A Scalable Microkernel Inspired Architecture for Transparent and Adaptive Supply Chain Management

H. D. Vithanage
Faculty of Computing
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it21308284@my.sliit.lk

H. M. S. N. Dehipola
Faculty of Computing
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it21291678@my.sliit.lk

K.D.R. Manditha
Faculty of Computing
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it21289484@my.sliit.lk

K.S. Weedagamaarachchi
Faculty of Computing
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it21576966@my.sliit.lk

Vishan Jayasinghearachchi
Faculty of Computing
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
vishan.j@sliit.lk

Jeewaka Perera
Faculty of Computing
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
jeewaka.p@sliit.lk

***Abstract* - Supply chain management (SCM) is a critical aspect of modern industry, requiring efficiency, transparency, and adaptability. Traditional SCM architecture often struggles with scalability, fault tolerance, and seamless integration of emerging technologies, leading to inefficiencies in dynamic environments like coconut peat production. To address these challenges, this research proposes a scalable microkernel-inspired architecture that enhances modularity, fault isolation, and real-time adaptability. By leveraging a plugin-based system, the architecture allows dynamic workflow execution, ensuring flexibility, scalability, and resilience. The integration of IoT sensors enables real-time monitoring, while blockchain technology enhances traceability and transparency. Additionally, gRPC communication and K3S container orchestration improve efficiency and resource utilization.**

The proposed architecture was evaluated through Architectural Trade-off Analysis Method (ATAM), Cost-Benefit Analysis Method (CBAM), and runtime profiling, assessing performance, scalability, and adaptability. Results indicate that the system significantly improves availability, fault tolerance, and modularity, making it well-suited for distributed SCM applications

Keyword* - Microkernel Architecture, Supply Chain Management (SCM), Modular Plugin-Based Systems, Software Architecture

I. INTRODUCTION

Supply chain management (SCM) has evolved into a cornerstone of modern industry, demanding efficiency, transparency, and adaptability. In agriculture, particularly in coconut peat production, the need for sustainable practices and ethical sourcing has heightened the call for advanced technological interventions. Coconut peat, a versatile byproduct of the coconut industry, is a vital component in horticulture and other industrial applications due to its biodegradable and water-retentive properties. However, traditional supply chain systems often lack transparency, adaptability, and resilience, leading to inefficiencies that hinder their ability to meet growing global demand. As discussed in [1], modern supply chains must integrate technology-driven strategies to enhance visibility and responsiveness, ensuring seamless coordination across various stakeholders. Failure to adapt often results in inefficiencies, delays, and inability to meet dynamic market demands.

This study proposes novel architecture tailored for the coconut peat supply chain to address these challenges based on microkernel architecture. The approach separates core system functions from additional functionalities managed via dynamic plugins, allowing modularity, scalability, and adaptability. Integrating cutting-edge technologies such as smart sensors and blockchain, this architecture enhances real-time data capture and secure traceability, ensuring each stage of the supply chain is transparent and verifiable. The system is designed to dynamically load and manage plugins for key processes, meeting the unique and evolving requirements of diverse clients.

The proposed architecture's emphasis on high availability and dynamic plugin management ensures robustness even in complex and distributed environments. By isolating plugin failures and maintaining seamless inter-process communication, the microkernel system minimizes downtime and optimizes resource usage. Moreover, blockchain integration guarantees transparency and accountability, addressing increasing consumer demand for ethically and sustainably sourced products. This research aims to demonstrate how this novel architecture can revolutionize SCM by creating a transparent, scalable, and adaptive system tailored to modern industrial challenges.

This research plans to summarize the design, implementation, and evaluation of the proposed architecture. The findings showcase the system’s ability to dynamically manage workflows, integrate modern technologies, and maintain high performance under diverse operational conditions. This study not only advances the state-of-the-art in SCM technology but also sets the stage for broader applications of microkernel-based systems in other industries requiring modularity, adaptability, and transparency.

