improve response times, creating a resilient infrastructure for modern web applications. An additional objective of PSOF is to ensure robust scalability by introducing mechanisms that dynamically adapt to varying network conditions. By integrating strategies that enhance resource management, caching efficiency, and routing precision, PSOF provides a unified solution to address the limitations of existing frameworks. The ultimate goal of PSOF is to create a high-performance web service architecture that meets the growing demands of real-time applications, ensuring consistent quality of service across diverse operational scenarios while bridging gaps in current optimization methodologies.

2. RESEARCH BACKGROUND AND CONTEXT

Web of Shadows[21]examines the misuse of legitimate internet services by malware, focusing on command-and-control, data exfiltration, and malware distribution. It introduces methodologies to identify and analyze malicious behaviours camouflaged within regular web service traffic. A comprehensive study of traffic patterns aids in understanding how malware adapts to exploit existing internet infrastructures. *Web-Based GIS*[22]This study introduces a GIS-based platform for biomass data management. The platform integrates spatial and tabular data layers, providing tools for users to access and analyze forestry and environmental information. The system emphasizes scalability, accessibility, and data accuracy for sustainable resource management.*QOS-QOE*[23] proposes an approach using feature selection techniques to improve quality models for mobile networks. It employs ordinal regression methods to prioritize significant features, enhancing the precision of Quality of Service (QoS) and Quality of Experience (QoE) evaluations in video and web-based services.

Medical-Webservice [24] presents a framework for achieving interoperability in hospital information systems. Composing web services enables efficient data sharing and seamless integration among diverse healthcare systems. The approach improves coordination, ensuring compatibility and enhancing overall healthcare management. *Dynamic Cluster*[25]introduces a novel dynamic clustering approach for aggregating web services using Minkowski similarity. The method optimizes service aggregation by calculating similarity metrics to group services with related functionalities. The clustering adapts to varying service characteristics, enhancing accuracy and efficiency in large-scale environments. The proposed technique facilitates effective grouping, ensuring users can efficiently access and utilize aggregated services. *GMF* [5] presents a modular framework designed for dark web analysis, emphasizing scalability and flexibility. The framework integrates advanced tools for data collection, content categorization, and behaviour analysis within dark web ecosystems. By supporting modular components, the framework enables tailored applications, allowing researchers to explore hidden activities while maintaining adaptability for various investigative

objectives.*APIRec*[26] is a system for recommending web APIs based on deep learning and diversity-aware techniques. It incorporates semantic analysis and contextual understanding to match APIs with user needs. The model emphasizes diversity in recommendations to ensure comprehensive solutions. The system enhances developer productivity by providing accurate and varied API suggestions through knowledge-driven mechanisms.

Snipweb[27]introduces a method for uncovering input-handling vulnerabilities in web applications. The approach analyses application behaviour under crafted inputs to identify security flaws. Techniques include automated testing and targeted injection methods to expose vulnerabilities such as cross-site scripting and injection attacks. The paper highlights systematic processes for enhancing web application security. *Cognitive Access*[28]This paper explores the challenges of implementing cognitive accessibility features within digital services. It identifies technical, design, and organizational barriers that hinder effective integration. The work highlights the need for improved frameworks and standards to address these issues, aiming to make digital platforms more inclusive for users with cognitive impairments. *Feedback monitoring* [29] proposes an inclusive framework for evaluating web content accessibility through automation. It combines standardized accessibility guidelines with advanced algorithms to assess compliance. The framework supports continuous monitoring and feedback, ensuring that web content remains accessible to users with diverse needs. The modular structure allows integration into existing development workflows.

ESPRESSO[30] is a framework that enhances search capabilities within decentralized web environments. It integrates distributed indexing, ranking algorithms, and metadata aggregation to improve the efficiency of decentralized search. The framework ensures secure and privacy-aware search mechanisms, addressing challenges distributed architectures pose while maintaining usability and scalability. *Graph-Based Web Service*[31] presents a framework leveraging multi-source information graphs for web service recommendations. Integrating functional attributes, user preferences, and service relationships constructs a graph-based representation of the web service ecosystem. Advanced algorithms analyze the graph to identify and recommend optimal services. This approach ensures personalized recommendations while enhancing the utility of service ecosystems. *Label Semantics*[3]proposes an enhanced web service clustering method combining label-based functional semantics and collaboration metrics. Incorporating labelled data improves the semantic representation of services, while service collaboration insights refine clustering accuracy. The technique organizes services into more coherent and meaningful clusters, supporting better discovery and management. Quality WS [32] proposes a semi-supervised learning method for classifying web services based on quality metrics. By combining labelled and unlabeled data, the approach enhances classification accuracy. It employs advanced algorithms to identify patterns and categorize services effectively, ensuring a comprehensive assessment of quality attributes in service ecosystems.

EdgeX [33]examines the implementation of a web service application using the EdgeX OpenEdge server, highlighting its potential for supporting IoT services. The framework focuses on enabling distributed edge computing to manage resource-constrained IoT devices efficiently. The architecture integrates microservices to facilitate device communication, data preprocessing, and local analytics. EdgeX's modularity allows flexible deployment of applications, while its compatibility with various protocols ensures seamless device integration. The study evaluates the platform's ability to reduce latency, enhance resource utilization, and provide scalability in IoT environments, emphasizing its viability as a solution for edge computing in dynamic and heterogeneous ecosystems.ARTP [34] is a machine learning-based framework for real-time detection and prevention of Distributed Denial of Service (DDoS) attacks on web systems. The methodology employs anomaly detection to identify irregular traffic patterns indicative of DDoS activities. Feature extraction techniques process network traffic data, allowing machine learning algorithms to classify legitimate and malicious requests. The framework incorporates real-time monitoring and adaptive learning to counteract evolving attack strategies. The prevention mechanism dynamically mitigates threats by filtering malicious traffic while ensuring uninterrupted access for genuine users. By leveraging data-driven insights and robust machine learning techniques, ARTP offers an efficient approach to safeguarding web infrastructures from DDoS attacks. More optimization techniques are being employed in networking domain research to get better results [35]-[59]

3. PRONGHORN SWIFT OPTIMIZATION FRAMEWORK (PSOF)

The Pronghorn Swift Optimization Framework (PSOF) utilizes Pronghorn's agility, speed, precision, and resilience attributes to enhance the performance and scalability of web services. This

structured approach aims to create a swift, low-latency response system by mirroring the Pronghorn's natural efficiency in rapid, sustained movement over long distances. The Pronghorn Swift Optimization Framework (PSOF) provides a comprehensive, agile solution for enhancing web service performance and scalability. By integrating these optimization techniques, PSOF leverages Pronghorn's speed, adaptability, and resourcefulness, creating a responsive, high-performing web service architecture.

3.1. Sprint Compression

Sprint Compression in the PSOF aims to reduce data size effectively, enabling swift data transfer between clients and servers. This compression technique is crucial for optimizing latency and decreasing network transmission time. By minimizing the data payload, Sprint Compression reduces the drag that can slow down performance, enhancing web service agility. In the optimization of data compression, let the size of the uncompressed data be represented as D_u and the compressed data as D_c. The compression ratio C_r quantifies the reduction achieved and is defined as shown in Eq.(1).

$$C_r = \frac{D_u}{D_c} \quad (1)$$

where a higher C_r indicates a more efficient compression. Sprint Compression in PSOF by targeting maximum C_r, supports rapid data transfers by minimizing D_c relative to D_u. Each element in the compression process contributes to achieving an optimized transfer process with reduced latency.

Considering the time required to compress data, let T_c be the time taken to compress the data, T_d be the time required to decompress it, and T_t represent the transmission time of compressed data. The total time T_{total} for transmitting and reconstructing data can be described as expressed mathematically in Eq.(2).

$$T_{total} = T_c + T_t + T_d \quad (2)$$

Sprint Compression optimizes T_{total} by minimizing T_c and T_d through efficient algorithms, as well as reducing T_t by lowering D_c. An optimized algorithm ensures that both T_c and T_d are kept at minimal values, maintaining the flow of requests and responses without unnecessary delays.

In Sprint Compression, entropy $H(D)$, representing the average amount of information per data unit, plays a critical role. A lower entropy results in more predictable data patterns, which compression algorithms can exploit to reduce data size. Entropy for data D with probability distribution $p(x)$ is calculated as represented mathematically in Eq.(3).

$$H(D) = -\sum_{x \in D} p(x) log_2 p(x) \quad (3)$$

Efficient data compression algorithms minimize $H(D)$, achieving an optimized D_c by capitalizing on redundancies. In Sprint Compression, reducing $H(D)$ optimizes the process, akin to the Pronghorn's swift response to low-resistance paths, improving web service speed.

The latency impact of data transfer can be further understood by considering bandwidth B, which represents the amount of data transmitted per unit time, with data transfer time T_d defined as shown in Eq.(4).

$$T_d = \frac{D_c}{B} \quad (4)$$

Optimized Sprint Compression reduces D_c to optimize T_d, enhancing the throughput across web services. By minimizing T_d, the framework achieves near-instantaneous data transfers, similar to the Pronghorn's agile movements across terrain with minimal resistance.

Another significant factor in Sprint Compression involves the bit rate R, or the number of bits per unit of time required to represent compressed data. An efficient Sprint Compression algorithm lowers R to match or fall below network capacity C, preventing bottlenecks. This relationship is expressed mathematically in Eq.(5).

$$R = \frac{D_c.8}{T_d} \quad (5)$$

where an optimized bit rate R aligns with the network's capacity, allowing the data to pass without congestion. The pronghorn-inspired approach adjusts D_c and R for optimized bandwidth utilization, minimizing lag.

Error correction is essential in Sprint Compression to prevent data loss or corruption during transmission. Let the probability of an error occurring in transmission be P_e. The probability of successfully transmitting compressed data without error is depicted in Eq.(6).

$$P_s = 1 - P_e \quad (6)$$

An optimized error correction approach in Sprint Compression minimizes P_e by implementing checksums or error correction codes (ECC). This enhances reliability, mirroring the Pronghorn's optimized path selection, where the risk of errors in navigating challenging terrain is minimized.

The compression rate R_c further characterizes the efficiency of Sprint Compression. Defined as the ratio of the reduction in data size per second, R_c aligns with how effectively the PSOF reduces data load. R_c can be calculated as shown in Eq.(7).

$$R_c = \frac{D_u - D_c}{T_c} \quad (7)$$

A higher R_c reflects a faster, more efficient compression process, optimizing Sprint Compression to support rapid, continuous data flow. By focusing on maximizing R_c, PSOF aligns with pronghorn-inspired performance, ensuring agile data exchanges.

The efficiency E of Sprint Compression, measuring the effectiveness of the compression process compared to the theoretical limit, is calculated mathematically in Eq.(8).

$$E = \frac{C_r}{H(D)} \quad (8)$$

Higher E values indicate a closer approach to optimal compression, where $H(D)$ is minimized, and C_r reaches its peak. This efficiency parallels the Pronghorn's energy-conserving movements, where the path taken minimizes resistance, optimizing travel speed. By utilizing these parameters, Sprint Compression enhances web service responsiveness by aligning with Pronghorn's optimized, swift characteristics.

3.2. Leap Serialization

Leap Serialization in the PSOF employs an optimized approach to reduce latency by streamlining data serialization and deserialization processes. Leap Serialization achieves efficiency through minimized data format overhead, enhancing the speed of encoding and decoding, similar to how a pronghorn navigates swiftly by leaping over obstacles. Central to PSOF, this process ensures agile data movement between applications, transforming structured data into a compact format for faster network transmission.

Encoding efficiency is critical to reduce serialized data size. Let E_d represent the encoding duration, which depends on the size of the data D_s and the complexity C of the data structure is represented mathematically in Eq.(9).

$$E_d = D_s \times C \quad (9)$$

Optimizing E_d It is essential to align with PSOF's aim for minimal latency. A lower E_d reduces the computational load for encoding, allowing Leap Serialization to facilitate rapid data exchanges reflecting Pronghorn's optimized leaps over terrain.

Compression methods are used with serialization to decrease data redundancy and enhance speed further. A reduction factor represents the effectiveness of compression for serialized data. R_f, calculated as shown in Eq.(10).

$$R_f = \frac{D_O}{D_s} \quad (10)$$

where D_O is the original data size and D_s is the size after serialization. A higher R_f signifies greater efficiency, optimizing Leap Serialization to support swift transmission by minimizing data that needs to pass through the network, thus emulating the Pronghorn's efficient, high-speed movement.

A key factor influencing serialization speed in Leap Serialization is the number of fields F_n in the data object. Each field requires individual encoding, which impacts the total serialization time T_s. The relationship is expressed mathematically in Eq.(11).

$$T_s = F_n \times E_d \quad (11)$$

Minimizing F_n without losing data integrity, reduces T_s, allowing for streamlined data transfer. This aspect of Leap Serialization mirrors Pronghorn's agile pathfinding, where optimal leaps minimize the total energy expended.

Network transmission efficiency in Leap Serialization depends on both the bit rate B_r of serialized data and network capacity N_c. Transmission delay T_t can be calculated as depicted in Eq.(12).

$$T_t = \frac{D_s}{B_r} \quad (12)$$

An optimized T_t enhances LeapSerialization's effectiveness by aligning the serialized data output with network capabilities, reducing the likelihood of data congestion and allowing data to move fluidly across systems.

The deserialization process also contributes significantly to overall latency. Let D_d denote deserialization duration, determined by the serialized data size $D_s$and the computational efficiency C_e of the decoding algorithm is represented mathematically in Eq.(13).

$$D_d = \frac{D_s}{C_e} \quad (13)$$

Higher $C_e$values ensure faster deserialization, enabling Leap Serialization to handle incoming data with minimal delay. This efficiency echoes the Pronghorn's instinct to conserve energy by choosing the most efficient paths.

Leap Serialization further optimizes latency by encoding data in a binary format. Binary encoding reduces the data size compared to text formats like JSON or XML. Let B_f represent the binary format size and T_f the text format size. The conversion ratio $C_r$is represented mathematically in Eq.(14).

$$C_r = \frac{T_f}{B_f} \quad (14)$$

A high C_r indicates substantial data reduction, improving LeapSerialization's speed by minimizing payload. The lean binary format supports efficient data leaps between systems, akin to the Pronghorn's streamlined leaps that eliminate unnecessary detours.

Error correction in serialized data is managed to ensure reliability without excessive redundancy. Let P_e represent the probability of error and E_r the error correction capacity. The error resilience $R_e$of Leap Serialization is mathematically expressed in Eq.(15).

$$R_e = 1 - P_e \times E_r \quad (15)$$

Optimized error resilience keeps data intact while reducing the need for retransmission. By maintaining robust error control, Leap Serialization enhances reliability, similar to how the Pronghorn navigates its terrain with precision and low risk of missteps.

The compression time C_t is also crucial in Leap Serialization, especially for large data sets. Let S_d represent the data set size and F_c the efficiency of the compression algorithm. The compression duration C_t can be determined as Eq.(16).

$$C_t = \frac{S_d}{F_c} \quad (16)$$

A higher $F_c$reduces C_t, allowing serialized data to flow without unnecessary delay. Optimized compression reflects the Pronghorn's instinct for rapid, continuous movement, where speed is prioritized without compromising on reliability.

Leap Serialization efficiency can be quantified through an optimization coefficient.O_c, which measures the ratio of reduction in size to time saved, expressed as shown in Eq.(17).

$$O_c = \frac{R_f}{T_s + T_f + D_d} \quad (17)$$

A high O_c aligns with the PSOF framework's goal of swift, seamless data exchanges, mirroring the Pronghorn's adaptive capabilities for optimized, energy-efficient traversal across expansive terrain. Leap Serialization provides a structured approach to streamline data movement, leveraging encoding precision, minimized data fields, binary formats, and efficient compression. By optimizing each element, Leap Serialization enables high-speed, low-latency data handling across networked environments, inspired by the Pronghorn's agility in rapidly overcoming obstacles.

3.3. Boundless Asynchronous Processing

This phase provides a continuous, unbounded flow, mirroring the Pronghorn's ability to sustain rapid movements across varied terrain. By decoupling request handling from response generation, Boundless Asynchronous Processing reduces latency and increases throughput, facilitating high-speed, seamless data exchanges. In Boundless Asynchronous Processing, let R_{in} denote the rate of incoming requests and R_{out} the rate of responses. An optimized system requires balancing. R_{in} and $R_{out}$to prevent bottlenecks. The mathematical representation of optimal performance is shown in Eq.(18).

$$R_{in} \approx R_{out} \quad (18)$$

Maintaining this equilibrium ensures that requests are processed as they arrive, preventing latency build-up and optimizing the throughput, akin to the Pronghorn's swift adaptation to varying speed requirements in its environment.

A critical aspect of asynchronous processing involves the duration T_p for processing each request. For an individual task i, the average processing time can be described in Eq.(19).

$$T_{p,i} = T_{queue,i} + T_{exec,i} \quad (19)$$

where $T_{queue,i}$represents the time spent in the queue and $T_{exec,i}$ the execution time. Optimizing $T_{p,i}$ for each request enables the system to achieve high responsiveness, avoiding delays typical of synchronous systems and ensuring an efficient data flow.

Considering concurrent requests, let N denote the number of tasks processed simultaneously, and T_c the time saved through concurrency. The effective time T_{eff} when processing N requests asynchronously is expressed as Eq.(20).

$$T_{eff} = \frac{\sum_{i=1}^{N} T_{p,i}}{N} \quad (20)$$

Minimizing T_{eff}Asynchronous handling allows Boundless Asynchronous Processing to manage larger workloads without delay, reflecting Pronghorn's ability to manage sustained speed across multiple sprints.

To further optimize performance, the handling capacity $C_h$defines the maximum number of requests a server can manage concurrently. For sustainable processing, R_{in} must not exceed C_h as expressed mathematically in Eq.(21).

$$R_{in} \leq C_h \quad (21)$$

By regulating $R_{in}$to stay within C_h, the system prevents overloading, enabling efficient utilization of resources. This approach mirrors the Pronghorn's energy conservation strategy, where sustained effort prevents exhaustion, ensuring agility across distances.

In Boundless Asynchronous Processing, event-driven handling minimizes idle time, allowing for optimal resource utilization. Let E_{idle} represent idle time per task and T_{exec} total execution time. The efficiency $E_f$of asynchronous handling is expressed as Eq.(22).

$$E_f = \frac{T_{exec}}{T_{exec}+E_{idle}} \quad (22)$$

Maximizing E_f increases the number of requests processed per unit time, enhancing system responsiveness, akin to the Pronghorn's continuous momentum without rest breaks, ensuring a smooth transition from one task to the next.

Resource consumption in asynchronous processing plays a crucial role in maintaining system efficiency. Let C_{mem} denote memory consumption per request and C_{total} total memory capacity. For stable operation, the inequality $N \times E_f \leq C_{total}$ must hold as expressed in Eq.(23).

$$N \times C_{mem} \leq C_{total} \quad (23)$$

This constraint ensures that resource allocation remains within capacity, avoiding memory overflows that can impede performance. By aligning with the Pronghorn's instinct for balanced energy use, Boundless Asynchronous Processing manages resources to sustain high-speed operations.

The probability P_{block} of encountering blocked operations in synchronous handling is high due to sequential dependencies. In asynchronous processing, the reduction factor R_b for blocked operations can be defined mathematically in Eq.(24).

$$R_b = 1 - P_{block} \quad (24)$$

Increasing R_b optimizes the workflow by allowing requests to be completed independently, avoiding delays due to sequential constraints. This reduction of blockages is analogous to the Pronghorn's ability to evade obstacles, enhancing the continuity of data handling.

Another aspect of Boundless Asynchronous Processing involves the latency L associated with message passing. For asynchronous requests, latency depends on network factors and internal processing. Let T_{net} denote network latency and T_{proc}internal processing delay; then total latency $L_{total}$is expressed mathematically in Eq.(25).

$$L_{total} = T_{net} + T_{proc} \quad (25)$$

Minimizing L_{total} through streamlined processing aligns with PTSD's goal of responsive interactions. By lowering T_{net} and T_{proc}, Boundless Asynchronous Processing achieves low-latency responses, enhancing the user experience.