II. LITERATURE REVIEW

The proposed study explores a scalable microkernel-inspired architecture for transparent and adaptive supply chain management (SCM). This review synthesizes key insights from existing literature on microkernel systems, distributed architectures, and their applications in various domains,

including supply chain management, to establish the foundation for this novel approach.

The microkernel architecture or plugin architecture is characterized by its simple core which handles only the most important functions to operate the system. Such as inter-process communication (IPC) and basic scheduling, and load handling. While the other additional functions/ services run in user space as independent plugins. This separation is crucial for achieving high system availability as it sets apart failures to individual plugins, thus preventing a single failure from crashing into the whole system. Liedtke's seminal work on microkernel arguing for a design that minimizes the core footprint to improve performance and reliability [2].

System availability is an important metric in any system. It is the ability to remain operational and responsive. However, the dependency of the stable core may introduce risks. If the core fails, the entire system would be inoperable even if the plugin. Several studies have shown that microkernel architecture inherently supports high availability. For instance, the paper [3] highlights how the isolation system components in user space can prevent a system-wide failure. However, the study also shows the potential performance overhead introduced by frequent IPC which could impact system performance in high traffic environment. Recent advancements have made cloud-native technologies such as containerization and orchestration tools like Kubernetes to further improve the system availability. These technologies can be used to ensure even in heavy load or partial system failures, the system availability by managing service redundancy and traffic load distribution [4].

Management of the Plugins includes enabling the dynamic loading, execution and unloading in the architecture is crucial factor to consider as its directly related with process. This flexibility is very important for supply chain management where the workflow and process may frequently change due to varying client requirements. The process of loading, execution and unloading is managed through a well-defined plugin interface that ensures consistency across different plugins. In the paper [1] describes the difficulties arising in microkernel-based systems due to IPC between different processes, with overarching stress on the efficient messaging schemas. A Microkernel must ensure that the IPC system offered incorporates the dynamic aspect of plugin loading and unloading the IPC system must be scalable in a way that ensures seamless communication between the core and its plugins. The use of formal methods to verify plugin interactions with the core system is another crucial factor. This verifies that the plugins do not introduce inconsistencies or security vulnerabilities into whole system [5].

Microkernel principles have been successfully adapted for distributed systems, as seen in the DROPS (Dresden Real-Time Operating System) framework. DROPS integrates real-time and time-sharing components, enabling efficient communication and resource allocation across distributed nodes. This capability is crucial for applications like multi-robot systems, where real-time decision-making and fault tolerance are essential. DROPS highlights the viability of using microkernel foundations for scalable and distributed architecture [5].

Further supporting these principles, the Industrial IoT Monitoring Architecture proposed by Raposo et al. integrates layered software design for monitoring Wireless Sensor Networks (WSNs) in industrial applications. The architecture separates sensor node monitoring, network diagnostics, and management systems, ensuring low resource consumption and multi-standard compatibility. It employs active monitoring techniques to enhance system reliability and adaptability, like microkernel principles, where components operate independently without burdening the core system. The research validates this approach through a proof-of-concept implementation, demonstrating high efficiency with minimal energy overhead in Industrial 4.0 applications [8].

In the domain of socially assistive robotics, the HealthBot architecture, a three-layered robotic system for elderly care and healthcare automation. This system mirrors microkernel principles by separating robot behavior from execution, enabling rapid customization, iterative improvements, and enhanced fault isolation. The Behavior Execution Engine (BEE) processes Robot Behavior Descriptions (RBDs), allowing non-programmers, such as healthcare professionals, to modify robot interactions without altering the core software. This modular approach enhances system flexibility and has been successfully deployed in real-world field trials [9].

The literature demonstrates that microkernel architectures are well-suited for applications requiring modularity, scalability, and fault tolerance. By leveraging these principles, the proposed system aims to address critical inefficiencies in SCM, providing a transparent and adaptive framework for modern industrial challenges. This study builds on the strengths of existing microkernel-based systems and extends their application to supply chain processes, setting the stage for transformative advancements in the field.