The throughput $Th$of asynchronous systems represents the number of requests processed per unit of time and is a crucial measure of efficiency. Throughput is calculated as shown in Eq.(26).

$$Th = \frac{R_{in}-R_{out}}{T_{total}} \quad (26)$$

An optimized throughput ensures maximum task completion within the available time frame, mirroring the Pronghorn's optimized speed. Maximizing ThBoundless Asynchronous Processing guarantees high performance under fluctuating loads, maintaining continuity in web service operations.In

Boundless Asynchronous Processing, each request is handled with minimal delay through non-blocking operations, high concurrency, and efficient resource utilization. Thus, The framework balances responsiveness and resource management, effectively supporting high-speed, low-latency operations. Emulating the Pronghorn's instinct for swift adaptation, Boundless Asynchronous Processing empowers web services to manage fluctuating demands while maintaining optimal performance, enhancing overall throughput and service agility in the framework.

3.4. Steady Pace Load Balancing

Steady Pace Load Balancing in the PSOF applies a systematic approach to distributing incoming network traffic evenly across multiple servers, ensuring that no single server bears an excessive load. This balanced distribution enables consistent service speed, agility, and reliability, akin to the Pronghorn's sustainable, steady pace. Steady Pace Load Balancing optimizes web service performance by preventing resource exhaustion, enabling seamless scalability and high availability across fluctuating traffic levels. In Steady Pace Load Balancing, incoming requests are represented as R_{in}, are managed by multiple servers to ensure even distribution. Let S denote the number of available servers and R_s the requests allocated per server. The fundamental load distribution is defined as Eq.(27).

$$R_s = \frac{R_{in}}{S} \quad (27)$$

This relationship ensures an optimal allocation of requests where R_s remains consistent with each server's capacity. By balancing R_s across S, Steady Pace Load Balancing supports steady service performance without overburdening any individual server, reflecting the Pronghorn's steady movement across long distances.

The capacity of each server, represented by C_s, limits the maximum number of requests it can process effectively. To avoid overload, the condition $R_s \leq C_s$ must be met as specified in Eq.(28).

$$R_s \leq C_s \quad (28)$$

This constraint ensures that each server handles only as much traffic as it can efficiently process, preventing latency spikes due to overloading. Like the Pronghorn's pace regulation to avoid exhaustion, Steady Pace Load Balancing aligns incoming requests with each server's capabilities.

Considering network latency L_n, which affects the overall response time; balancing aims to minimize L_n by optimizing server utilization. For an optimized load-balancing configuration, the target latency L_{opt} should be less than or equal to the maximum tolerable latency L_{max} which is represented mathematically in Eq.(29).

$$L_{opt} \leq L_{max} \quad (29)$$

Achieving this balance prevents delays in server response times, maintaining a smooth and efficient request-response cycle. By targeting L_{opt}, the PSOF framework aligns with the Pronghorn's instinctive optimization of movement for sustained performance.

Throughput Th_s, defined as the number of requests processed by each server per unit of time, must match demand to prevent performance degradation. For effective load balancing, the throughput across all servers Th_{total} should equal the total incoming request rate as represented mathematically in Eq.(30).

$$Th_{total} = \sum_{i=1}^{S} Th_{s_i} = R_{in} \quad (30)$$

This total throughput calculation ensures that Steady Pace Load Balancing maintains service continuity under varying loads, achieving equilibrium akin to the Pronghorn's sustained movement in response to external pressures.

Efficiency in load balancing can be further quantified by load variance V_l, representing the difference in workload distribution among servers. Minimizing V_l enhances system stability and resource utilization. Load variance is calculated as expressed in Eq.(31).

$$V_l = \frac{\sum_{i=1}^{S} \left(R_{s_i} - \bar{R}_s\right)^2}{S} \quad (31)$$

where $\bar{R}_s$ represents the average requests per server. Lower V_l reflects a more balanced distribution, enabling consistent server performance and resembling the Pronghorn's smooth pacing, where no part of its movement disrupts the overall balance.

Another critical aspect of Steady Pace Load Balancing involves fault tolerance, defined by the probability P_f of a server failing to process a request. With fault tolerance mechanisms, steady pace load balancing is minimized. P_f by redistributing traffic upon server failure. The fault tolerance factor F_t is expressed as shown in Eq.(32).

$$F_t = 1 - P_f \quad (32)$$

A high F_t ensures that failed requests are re-routed to active servers, maintaining stability and preventing service interruptions. By managing server failures, the framework emulates the Pronghorn's instinctive adaptability, enhancing reliability in dynamic conditions.

Server utilization U_s measures the active workload on each server, calculated by the ratio of requests processed R_s to the server's total capacity C_s as depicted in Eq.(33).

$$U_s = \frac{R_s}{C_s} \quad (33)$$

A utilization rate within optimal levels maintains system efficiency and prevents overloading. This balance, where $U_s$is neither too low nor too high, ensures that resources are effectively used, reflecting thepronghorn's ability to sustain steady effort without depleting energy.

Balancing algorithms used in Steady Pace Load Balancing, such as round-robin or least-connections, adapt dynamically based on real-time server load data. Let $R_{current}$ denote the current load of each server, and R_{min} represent the load on the least-loaded server. The dynamic allocation rule is specified in Eq.(34).

$$R_{new} = min(R_{current}) \quad (34)$$

This allocation ensures that new requests are directed toward the server with the lowest load, distributing resources efficiently. The adaptive distribution, akin to the Pronghorn's route selection based on terrain, maintains balance and optimizes response times.

Latency distribution L_d across servers indicates the uniformity of request handling. To minimize $L_d$discrepancies, latency variance V_l across servers should approach zero.

$$V_l = \frac{\sum_{i=1}^{S}\left(L_{s_i} - \bar{L}_s\right)^2}{S} \quad (35)$$

In Eq.(35), where $\bar{L}_s$ denotes the average latency per server. A low V_lPromote consistent server response times, enabling uniform service delivery and emulating the Pronghorn's balanced strides. Steady Pace Load Balancing within PSOF uses optimized traffic distribution, load variance minimization, fault tolerance, and dynamic load allocation to maintain equilibrium in high-demand environments. Steady Pace Load Balancing achieves consistent, optimized service delivery across servers through this balanced approach, reflecting the Pronghorn's adaptive and enduring pace over varied terrain.

3.5. Quick Cache Recall

Quick Cache Recall in the PSOF enhances web service performance by reducing data retrieval time through efficient caching mechanisms. This process uses memory-based storage to temporarily hold frequently accessed data, allowing the system to retrieve this data quickly without repeated access to the primary database. Like the Pronghorn's swift adaptation to its environment, Quick Cache Recall ensures agility in data access, optimizes response times, and reduces latency in high-demand scenarios.

The cache hit rate can quantify the effectiveness of Quick Cache Recall H_r, which represents the percentage of requests successfully retrieved from the cache rather than the primary database. Defined as the ratio of cache hits C_h to the total requests R_t,H_r is calculated as shown in Eq.(36).

$$H_r = \frac{C_h}{R_t} \quad (36)$$

A high H_r reflects efficient cache utilization, minimizing database queries and reducing response times. Quick Cache Recall in PSOF aims to maximize H_r, allowing rapid data retrieval akin to the Pronghorn's instinctive recall of efficient routes, enhancing overall system responsiveness.

The data retrieval time $T_r$is divided between the cache retrieval time T_{cache} and the database retrieval time T_{db}. In cases of a cache hit, the retrieval time T_{hit} is expressed mathematically in Eq.(37).

$$T_{hit} = T_{cache} \quad (37)$$

The event of a cache miss, the retrieval time T_{miss} the cache retrieval and database access time are incorporated as expressed in mathematical equation Eq.(38).

$$T_{miss} = T_{cache} + T_{db} \quad (38)$$

Quick Cache Recall aims to minimize T_{miss}The system can provide fast responses and maintain an uninterrupted flow of data requests by optimizing caching strategies. This efficient recall of data aligns with the Pronghorn's quick reflexes, where instant decisions on terrain enhance its speed and agility.

Cache capacity C_{max} plays a critical role in determining the volume of data stored for quick

access. As data accumulates, the cache size $C_s$must be managed to stay within C_{max} to prevent overflow. The conditions for maintaining optimal cache usage are represented in Eq.(39).

$$C_s \leq C_{max} \quad (39)$$

This constraint ensures that the cache remains within memory limits, allowing the system to operate efficiently without consuming excessive resources. Like the Pronghorn's capacity to gauge distances and conserve energy, Quick Cache Recall manages storage for high efficiency.

Eviction policies are crucial for maintaining cache efficiency, dictating which data to remove when it reaches its limit. Let E_r represent the rate of cache eviction, which balances the inflow of new data D_{in} with the outgoing data D_{out} to maintain the cache within capacity is depicted mathematically in Eq.(40).

$$E_r = D_{out} - D_{in} \quad (40)$$

An optimized E_r aligns with demand, ensuring that frequently accessed data remains accessible while lesser-used data is evicted. This strategy mirrors Pronghorn's adaptive approach to maintaining optimal performance by prioritizing high-value resources.

The probability $P_{hit}$of a cache hit is directly proportional to the cache's efficiency in storing frequently accessed data. P_{hit}can be represented by the ratio of frequently accessed data $D_f$to total data D_t as shown in Eq.(41).

$$P_{hit} = \frac{D_f}{D_t} \quad (41)$$

Quick Cache Recall optimizes $P_{hit}$to maintain high availability of critical data, reducing reliance on the primary database. Similar to the Pronghorn's instinct for efficiency, a high P_{hi} enhances agility in data handling, allowing for fast and reliable responses.

Cache refresh rate R_f, which represents the frequency of updating the cache with new data, impacts the cache's relevance. The refresh interval $T_{refresh}$is inversely proportional to R_f which is represented mathematically in Eq.(42).

$$T_{refre} = \frac{1}{R_f} \quad (42)$$

A carefully managed R_f Keep the cache up-to-date without excessive overhead, ensuring the Quick Cache Recall process is efficient. This approach reflects the Pronghorn's adaptation to changing environments, where strategic decisions maintain speed and resourcefulness.

Memory efficiency $M_e$within the cache ensures optimal resource utilization, calculated as the ratio of compelling data stored $D_{eff}$to the total allocated memory M_{total}.

$$M_e = \frac{D_{eff}}{M_{total}} \quad (43)$$

In Eq.(43) where High M_e values indicate that cache memory is used effectively, storing only relevant data and avoiding wastage. In Quick Cache Recall, optimizing M_e enhances system efficiency, similar to the Pronghorn's selective use of resources to maintain agility and endurance over long distances.

The cache response time T_{resp} the time taken to retrieve data from the cache and serve the request must remain low to maximize performance. The idealT_{resp} is expressed as Eq.(44).

$$T_{resp} = \frac{C_h \times T_{cache}}{R_t} \quad (44)$$

This response time metric helps in evaluating cache efficiency, where lower T_{resp} values indicate quick access to stored data. Through optimized Quick Cache Recall, PSOF achieves rapid response times, enabling web services to handle high volumes of requests seamlessly, reflecting Pronghorn's swift movements across vast terrains. Quick Cache Recall enhances overall system agility by reducing database dependency and maintaining high cache efficiency. The strategic use of cache hits, optimized retrieval times, memory efficiency, and precise eviction policies provide fast and continuous access to critical data.

3.6. Edge Sprinting

Edge Sprinting in the PSOF enhances data transmission efficiency by positioning computation closer to users. This process leverages edge servers distributed across different locations to minimize data travel distance and latency, optimizing response times, similar to how the Pronghorn strategically navigates its terrain for swift movement. Edge Sprinting enhances service performance and scalability through this approach, particularly for latency-sensitive applications. In Edge Sprinting, data latency L_d plays a critical role in determining the responsiveness of data exchanges. Let D represent the physical distance between the user and the edge

server, and V denote the data transmission speed. The latency L_d is defined mathematically in Eq.(45).

$$L_d = \frac{D}{V} \quad (45)$$

Reducing D by placing servers closer to the user lowers L_d, facilitating quicker data exchanges. The Edge Sprinting approach mirrors the Pronghorn's strategy of minimizing path resistance and achieving optimal speed across vast terrains.

Bandwidth efficiency B_e in Edge Sprinting, the utilization rate of network bandwidth is represented when transferring data. For a data size S_d and available bandwidth B, bandwidth efficiency is calculated.

$$B_e = \frac{S_d}{B} \quad (46)$$

In Eq.(46), where higher B_e values indicate efficient usage of bandwidth. By optimizing B_e, Edge Sprinting reduces network congestion, improving data transfer speed between edge servers and users. This efficiency reflects the Pronghorn's instinctive energy conservation, allowing quick sprints without overtaxing resources.

Edge server placement, a crucial aspect of Edge Sprinting, depends on user distribution U_d across a region. Let N_e represent the number of edge servers and R_u the request rate from users. Optimal edge server distribution is achieved when the condition is expressed mathematically in Eq.(47).

$$U_d \approx \frac{R_u}{N_e} \quad (47)$$

Aligning U_d with R_u/N_e ensures that each server effectively manages local demand, avoiding overload and reducing response times. This strategic positioning aligns with the Pronghorn's navigation to advantageous terrains, ensuring sustained speed and stability across high-traffic routes.

Processing latency P_l at the edge server also affects overall response time. For a data processing task T_p,P_l is represented as shown in Eq.(48).

$$P_l = T_p \times N_d \quad (48)$$

where N_d denotes the number of data packets handled. Lower P_l values in Edge Sprinting contribute to faster service delivery, resembling the Pronghorn's rapid response in familiar terrain, where minimal resistance leads to swift movement.

Network latency N_l measures the time data packets travel between edge servers and the central data center. This latency is calculated based on data size S_d and transmission rate R_t.

$$N_l = \frac{S_d}{R_t} \quad (49)$$

In Eq.(49), where Reducing N_l efficient edge server placement in Edge Sprinting minimizes delays associated with distant data centers. This setup, akin to the Pronghorn's choice of proximity-based paths, optimizes service response by reducing travel distances for data.

Data replication across edge servers maintains data availability and reduces access time. The replication factor R_f, indicating the number of copies stored at different servers, ensures data accessibility. For a data object D_o with replication across N_e servers, R_f is given as shown in Eq.(50).

$$R_f = \frac{D_o \times N_e}{D_{total}} \quad (50)$$

where D_{total} represents total data managed. Optimizing R_f ensures each server has a locally accessible copy, reducing the need for remote retrieval and enhancing data access speed. This replication strategy parallels the Pronghorn's memory of resource locations, supporting rapid access without repetitive travel.

Edge Sprinting also leverages load distribution to balance the incoming request rate R_{in} across edge servers. Let C_s denote the handling capacity of each server and R_s the rate of requests assigned to each server. The load balancing condition is represented mathematically in Eq.(51).

$$R_s \leq C_s \quad (51)$$

Ensuring $R_s \leq C_s$ distributes load evenly, preventing any single server from becoming a bottleneck. This balance reflects the Pronghorn's instinct to distribute energy efficiently across movements, ensuring consistent performance.

Throughput T_h, measuring the rate of data processed by each edge server plays a critical role in maintaining smooth data exchanges.

$$T_h = \frac{S_d \times R_{in}}{T_{proc}} \quad (52)$$

In Eq.(52), where T_{proc} denotes the processing time per data unit. By maximizing T_h, Edge Sprinting

maintains high data processing rates, enabling fast service delivery that resembles Pronghorn's adaptive capacity for continuous movement without lag.

Edge resilience E_r ensures that each edge server continues to operate smoothly even during high demand or minor failures. Let P_f represent the failure probability of an edge server. The resilience factor E_r is expressed as Eq.(53)

$$E_r = 1 - P_f \quad (53)$$

A higher E_r enhances the stability of Edge Sprinting, ensuring consistent access to services regardless of individual server issues. This resilience reflects the Pronghorn's instinct for navigating challenging terrain without interruption. Edge Sprinting in PSOF enhances web service responsiveness and scalability through optimized latency, bandwidth, and load distribution.

3.7. Focus Query Optimization

Focus Query Optimization in the PSOF enhances database query performance by streamlining query execution paths and minimizing data retrieval time. This optimization technique eliminates unnecessary steps in the query process, ensuring efficient data access and reducing latency. Focus Query Optimization echoes Pronghorn's instinctive efficiency in avoiding obstacles and conserving energy, ensuring that data requests are handled quickly and precisely. Efficient queries in Focus Query Optimization are defined by their execution time T_q, which depends on the complexity of the query C_q and the amount of data D_r retrieved. The basic formula for query time is given as Eq.(54).

$$T_q = C_q \times D_r \quad (54)$$

Reducing T_q Through query simplification and targeted data access, the system can handle requests swiftly, mirroring the Pronghorn's calculated movements across its terrain, where minimal energy expenditure is prioritized to sustain speed over long distances.

Indexing represents a critical component in Focus Query Optimization, as it reduces search time by organizing data in a structured manner. Let T_{idx} represent the time saved through indexing for a dataset size S_d with index efficiency E_{idx}. The time savings T_{idx} is expressed as shown in Eq.(55).

$$T_{idx} = \frac{S_d}{E_{idx}} \quad (55)$$

A higher E_{idx} enables faster data retrieval, minimizing T_q for repeated queries. This efficiency reflects the Pronghorn's adaptation to optimal routes, where every movement reduces time spent on unnecessary detours, enhancing overall speed.

The selectivity S_q a query that measures the fraction of data retrieved relative to the total dataset significantly reduces unnecessary data processing. Selectivity S_q is defined as Eq.(56).

$$S_q = \frac{D_r}{D_t} \quad (56)$$

where D_t represents the total data. Lower S_q values achieved through optimized filtering result in faster query responses, aligning with the Pronghorn's ability to navigate selectively and bypass obstacles to maintain momentum and precision.

Caching frequently accessed query results further enhances query efficiency in Focus Query Optimization. Let H_c represent the cache hit rate for queries and T_{cache} the retrieval time from the cache. The overall retrieval time $T_{retrieve}$ for a cached query is expressed as Eq.(57).

$$T_{retrieve} = H_c \times T_{cache} + (1 - H_c) \times T_q \quad (57)$$

Maximizing H_c reduces dependence on database retrieval, enhancing response times and reducing T_q for recurring queries. This caching approach resembles Pronghorn's instinct for familiarity, where habitual paths are recalled quickly, minimizing the time spent searching.

Focus Query Optimization also considers the cost C_{exec} of executing complex queries, where each step in the query process requires resources. The execution cost is determined by Eq.(58).

$$C_{exec} = C_q \times R_r \quad (58)$$

where R_r represents the resource usage per query. By reducing C_q through query refinement, Focus Query Optimization reduces C_{exec}, ensuring that resources are allocated effectively, similar to how the Pronghorn efficiently uses its energy reserves, conserving them for critical movements.

In scenarios involving multiple joined tables, query optimization includes minimizing the number of joins J required. Each join increases processing

time, so reducing J enhances query efficiency. The time T_{join} for queries with multiple joins is calculated as expressed mathematically in Eq.(59).

$$T_{join} = J \times T_q \quad (59)$$

Reducing J through optimized data structure design and selective filtering aligns with Pronghorn's adaptive approach to reduce unnecessary movements, achieving efficient data processing through focused paths.

Partitioning further supports Focus Query Optimization by dividing large datasets into smaller, more manageable segments. Let P_n represent the number of partitions and T_p the time taken to retrieve data from each partition. Total partition retrieval time T_{part} is determined in Eq.(60).

$$T_{part} = \frac{T_q}{P_n} \quad (60)$$

A higher P_n reduces T_{part}, enhancing retrieval speed, particularly for queries targeting specific data segments. This partitioning approach aligns with Pronghorn's terrain navigation, where particular routes are chosen to maintain speed and avoid slower pathways.