III. METHODOLOGY

TABLE I: Architectural Trade-off Analysis

Attribute	Monolithic Architecture	Microkernel Architecture	Microservices Architecture
Modularity	Low	Medium	High
Scalability	Low to Medium	Medium	High
Maintainability	Low	Medium	High
System Availability	Medium	High	High
Performance	High	Medium	Medium
Complexity	Low	Medium	High
Deployment Flexibility	Low	Medium	High
Technology Flexibility	Low	Medium	High
Fault Isolation	Low	High	High
Development Speed	High	Medium	Medium
Technology Stack	Low	Medium	High
Security	Medium	High	High

[6,7]

To further analyze the effectiveness of the proposed novel architecture, a comparison is made with monolithic, microservices and microkernel architecture. The Architectural Trade-off Analysis Table (ATAT) presents an overview of key attributes such as modularity, scalability, maintainability, and system availability.

This comparison highlights the advantages of a microkernel-inspired architecture, demonstrating its balance between the monolithic approach (low modularity, low scalability) and microservices approach (high modularity, high complexity). The microkernel model provides moderate complexity with high scalability and flexibility, making it a suitable choice for supply chain management where adaptability and efficiency are essential.

By evaluating factors such as fault isolation, deployment flexibility, and system security, the ATAT supports the claim that novel architecture is optimal for integrating blockchain and IoT while maintaining high system availability. The ability to dynamically create plugins, scale individual components without affecting the entire system, and seamlessly integrate new IoT devices provides a significant advantage over traditional architectures.

This analysis reinforces the feasibility and effectiveness of using a microkernel-inspired architecture for supply chain management, ensuring transparency, adaptability, and high performance in dynamic operational environments.

A. System Design

The research adopts an experimental approach involving prototype development with future field testing. The proposed system is tested in the context of a coconut peat production supply chain, simulating real-world scenarios to validate its performance.

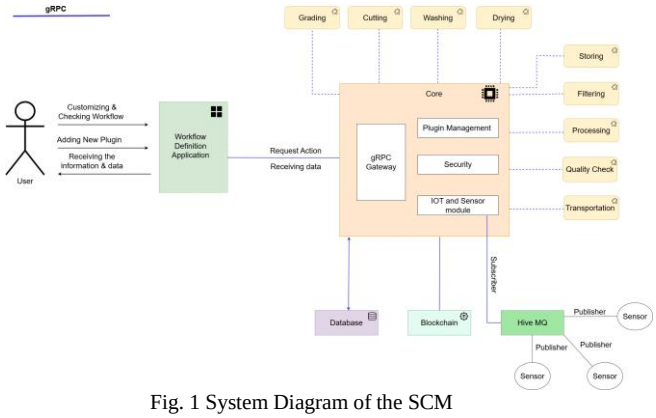

Fig. 1 System Diagram of the SCM

Fig.1 illustrates the novel architecture used in the coconut peat supply. Each plugin represents a step in the SCM and integrates with various vital components. The system includes the following key components:

1) Customization Workflow Application:

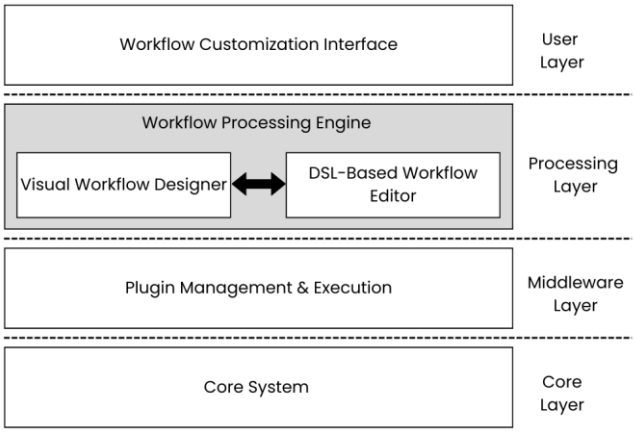

Fig. 2 Component Diagram of Workflow customization tool

The workflow customization tool allows users to design and modify manufacturing workflows through an intuitive drag-and-drop interface. Users can select predefined plugins, such as grading, cutting, washing, or drying, and place them on the workflow canvas. Each plugin represents a various process in the coconut peat supply chain and allows further customization based on specific requirements. In the grading process, for instance, users can adjust parameters like the acceptable color range for husks to be classified as "qualified" and “acceptable”. These features enable cocopeat domain experts to customize workflows without requiring technical expertise.

When creating a new plugin, users have two options. Users can create a new plugin either by figuring a flowchart or writing instructions in a domain-specific language (DSL) [11]. The plugin's logic is visually represented via the flowchart-based approach, which enables users to define steps and phases through an interactive user interface (UI). In the meantime, the DSL method allows for the accuracy of the plugin's functionality and supports users who prefer a structured text-based style. This adaptability ensures that both non-technical and technical users can effectively contribute to workflow customization.

The tool performs validation checks when a workflow is finalized, to ensure accuracy, consistency, and conformity to system limitations. After successful validation, the workflow is sent to the core system for execution. This structured approach implies that processes are reliable and flexible, enabling a simplified procedure for establishing supply chain actions while maintaining system integrity [12].

2) Blockchain Integration:

This paper explores methods to minimize gas costs in smart contracts by reducing code complexity. In blockchain networks like Ethereum, gas fees depend on computational resources used during execution, and high-complexity contracts result in higher costs. Recent research has demonstrated that frameworks can optimize contract execution without compromising security [13]. High-complexity contracts lead to higher costs, impacting efficiency and scalability. The study focuses on optimizing Solidity code, leveraging design patterns like proxy contracts, reducing storage operations, and minimizing loops. By streamlining contract logic, developers can enhance performance, lower transaction fees, and improve

user adoption, making decentralized applications more cost-effective and sustainable.

3) IoT Integration:

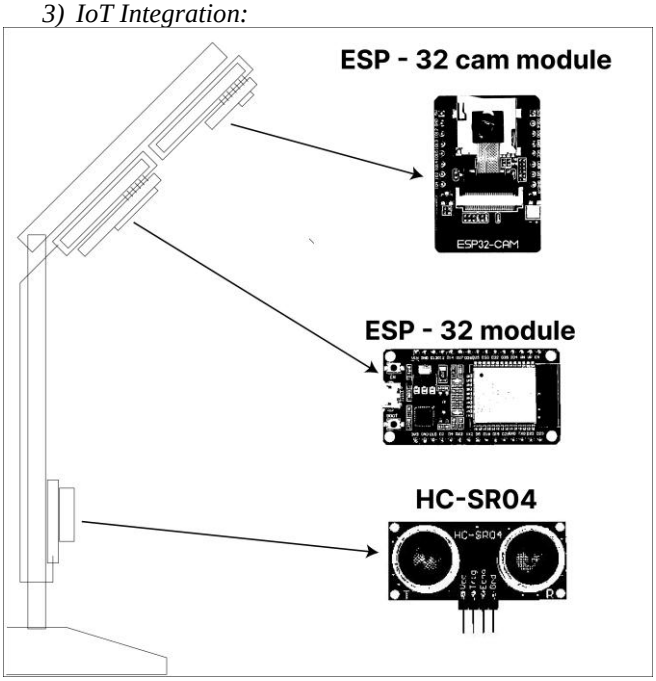

Fig. 3 Customized Smart Sensor Device

4) This paper presents a cost-effective approach for automated husk grading using optimized image processing algorithms deployed on two ESP32 microcontrollers. By leveraging a streamlined set of image processing techniques such as basic color thresholding and edge detection, the proposed system achieves accurate and efficient classification despite the limited memory capacity inherent to the ESP32. Furthermore, the choice of ESP32 as a low-cost development platform makes the solution accessible for small-scale and distributed applications, addressing the practical constraints of resource-constrained environments while maintaining reliable performance. Similar approaches in image processing techniques have been explored in fire detection systems, emphasizing the effectiveness of color and edge detection methods [14].