Parallel processing, where queries are divided into smaller sub-tasks executed concurrently, significantly improves query performance. Let N_p denote the number of parallel processes and $T_{parallel}$ represent the reduced query time with parallel processing.

$$T_{parallel} = \frac{T_q}{N_p} \quad (61)$$

In Eq.(61) where, Increasing N_p shortens $T_{parallel}$, enabling efficient data retrieval for high-demand queries. This parallel approach resembles the Pronghorn's ability to adapt its movements based on multiple stimuli, responding with rapid and coordinated actions to maintain efficiency.

Optimization also includes adjusting the query plan representing the sequence in which database operations are executed. In Eq.(62), where the query plan cost C_{plan}, depending on the estimated time for each operation T_{op} and the number of operations O_n.

$$C_{plan} = \sum_{i=1}^{O_n} T_{op,i} \quad (62)$$

Reducing O_n through an optimized query plan lowers C_{plan}, allowing faster query completion. This focus on a streamlined path parallels the Pronghorn's instinct to follow efficient routes, avoiding unnecessary energy expenditure while maintaining speed. In Focus Query Optimization, query efficiency is achieved by reducing retrieval time, enhancing selectivity, minimizing joins, implementing caching, and leveraging parallel processing.

3.8. Pool Sprint Connections

Pool Sprint Connections in the PSOF enhances connection management by reusing established connections, reducing the time and resources needed to develop new ones. This approach minimizes latency and improves overall response times by pooling and optimizing existing connections, emulating the Pronghorn's swift, energy-efficient sprints that conserve resources over distances. By reusing connections, Pool Sprint Connections enables rapid, continuous data exchanges that maintain service agility. In Pool Sprint Connections, the total connection time T_c for each request depends on the setup time T_{setup} and transmission time T_{trans} expressed mathematically in Eq.(63).

$$T_c = T_{setup} + T_{trans} \quad (63)$$

Minimizing T_{setup} by reusing existing connections directly reduces T_c, enabling faster response times for repeated requests. This method aligns with the Pronghorn's sustaining speed without repeatedly exerting energy, as the Pronghorn conserves its resources through efficient and direct paths.

The number of connections N_c pooled at any time must be balanced to prevent server overload. Let C_{maz} denote the maximum allowable connections. The pooling condition is defined as Eq.(64).

$$N_c \leq C_{max} \quad (64)$$

Maintaining $N_c \leq C_{max}$ ensures optimal performance by avoiding excessive server load. By managing the active connections within these limits, Pool Sprint Connections resembles the Pronghorn's instinct to regulate speed and conserve energy, adapting to its environment without overtaxing its capacity.

Pooling efficiency E_p in Pool Sprint Connections measures the reduction in setup time for each connection relative to the total number of requests R_t. The formula for pooling efficiency is represented mathematically in Eq.(65).

$$E_p = \frac{(T_{setup} \times R_t) - (T_{reuse} \times N_r)}{T_{setup} \times R_t} \quad (65)$$

where T_{reuse}represents the reduced time when reusing connections and N_r the reused connections. A high E_p reflects efficient connection reuse, reducing latency for each request. This efficiency reflects the Pronghorn's optimized sprinting, where each movement is directed to minimize time spent on repetitive actions.

Bandwidth allocation B_a for pooled connections also plays a vital role in maintaining high-speed data transfers. For a pooled connection set $P_c$with bandwidth B_{max}, the total bandwidth requirement B_t must satisfy.

$$B_t = N_c \times B_a \leq B_{max} \quad (66)$$

where in Eq.(66), Balancing $B_t$within B_{max} ensures that each connection can transmit data without network congestion. By maintaining bandwidth efficiency, Pool Sprint Connections support high-speed communication, resembling the Pronghorn's intuitive balance of speed and energy conservation across challenging terrains.

Connection reuse rate R_r represents the frequency of reusing pooled connections and is critical for optimizing response times. For $R_t$requests and a reuse count N_r, the reuse rate R_r is determined in Eq.(67).

$$R_r = \frac{N_r}{R_t} \quad (67)$$

A high R_r indicates that connections are reused efficiently, minimizing setup times for new requests and reducing latency. This optimized reuse reflects the Pronghorn's ability to maintain high-speed movement by following familiar, efficient routes without frequent stops, ensuring uninterrupted progress.

Latency reduction L_r through connection pooling, the time saved in each data exchange is measured by reusing connections. For a connection setup time T_{setup} and reuse time T_{reuse}, latency reduction is calculated as shown in Eq.(68).

$$L_r = T_{setup} - T_{reuse} \quad (68)$$

Higher L_r values signify significant time savings, allowing faster data exchange. This latency reduction mirrors the Pronghorn's instinctive sprints, where obstacles are minimized, and speed is maintained, conserving effort while covering distances swiftly.

The connection life L_c defines the duration for which each pooled connection remains active, balancing reuse efficiency with resource management. For a connection expiration threshold E_{th}, the connection life condition is expressed mathematically in Eq.(69).

$$L_c \leq E_{th} \quad (69)$$

Maintaining $L_c \leq E_{th}$ensures that connections are periodically refreshed, preventing degradation in performance due to outdated connections. This management strategy echoes the Pronghorn's adaptive pace, calibrating each sprint to sustain energy without exhaustion.

Connection allocation efficiency A_e, representing the proportion of active to idle connections, further improving resource usage. Let N_a denote active connections and N_i idle connections. The allocation efficiency A_e isrepresented mathematically in Eq.(70).

$$A_e = \frac{N_a}{N_a + N_i} \quad (70)$$

High A_e values indicate that most connections are actively used, optimizing server performance and reducing resource wastage. This efficient allocation resembles the Pronghorn's instinct for purposeful movement, where energy is directed toward practical actions rather than idle exertion.

The throughput Th_c of pooled connections, which measures the data transmitted per unit of time across all active connections, is critical for maintaining high service levels. For data D transmitted and transmission time T, the throughput $Th_c$is calculated as expressed in Eq.(71).

$$Th_c = \frac{D}{T} \quad (71)$$

Higher Th_c values ensure efficient data handling, supporting rapid communication across networked applications. This optimized throughput echoes the Pronghorn's continuous movement, where speed is maintained without unnecessary pauses, enhancing overall performance. Pool Sprint Connections in PSOF leverages optimized connection reuse, bandwidth efficiency, and active connection management to reduce latency and maximize data transmission speed. By sustaining swift, continuous connections, Pool Sprint Connections emulates the

Pronghorn's ability to maintain speed with calculated, resourceful movement across distances. Pool Sprint Connections enhances system responsiveness and scalability through this approach, enabling seamless data handling across demanding environments.

3.9. Reactive Scaling Pulse

The reactive scaling pulse in the PSOF ensures dynamic resource allocation to meet fluctuating demand while maintaining performance and responsiveness. This scaling mechanism allows the system to expand or reduce resource availability as needed, preserving optimal operation during peak and low-traffic periods. Reflecting the Pronghorn's adaptive agility, Reactive Scaling Pulse provides quick, energy-efficient responses to environmental changes, ensuring that resources are deployed only when necessary, conserving system energy and optimizing load handling.

The rate of demand change, represented as R_d, influences the scaling decision. For an initial demand D_i and a change in demand ΔD over a period T, the rate of demand change is given by Eq.(72).

$$R_d = \frac{\Delta D}{T} \quad (72)$$

when $R_d$exceeds a predefined threshold T_h, the Reactive Scaling Pulse triggers additional resources. This adaptability mirrors the Pronghorn's automatic response to external pressures, where speed adjusts instantly to maintain optimal movement.

Scaling latency L_s, or the time taken to activate or deactivate resources, affects the system's ability to respond effectively to demand fluctuations. For a scaling activation time $T_a$and deactivation time T_d, the total scaling latency is calculated as shown in Eq.(73).

$$L_s = T_a + T_d \quad (73)$$

Reducing L_s ensures that the Reactive Scaling Pulse activates resources swiftly, enhancing responsiveness during high-demand periods. This efficiency reflects the Pronghorn's ability to react promptly, maintaining speed without delay when faced with varying terrain conditions.

The total resource allocation $R_t$at any time depends on the baseline resources $R_b$and the additional resources R_a deployed in response to demand surges. The equation for resource allocation is expressed in Eq.(74).

$$R_t = R_b + R_a \quad (74)$$

Reactive Scaling Pulse maintains R_t at optimal levels, ensuring the system handles incoming load efficiently. This controlled increase and decrease of resources mirrors the Pronghorn's capacity to modulate energy output as required, achieving sustained performance over extended sprints.

Elasticity E, a measure of the system's ability to scale resources up or down based on demand, is represented by the ratio of change in allocated resources ΔR to the shift in demand ΔD as shown in Eq.(75).

$$E = \frac{\Delta R}{\Delta D} \quad (75)$$

A high elasticity E ensures that the Reactive Scaling Pulse matches resource availability to demand closely, preventing under- or over-provisioning. This elasticity echoes the Pronghorn's agility, where immediate adjustments maintain balance and enable seamless movement across changing landscapes.

Resource utilization U_r measures the efficiency of resource deployment, calculated as the ratio of resources actively used R_u to the total allocated resources R_t.

$$U_r = \frac{R_u}{R_t} \quad (76)$$

In Eq.(76) where maintaining U_r near-optimal levels prevent resource wastage and ensure each active component contributes to performance, reflecting the Pronghorn's efficient energy allocation for rapid and precise movements.

Cost efficiency C_e in Reactive Scaling Pulse evaluates the financial impact of scaling actions, which is essential for cost-sensitive applications. For scaling cost $C_s$and the total number of resources N_r, cost efficiency is represented as Eq.(77).

$$C_e = \frac{N_r}{C_s} \quad (77)$$

By maximizing C_e, Reactive Scaling Pulse minimizes expenses associated with resource allocation, ensuring a balance between performance and cost. This cost-conscious approach mirrors the Pronghorn's instinctive conservation of energy, where effort is expended judiciously to achieve sustained speed.

The scaling threshold T_{scale}, a predetermined limit that triggers resource changes

prevents unnecessary scaling actions. For baseline demand D_b and threshold demand D_{th}, the scaling condition is represented mathematically in Eq.(78).

$$D_b \geq D_{th} \quad (78)$$

This condition ensures that the Reactive Scaling Pulse activates only during significant demand shifts, maintaining system stability. This selective activation reflects the Pronghorn's adaptive instincts, where rapid responses are conserved for crucial moments, avoiding unnecessary expenditure of resources.