5) Containerization and Orchestration

The system leverages Docker and Kubernetes (K3S) to ensure high availability, fault tolerance, and scalability of plugins and core services. Docker enables containerized deployment, allowing each plugin to run in an isolated environment, reducing dependency conflicts and improving system stability. K3S, a lightweight Kubernetes distribution, orchestrates these containers, ensuring that plugins can be dynamically deployed, scaled, and managed with minimal resource overhead. In the event of a failure, Kubernetes automatically restarts affected containers to maintain system uptime. Additionally, new plugins can be created at runtime and seamlessly integrated into the existing workflow without system downtime, providing unparalleled flexibility in adapting to changing supply chain requirements. The ability to scale individual plugins independently optimizes resource usage, ensuring that workloads are distributed efficiently across available nodes. This approach guarantees that the architecture remains resilient and adaptable, capable of handling increasing workloads and integrating new functionalities without disrupting existing processes.

B. Evaluation metrics

The aimed novel architecture is evaluated using two key methods: static evaluation and runtime evaluation. These methods ensure a comprehensive assessment of the architecture's scalability, performance, fault tolerance, and resource efficiency in the coconut-peat supply chain management system.

1) Static Evaluation

The Cost-Benefit Analysis Method (CBAM) is a structured evaluation approach used in software architecture assessment to analyze trade-offs, balance cost and benefits, and optimize system design. It enables decision-makers to quantify the impact of architectural choices based on stakeholder priorities, system requirements, and long-term sustainability.

1. Identify Architectural Decisions (ADs)

TABLE II : Architectural Decisions

Architectural Decision (AD)	Description
Microkernel with Plugins	Enables modularity and runtime flexibility.
gRPC for Communication	Ensures fast and efficient data exchange between components.
MongoDB as Database	Provides scalability and NoSQL benefits.
Blockchain for Transparency	Enhances traceability in the supply chain.
IoT Integration	Captures real-time supply chain data.
K3S for Scalability	Uses Kubernetes lightweight distribution for scaling microkernel plugins.

This step involves listing key architectural choices that impact system performance, scalability, and adaptability. In this case, decisions such as using a novel architecture design, gRPC for communication, and Kubernetes for orchestration are identified.

2. Determine Quality Attributes to Measure

TABLE III: Quality Attributes to Measure

Quality Attribute	Why It Matters
Performance	How fast the system processes transactions.
Scalability	Ability to add/remove plugins without affecting the system.
Transparency	Trust and immutability of transactions via blockchain.
Flexibility	Ease of integrating new IoT devices.
Fault Tolerance	Ability to recover from plugin failures.
Resource Utilization	Efficiency of memory, CPU, and network.

Relevant system attributes such as performance, scalability, fault tolerance, flexibility, and resource utilization are selected. These attributes help evaluate how well the architecture meets stakeholder expectations. In this use case there are 3 stakeholders, manufacturer, exporter & customer.

3. Assign Utility Values to Quality Attributes

TABLE IV: Utility Values to Quality Attributes

Quality Attribute	Stakeholder Utility Score (1-5)
Performance	5
Scalability	4
Transparency	4
Flexibility	5
Fault Tolerance	3
Resource Utilization	3

Stakeholders assign importance scores to each quality attribute based on business and technical priorities. Higher values indicate attributes that provide the most benefit to the system and end-users

4. Assess Costs and Benefits of Architectural Decisions

TABLE V: Costs and Benefits of Architectural Decisions

Architectural Decision	Quality Attributes Affected	Cost (1-5)	Benefit (1-5)
Microkernel with Plugins	Scalability, Flexibility, Fault Tolerance	3	5
gRPC for Communication	Performance, Scalability	2	4
MongoDB as Database	Performance, Scalability	3	4
Blockchain for Transparency	Transparency, Security	4	5
IoT Integration	Performance, Flexibility	3	4
K3S for Scalability	Scalability, Resource Utilization	3	4

Each decision is analyzed for its implementation cost and the benefits it provides. Decisions with high benefits and low costs are preferred, ensuring an optimal balance between performance and investment.