The response time T_r Scaling up or scaling down events impacts overall performance, particularly during rapid demand fluctuations. For time intervals T_{up} and T_{down} during scaling, the total response time is expressed mathematically in Eq.(79).

$$T_r = T_{up} + T_{down} \quad (79)$$

Lower T_r values ensure a timely reaction to demand changes, supporting uninterrupted service and reflecting the Pronghorn's swift adjustment to external stimuli, maintaining agility over varying distances.

Load balancing efficiency L_b in Reactive Scaling Pulse assesses the effectiveness of distributing incoming requests across scaled resources. Let R_{total} represent total incoming requests and $R_{distributed}$ the requests managed by scaled resources. Load balancing efficiency is given as expressed mathematically in Eq.(80).

$$L_b = \frac{R_{distributed}}{R_{total}} \quad (80)$$

Maximizing L_b supports smooth distribution of demand, preventing overload on individual resources and sustaining optimal performance. This efficiency mirrors the Pronghorn's strategic pacing, where energy is distributed evenly to maintain endurance and speed over challenging terrain. Reactive Scaling Pulse's ability to dynamically scale resources based on demand, manage costs, and optimize resource usage aligns with Pronghorn's instinct for agility and conservation. By balancing these factors, Reactive Scaling Pulse ensures that resources are allocated effectively, maintaining high performance and responsiveness across fluctuating workloads.

4. SIMULATION SETTING AND PARAMETERS

Web services enable seamless application communication using standardized protocols such as HTTP/2, SOAP, and REST. These services form the backbone of distributed systems by allowing platforms to interact efficiently. Evaluating the performance and scalability of web services under varying conditions is critical, particularly for frameworks like the PSOF, which aims to enhance latency reduction, scalability, and resource efficiency. Simulation is pivotal in testing such frameworks by providing a controlled, repeatable environment. The NS-3 simulator offers an ideal platform for evaluating PSOF's performance. Designed for network simulation, NS-3 models application behaviours, traffic patterns, and protocol performance under dynamic conditions. NS-3 helps assess how PSOF components interact and optimize web service efficiency by simulating real-world workloads and network topologies. Below is the simulation setting table for PSOF evaluation in NS-3.

Table 1. Simulation Settings.

Parameter	Value
Simulation Time	600 seconds
Network Type	Hybrid (Wired and Wireless)
Number of Nodes	50
Application Traffic	HTTP/2, REST API Calls
Data Payload Size	128 KB
Bandwidth	100 Mbps
Propagation Delay	2 ms
Queue Size	100 packets
Mobility Model	Random Waypoint
Cache Size per Node	500 MB
Edge Server Placement	Distributed (3 Edge Nodes)
Protocol Stack	TCP/IP with QUIC

This configuration enables precise testing of PSOF's components like load balancing, caching, and edge computing, replicating real-world traffic dynamics and scalability challenges.

5. RESULTS AND DISCUSSIONS

The evaluation of the PSOF in NS-3 highlights its effectiveness compared to Edge-X and ARTP across different node densities. The Packet

Delivery Ratio (PDR), defined as the percentage of successfully delivered packets to the total packets sent, serves as the primary metric for analysis. As node density increases from 50 to 500 nodes, PSOF consistently achieves higher PDR values, with an average of 71.945%, outperforming Edge-X and ARTP, which record averages of 46.985% and 56.209%, respectively. For 50 nodes, PSOF achieves the highest PDR of 78.33%. In contrast, Edge-X and ARTP deliver lower ratios of 58.40% and 64.94%, highlighting the superior resource allocation and load-balancing capabilities of PSOF under lower traffic loads.

The PDR for all frameworks declines due to congestion and resource competition when node density is increased. At 500 nodes, PSOF maintains a competitive PDR of 65.67%, compared to the significantly lower values of 34.12% and 47.31% for Edge-X and ARTP. This stability demonstrates PSOF's scalability and ability to manage high-traffic volumes effectively. The superior performance of PSOF can be attributed to its optimized components, such as Quick Cache Recall, Edge Sprinting, and Reactive Scaling Pulse, which reduce latency, improve routing efficiency, and balance server loads dynamically. These enhancements ensure consistent data delivery across varying network conditions, emphasizing PSOF's potential for improving web service reliability and scalability in distributed environments.

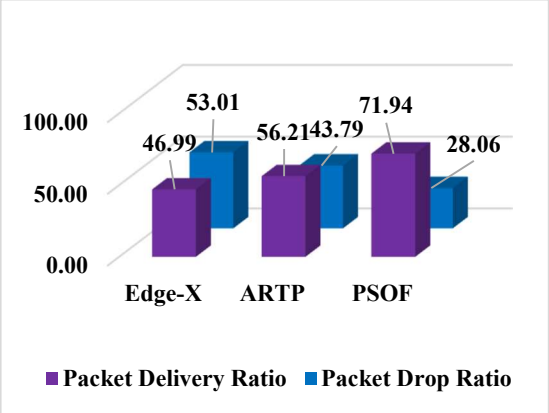


Fig. 1. Packet Delivery and Packet Drop

Fig. 1 highlights its effectiveness in minimizing the Packet Drop Ratio (PDrR). PDrR is the percentage of data packets that fail to reach their destination relative to the total packets sent. Lower PDrR indicates better network reliability and efficiency.PSOF consistently outperforms Edge-X and ARTP by maintaining a significantly lower PDrR across all node densities. With 50 nodes, PSOF achieves a PDrR of 21.67%, compared to 41.60% for Edge-X and 35.06% for ARTP. This demonstrates the robust packet management strategies of PSOF under light traffic conditions.

As node density increases, PDrR rises for all frameworks due to congestion and resource contention. At 500 nodes, PSOF maintains a PDrR of 34.33%, significantly lower than the 65.88% recorded by Edge-X and 52.69% by ARTP. The average PDrR for PSOF across all scenarios is 28.055%, showcasing its ability to maintain efficient packet delivery even under high network loads. Edge-X and ARTP record averages of 53.015% and 43.791%, respectively. The improvements achieved by PSOF can be attributed to its optimized mechanisms, such as Reactive Scaling Pulse for dynamic resource allocation and Quick Cache Recall for enhanced data accessibility, which reduces packet loss during congestion. These results emphasize the capability of PSOF to ensure reliable data transmission in distributed environments.

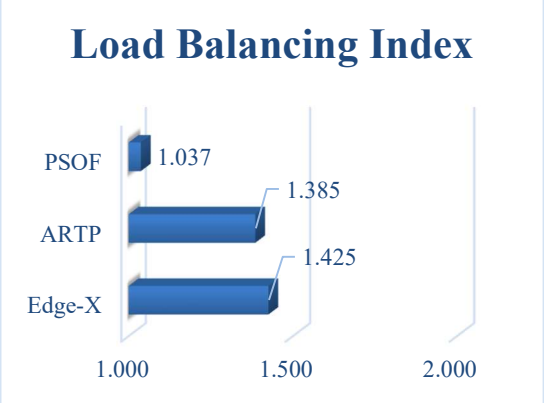
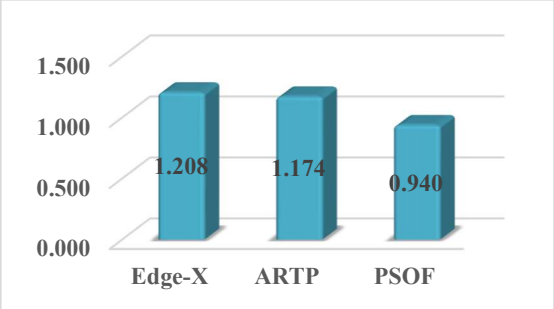


Fig. 2. Load Balancing Index

The evaluation of the PSOF in terms of the Load Balancing Index (LBI) demonstrates its efficiency compared to Edge-X and ARTP. LBI is defined as a measure of the uniformity in resource utilization across servers, with lower values indicating better load distribution. Fig. 2. Illustrates the LBI pictorially. PSOF achieves an LBI of 1.037, significantly outperforming Edge-X (1.425) and ARTP (1.385). The reduced LBI of PSOF highlights its ability to balance workloads more effectively, minimizing resource bottlenecks and enhancing system stability.

The superior performance of PSOF can be attributed to its optimized mechanisms, such as Steady Pace Load Balancing and Reactive Scaling Pulse. These components dynamically distribute incoming requests and adjust resource allocation based on real-time traffic, ensuring that server loads remain uniform. The higher LBI values recorded for Edge-X and ARTP suggest uneven resource usage, leading to overburdened servers and reduced overall system efficiency. PSOF's consistent load distribution ensures minimal latency, enhanced scalability, and improved user experience, making it a robust solution for managing dynamic web service environments.

Path Optimality measures how efficiently data packets traverse the network from source to destination, with lower values indicating more efficient and optimal paths. Fig. 3. Exhibits the outcome of PSOF in terms of Path Optimality.



PSOF consistently achieves better path optimality, maintaining an average value of 0.940, significantly outperforming Edge-X (1.208) and ARTP (1.174). At 50 nodes, PSOF records the lowest value of 0.900, compared to 1.05 for Edge-X and 1.03 for ARTP. This result highlights PSOF's ability to deliver highly efficient routing paths under light network loads. As the number of nodes increases, path optimality decreases slightly for all frameworks, reflecting the increased complexity of managing higher node densities. However, PSOF consistently achieves better performance. At 500 nodes, PSOF records a path optimality of 0.988, while Edge-X and ARTP record 1.4 and 1.35, respectively. This improvement is attributed to PSOF's advanced mechanisms, such as Leap Serialization and Focus Query Optimization, which streamline data transmission and ensure efficient path selection. The superior performance of PSOF in achieving optimal paths enhances network reliability, minimizes congestion, and reduces overall latency, demonstrating its robustness in large-scale and dynamic environments.

Throughput is the rate at which data packets are successfully processed and transmitted by the system, typically measured in bits per second or packets per second. Higher throughput indicates better system efficiency and resource utilization. The obtained throughput values of PSOF are depicted pictorially in Fig. 4.