5. Calculate Return on Investment (ROI)

TABLE VI: Return on Investment

Architectural Decision	Benefit	Cost	ROI Calculation	ROI Score
Microkernel with Plugins	5	3	(5-3)/3 = 0.67	0.67
gRPC for Communication	4	2	(4-2)/2 = 1.00	1.00
MongoDB as Database	4	3	(4-3)/3 = 0.33	0.33
Blockchain for Transparency	5	4	(5-4)/4 = 0.25	0.25
IoT Integration	4	3	(4-3)/3 = 0.33	0.33
K3S for Scalability	4	3	(4-3)/3 = 0.33	0.33

ROI is computed using the formula

$$ROI = (Benefit - Cost) / Cost$$

helping prioritize architectural decisions. A higher ROI indicates a decision that maximizes efficiency while minimizing expenses, ensuring a cost-effective and scalable solution.

2) Runtime Evaluation

To evaluate the architecture in the runtime, the following metrics will be analyzed:

6. Response Time will be measured using ghz and k6, assessing how quickly the system processes requests under different levels of concurrent users.
7. Transaction Throughput will be evaluated using ghz and k6 to determine the number of requests handled per second and the system's ability to handle increasing loads efficiently.
8. Fault Tolerance will be tested by simulating failures in individual plugins and tracking system recovery with Prometheus and Jaeger, which provide distributed tracing and real-time monitoring.
9. Resource Utilization will be analyzed using pprof for CPU and memory profiling, while Prometheus will collect live system metrics, with Grafana providing visualized insights into efficiency.
10. Scalability will be tested through Kubernetes (K3S) by deploying and managing multiple plugin instances, measuring the system's ability to dynamically scale workloads without performance degradation.

C. Use Case

The coconut-peat production supply chain is selected as the primary use case for testing and validating the proposed novel architecture. This supply chain provides a real-world, industry-specific scenario to assess the architecture's applicability, scalability, and efficiency in managing complex, dynamic workflows. The coconut-peat industry involves multiple stages, stakeholders, and evolving requirements, making it an ideal domain for evaluating a modular, transparent, and adaptive system. Coco-peat, also known as coir pith, is a versatile byproduct of the coconut industry, widely used in horticulture as a soil substitute and conditioner. It also serves as a dry oil and lubricant absorbent in industrial applications and as a high-water absorption material for animal bedding. Due to its high-water retention, aeration, and biodegradability, coco-peat has gained significant traction in recent years as an environmentally friendly agricultural input. Sri Lanka, recognized as the fourth-largest producer of coconuts globally, plays a crucial role in the coco-peat supply chain, processing a large portion of its 2.8 to 3 billion annual coconut production into value-added products such as coco-peat. The global market for coco-peat, valued at USD 2.27 billion in 2022, is projected to reach USD 3.8 billion by 2031, with a compound annual growth rate (CAGR) of 4.4% [9]. This rapid expansion highlights the increasing demand for sustainable agricultural solutions and underscores the importance of optimizing supply chain processes in the industry. Given the dynamic nature of coconut-peat processing, the proposed novel architecture ensures efficient workflow management, real-time adaptability, and high system availability. The system will be tested across critical supply chain stages such as grading, cutting, washing, drying, and packaging, integrating IoT sensors for real-time monitoring and blockchain technology for enhanced transparency and traceability. This structured approach

enables scalable, flexible, and data-driven decision-making, ensuring the supply chain remains competitive in the evolving global market

IV. Results

The evaluation of the Scalable Microkernel-Inspired Architecture for Transparent and Adaptive Supply Chain Management was conducted using static and runtime evaluation techniques. The study analyzed the architecture's scalability, performance, transparency, and adaptability through ATAM, CBAM, and runtime profiling tools. The CPU profile and performance report provided deeper insights into system efficiency, computational bottlenecks, and response times under different workloads.