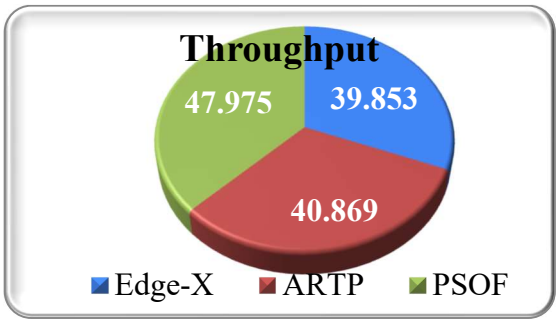


Fig. 4. Throughput

PSOF achieves an average throughput of 47.975, significantly surpassing Edge-X (39.853) and ARTP (40.869). At 50 nodes, PSOF records a throughput of 41.53, outperforming Edge-X (35.09) and ARTP (35.78). This demonstrates PSOF's ability to handle light traffic loads efficiently while maintaining a higher data transfer rate. As the network size increases, throughput values for all frameworks improve due to the availability of additional nodes to handle traffic. However, PSOF consistently delivers the highest throughput. At 500 nodes, PSOF achieves a throughput of 55.25, compared to 45.14 for Edge-X and 46.55 for ARTP. This improvement can be attributed to PSOF's components like Steady Pace Load Balancing and Reactive Scaling Pulse, which dynamically distribute workloads and allocate resources to optimize data transfer.

6. CONCLUSION

The proposed work, PSOF, has proven to be a robust solution for enhancing web services' performance, scalability, and efficiency. By drawing inspiration from the Pronghorn's speed, agility, and adaptability, PSOF integrates carefully designed steps to address critical challenges such as latency reduction, load balancing, resource optimization, and fault tolerance. Each step, from Sprint Compression to Reactive Scaling Pulse, creates a cohesive framework that improves web service operations under varying conditions. Simulation results have demonstrated that PSOF consistently outperforms frameworks like Edge-X and ARTP across multiple performance metrics, including packet delivery ratio, load balancing index, path optimality, and throughput. PSOF's superior performance is attributed to its

unique ability to adapt dynamically to fluctuating traffic loads while maintaining efficient resource allocation and routing precision. The framework effectively reduces latency, minimizes packet loss, and distributes workloads evenly, ensuring seamless operations even in high-density network environments. The scalability and reliability of PSOF make it an ideal choice for large-scale, distributed web services. By leveraging mechanisms such as Quick Cache Recall for rapid data access, Edge Sprinting for reduced data travel times, and Steady Pace Load Balancing for equitable resource utilization, PSOF addresses modern web service demands effectively. The framework's demonstrated capability to enhance performance across diverse scenarios highlights its potential to set a new benchmark for optimized web service architecture.

REFERENCES:

[1]. V. Lange and H. Daduna, "The Weber problem in logistic and services networks under congestion," *EURO Journal on Computational Optimization*, vol. 11, p. 100056, 2023, doi: https://doi.org/10.1016/j.ejco.2022.100056.

[2]. A. García-Domínguez, F. Palomo-Lozano, I. Medina-Bulo, A. Ibias, and M. Núñez, "Computing performance requirements for web service compositions," *Comput Stand Interfaces*, vol. 83, p. 103664, 2023, doi: https://doi.org/10.1016/j.csi.2022.103664.

[3]. Q. Liu, L. Wang, S. Du, and B. J. Van Wyk, "A Method to Enhance Web Service Clustering by Integrating Label-Enhanced Functional Semantics and Service Collaboration," *IEEE Access*, vol. 12, pp. 61301–61311, 2024, doi: 10.1109/ACCESS.2024.3392607.

[4]. R. Herrero, "REST and EDA architectures in IoT actuation," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 205–212, 2023, doi: https://doi.org/10.1016/j.iotcps.2023.05.002.

[5]. . M. Ruiz Ródenas, J. Pastor-Galindo, and F. Gómez Mármol, "A general and modular framework for dark web analysis," *Cluster Comput*, vol. 27, no. 4, pp. 4687–4703, 2024, doi: 10.1007/s10586-023-04189-2.

[6]. G. Ortiz *et al.*, "A microservice architecture for real-time IoT data processing: A reusable Web of things approach for smart ports," *Comput Stand Interfaces*, vol. 81, p. 103604, 2022, doi: https://doi.org/10.1016/j.csi.2021.103604.

[7]. S. Chickerur, S. Balannavar, P. Hongekar, A. Prerna, and S. Jituri, "WebGL vs. WebGPU: A Performance Analysis for Web 3.0," *Procedia Comput Sci*, vol. 233, pp. 919–928, 2024, doi: https://doi.org/10.1016/j.procs.2024.03.281.

[8]. A. Laadhar*et al.*, "Web of Things Semantic Interoperability in Smart Buildings," *Procedia Comput Sci*, vol. 207, pp. 997–1006, 2022, doi: https://doi.org/10.1016/j.procs.2022.09.155.

[9]. V. S. Borkar, S. Choudhary, V. K. Gupta, and G. S. Kasbekar, "Scheduling in wireless networks with spatial reuse of spectrum as restless bandits," *Performance Evaluation*, vol. 149–150, p. 102208, 2021, doi: https://doi.org/10.1016/j.peva.2021.102208.

[10]. F. Freitas, A. Ferreira, and J. Cunha, "A methodology for refactoring ORM-based monolithic web applications into microservices," *J Comput Lang*, vol. 75, p. 101205, 2023, doi: https://doi.org/10.1016/j.cola.2023.101205.

[11]. P. S. S. K. Gandikota, D. Valluri, S. B. Mundru, G. K. Yanala, and S. Sushaini, "Web Application Security through Comprehensive Vulnerability Assessment," *Procedia Comput Sci*, vol. 230, pp. 168–182, 2023, doi: https://doi.org/10.1016/j.procs.2023.12.072.

[12]. R. Härting, C. Reichstein, L. Bühler, A. Gugel, and K. Winter, "Success Factors of Web Portals for Financial Forecasting," *Procedia Comput Sci*, vol. 207, pp. 2154–2161, 2022, doi: https://doi.org/10.1016/j.procs.2022.09.275.

[13]. L. Coote *et al.*, "An early economic evaluation of Kooth, a web-based mental health platform for children and young people with emerging mental health needs," *Internet Interv*, vol. 36, p. 100748,2024,doi: https://doi.org/10.1016/j.invent.2024.100748.

[14]. D. Felício, J. Simão, and N. Datia, "RapiTest: Continuous Black-Box Testing of RESTful Web APIs," *Procedia Comput Sci*, vol. 219, pp. 537–545,2023,doi: https://doi.org/10.1016/j.procs.2023.01.322.

[15]. L. C. P. Velasco, M. X. D. Rentucan, J. J. M. Largo, and N. A. M. Racaza, "A web-based market validation tool for the modified Startup Business Company Validation Methodology," *Procedia Comput Sci*, vol. 234, pp. 937–945, 2024,doi: https://doi.org/10.1016/j.procs.2024.03.082.

[16]. A. R. Svaigen, A. Boukerche, L. B. Ruiz, and A. A. F. Loureiro, "BioMixD: A Bio-Inspired and Traffic-Aware Mix Zone Placement Strategy for Location Privacy on the Internet of Drones," *Comput Commun*, vol. 195, pp. 111–123, 2022, doi:

https://doi.org/10.1016/j.comcom.2022.07.012.

[17]. R. Karthikeyan and R. Vadivel, "Boosted Mutated Corona Virus Optimization Routing Protocol (BMCVORP) for Reliable Data Transmission with Efficient Energy Utilization," *Wirel Pers Commun*, 2024, doi: 10.1007/s11277-024-11155-7.

[18]. J. Ramkumar, A. Senthilkumar, M. Lingaraj, R. Karthikeyan, and L. Santhi, "Optimal Approach for Minimizing Delays in Iot-Based Quantum Wireless Sensor Networks Using Nm-Leach Routing Protocol," *J Theor Appl Inf Technol*, vol. 102, no. 3, pp. 1099–1111, 2024.

[19]. D. Jayaraj, J. Ramkumar, M. Lingaraj, and B. Sureshkumar, "AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network," *International Journal of Computer Networks and Applications*, vol. 10, no. 1, pp. 119–129, 2023, doi: 10.22247/ijcna/2023/218516.

[20]. J. Ramkumar, R. Vadivel, and B. Narasimhan, "Constrained Cuckoo Search Optimization Based Protocol for Routing in Cloud Network," *International Journal of Computer Networks and Applications*, vol. 8, no. 6, pp. 795–803, 2021, doi: 10.22247/ijcna/2021/210727.

[21]. M. Allegretta, G. Siracusano, R. González, M. Gramaglia, and J. Caballero, "Web of shadows: Investigating malware abuse of internet services," *ComputSecur*, vol. 149, p. 104182, 2025, doi: https://doi.org/10.1016/j.cose.2024.104182.

[22]. E. Lehtonen *et al.*, "An open web-based GIS service for biomass data in Finland," *Environmental Modelling & Software*, vol. 176, p.105972,2024,doi: https://doi.org/10.1016/j.envsoft.2024.105972.

[23]. M. García-Torres *et al.*, "Feature selection applied to QoS/QoEmodeling on video and web-based mobile data services: An ordinal approach," *Comput Commun*, vol. 217, pp. 230–245, 2024, doi: https://doi.org/10.1016/j.comcom.2024.02.004.

[24]. L. S. KONE TAPSOBA, Y. TRAORE, and S. MALO, "Interoperability approach for Hospital Information Systems based on the composition of web services," *Procedia Comput Sci*, vol. 219, pp. 1161–1168, 2023, doi: https://doi.org/10.1016/j.procs.2023.01.397.

[25]. S. K. Ayfan, D. Al-Shammary, A. M. Mahdi, and F. Sufi, "Dynamic clustering based on Minkowski similarity for web services aggregation," *International Journal of Information Technology*, vol. 16, no. 8, pp. 5183–5194, 2024, doi: 10.1007/s41870-024-02174-5.

[26]. F. Song, B. Wang, X. Xie, R. Pu, Q. Zhang, and W. Wang, "APIRec: deep knowledge and diversity-aware web API recommendation," *Service Oriented Computing and Applications*, 2024, doi: 10.1007/s11761-024-00427-6.

[27]. C. Brandi, G. Perrone, and S. Pietro Romano, "Sniping at web applications to discover input-handling vulnerabilities," *Journal of Computer Virology and Hacking Techniques*, vol. 20, no. 4, pp. 641–667, 2024, doi: 10.1007/s11416-024-00518-0.

[28]. T. Kärpänen, "Barriers to creating value with cognitive accessibility features in digital services," *Univers Access Inf Soc*, 2024, doi: 10.1007/s10209-024-01151-w.

[29]. J. Ara, C. Sik-Lanyi, A. Kelemen, and T. Guzsvinecz, "An inclusive framework for automated web content accessibility evaluation," *Univers Access Inf Soc*, 2024, doi: 10.1007/s10209-024-01164-5.

[30]. M. Ragab *et al.*, "ESPRESSO: A Framework to Empower Search on the Decentralized Web," *Data Sci Eng*, vol. 9, no. 4, pp. 431–448, 2024, doi: 10.1007/s41019-024-00263-w.

[31]. Z. Jia, Y. Fan, J. Zhang, X. Wu, C. Wei, and R. Yan, "A Multi-Source Information Graph-Based Web Service Recommendation Framework for a Web Service Ecosystem," *Journal of Web Engineering*, vol. 21, no. 8, pp. 2287–2312, 2022, doi: 10.13052/jwe1540-9589.2183.

[32]. M. N. Bonab, J. Tanha, and M. Masdari, "A Semi-Supervised Learning Approach to Quality-Based Web Service Classification," *IEEE Access*, vol. 12, pp. 50489–50503, 2024, doi: 10.1109/ACCESS.2024.3385341.

[33]. A. Dhulfiqar, M. A. Abdala, N. Pataki, and M. Tejfel, "Deploying a web service application on the EdgeX open edge server: An evaluation of its viability for IoT services," *Procedia Comput Sci*, vol. 235, pp. 852–862, 2024, doi: https://doi.org/10.1016/j.procs.2024.04.081.

[34]. P. Krishna Kishore, S. Ramamoorthy, and V. N. Rajavarman, "ARTP: Anomaly based real time prevention of Distributed Denial of Service attacks on the web using machine learning approach," *International Journal of*

Intelligent Networks, vol. 4, pp. 38–45, 2023, doi: https://doi.org/10.1016/j.ijin.2022.12.001.

[35]. R. Jaganathan, S. Mehta, and R. Krishan, "Preface," Intell. Decis. Mak. Through Bio-Inspired Optim., pp. xiii–xvi, 2024, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85192858710&partnerID=40&md5=f8f1079e8772bd424d2cdd979e5f2710

[36]. S. P. Geetha, N. M. S. Sundari, J. Ramkumar, and R. Karthikeyan, "ENERGY EFFICIENT ROUTING IN QUANTUM FLYING AD HOC NETWORK (Q-FANET) USING MAMDANI FUZZY INFERENCE ENHANCED DIJKSTRA'S ALGORITHM (MFI-EDA)," J. Theor. Appl. Inf. Technol., vol. 102, no. 9, pp. 3708–3724, 2024, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85197297302&partnerID=40&md5=72d51668bee6239f09a59d2694df67d6

[37]. J. Ramkumar and R. Vadivel, "Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network," Int. J. Comput. Networks Appl., vol. 8, no. 4, pp. 455–464, 2021, doi: 10.22247/ijcna/2021/209711.

[38]. R. Vadivel and J. Ramkumar, "QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications," in Incorporating the Internet of Things in Healthcare Applications and Wearable Devices, IGI Global, 2019, pp. 109–121. doi: 10.4018/978-1-7998-1090-2.ch006.

[39]. J. Ramkumar, K. S. Jeen Marseline, and D. R. Medhunhashini, "Relentless Firefly Optimization-Based Routing Protocol (RFORP) for Securing Fintech Data in IoT-Based Ad-Hoc Networks," Int. J. Comput. Networks Appl., vol. 10, no. 4, pp. 668–687, 2023, doi: 10.22247/ijcna/2023/223319.

[40]. R. Jaganathan and V. Ramasamy, "Performance modeling of bio-inspired routing protocols in Cognitive Radio Ad Hoc Network to reduce end-to-end delay," Int. J. Intell. Eng. Syst., vol. 12, no. 1, pp. 221–231, 2019, doi: 10.22266/IJIES2019.0228.22.

[41]. R. Jaganathan, S. Mehta, and R. Krishan, Bio-Inspired intelligence for smart decision-making. IGI Global, 2024. doi: 10.4018/9798369352762.

[42]. M. P. Swapna, J. Ramkumar, and R. Karthikeyan, "Energy-Aware Reliable Routing with Blockchain Security for Heterogeneous Wireless Sensor Networks," in Lecture Notes in Networks and Systems, V. Goar, M. Kuri, R. Kumar, and T. Senjyu, Eds., Springer Science and Business Media Deutschland GmbH, 2025, pp. 713–723. doi: 10.1007/978-981-97-6106-7_43.

[43]. J. Ramkumar and R. Vadivel, "CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks," in Advances in Intelligent Systems and Computing, D. P. Mohapatra and H. S. Behera, Eds., Springer Verlag, 2017, pp. 145–153. doi: 10.1007/978-981-10-3874-7_14.

[44]. R. Jaganathan, S. Mehta, and R. Krishan, "Preface," Bio-Inspired Intell. Smart Decis., pp. xix–xx, 2024, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85195725049&partnerID=40&md5=7a2aa7adc005662eebc12ef82e3bd19f

[45]. J. Ramkumar, R. Karthikeyan, and M. Lingaraj, "Optimizing IoT-Based Quantum Wireless Sensor Networks Using NM-TEEN Fusion of Energy Efficiency and Systematic Governance," in Lecture Notes in Electrical Engineering, V. Shrivastava, J. C. Bansal, and B. K. Panigrahi, Eds., Springer Science and Business Media Deutschland GmbH, 2025, pp. 141–153. doi: 10.1007/978-981-97-6710-6_12.

[46]. J. Ramkumar, C. Kumuthini, B. Narasimhan, and S. Boopalan, "Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol," in 2022 International Conference on Advanced Computing Technologies and Applications, ICACTA 2022, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICACTA54488.2022.9752899.

[47]. J. Ramkumar, R. Karthikeyan, and V. Valarmathi, "Alpine Swift Routing Protocol (ASRP) for Strategic Adaptive Connectivity Enhancement and Boosted Quality of Service in Drone Ad Hoc Network (DANET)," Int. J. Comput. Networks Appl., vol. 11, no. 5, pp. 726–748, 2024, doi: 10.22247/ijcna/2024/45.

[48]. L. Mani, S. Arumugam, and R. Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," in ACM International Conference Proceeding Series, Association for Computing Machinery, 2022. doi: 10.1145/3590837.3590907.

[49]. J. Ramkumar and R. Vadivel, "Improved Wolf prey inspired protocol for routing in cognitive radio Ad Hoc networks," Int. J. Comput. Networks Appl., vol. 7, no. 5, pp. 126–136, 2020, doi: 10.22247/ijcna/2020/202977.

[50]. M. Lingaraj, T. N. Sugumar, C. S. Felix, and J. Ramkumar, "Query aware routing protocol for mobility enabled wireless sensor network," Int. J. Comput. Networks Appl., vol. 8, no. 3, pp. 258–267, 2021, doi: 10.22247/ijcna/2021/209192.

[51]. K. S. J. Marseline, J. Ramkumar, and D. R. Medhunhashini, "Sophisticated Kalman Filtering-Based Neural Network for Analyzing Sentiments in Online Courses," in Smart Innovation, Systems and Technologies, A. K. Somani, A. Mundra, R. K. Gupta, S. Bhattacharya, and A. P. Mazumdar, Eds., Springer Science and Business Media Deutschland GmbH, 2024, pp. 345–358. doi: 10.1007/978-981-97-3690-4_26.

[52]. N. K. Ojha, A. Pandita, and J. Ramkumar, "Cyber security challenges and dark side of AI: Review and current status," in Demystifying the Dark Side of AI in Business, IGI Global, 2024, pp. 117–137. doi: 10.4018/979-8-3693-0724-3.ch007.

[53]. R. Jaganathan and R. Vadivel, "Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks," Int. J. Comput. Digit. Syst., vol. 10, no. 1, pp. 1063–1074, 2021, doi: 10.12785/ijcds/100196.

[54]. A. Senthilkumar, J. Ramkumar, M. Lingaraj, D. Jayaraj, and B. Sureshkumar, "Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing," Int. J. Comput. Networks Appl., vol. 10, no. 2, pp. 217–230, 2023, doi: 10.22247/ijcna/2023/220737.

[55]. R. Jaganathan, S. Mehta, and R. Krishan, Intelligent Decision Making Through Bio-Inspired Optimization. IGI Global, 2024. doi: 10.4018/979-8-3693-2073-0.

[56]. R. Karthikeyan and R. Vadivel, "Proficient Dazzling Crow Optimization Routing Protocol (PDCORP) for Effective Energy Administration in Wireless Sensor Networks," in IEEE International Conference on Electrical, Electronics, Communication and Computers, ELEXCOM 2023, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ELEXCOM58812.2023.10370559.

[57]. M. P. Swapna and J. Ramkumar, "Multiple Memory Image Instances Stratagem to Detect Fileless Malware," in Communications in Computer and Information Science, S. Rajagopal, K. Popat, D. Meva, and S. Bajeja, Eds., Springer Science and Business Media Deutschland GmbH, 2024, pp. 131–140. doi: 10.1007/978-3-031-59100-6_11.

[58]. [28] P. Menakadevi and J. Ramkumar, "Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data," in 2022 International Conference on Advanced Computing Technologies and Applications, ICACTA 2022, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICACTA54488.2022.9753203.

[59]. [29] J. Ramkumar and R. Vadivel, "Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks," Wirel. Pers. Commun., vol. 120, no. 2, pp. 887–909, 2021, doi: 10.1007/s11277-021-08495-z.