A. Static Evaluation Results

1) Architectural Trade-off Analysis (ATAM)

The architecture was compared against monolithic, microservices, and microkernel architectures based on key attributes:

1. Modularity & Scalability: The microkernel approach balances modularity and scalability, allowing plugins to be loaded dynamically without affecting the core system.
2. Maintainability & System Availability: The isolation of plugins ensures that failures do not propagate to the core, enhancing fault tolerance.
3. Performance: The microkernel architecture incurs a slight performance overhead due to inter-process communication (IPC), but it remains efficient for distributed environments.
4. Deployment & Technology Flexibility: The architecture enables seamless integration of IoT sensors, gRPC-based communication, and blockchain transparency.

2) Cost-Benefit Analysis Method (CBAM)

The cost and benefits of different architectural decisions were analyzed. The return on investment (ROI) for each decision was calculated, highlighting the efficiency of gRPC for communication (ROI = 1.00) and the benefits of microkernel plugins for modular scalability (ROI = 0.67). Decisions like blockchain transparency and IoT integration were beneficial but had higher implementation costs.

TABLE VII: Cost-Benefit Analysis Method

Architectural Decision	Quality Attributes Affected	Cost	Benefit	ROI
Microkernel with Plugins	Scalability, Fault Tolerance	3	5	0.67
gRPC for Communication	Performance, Scalability	2	4	1.00
MongoDB as Database	Performance, Scalability	3	4	0.33
Blockchain for Transparency	Transparency, Security	4	5	0.25
IoT Integration	Performance, Flexibility	3	4	0.33
K3S for Scalability	Scalability, Resource Utilization	3	4	0.33

The gRPC implementation had the highest ROI due to its low cost and high-performance benefits. The plugins showed significant benefits for fault isolation and system adaptability, reinforcing the effectiveness of the chosen architecture.

B. Runtime Evaluation Results

1) Performance Metrics

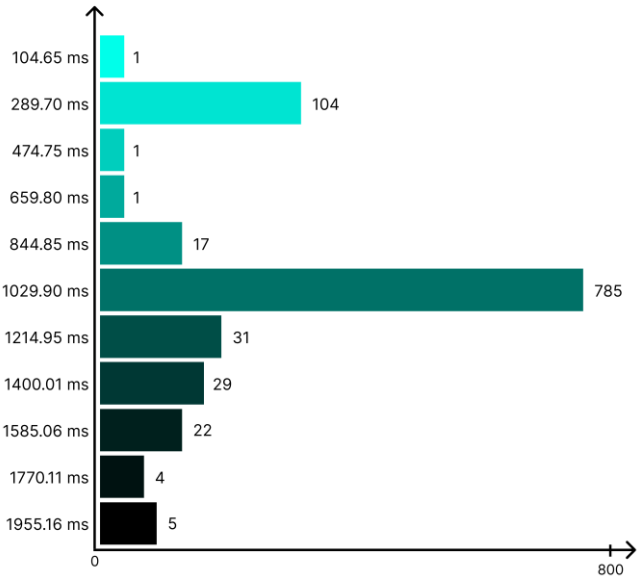

Fig. 4 Latency distribution chart

From the runtime profiling (Figure 4), the system's performance was evaluated based on response times, request throughput, and computational efficiency.

1. Total Execution Time: 18.69 seconds
2. Total Requests Processed: 1000
3. Slowest Response Time: 1.96 seconds
4. Fastest Response Time: 104.65 milliseconds
5. Average Latency: 913.82 milliseconds
6. Requests per Second: 53.52

The system handled 53.52 requests per second, showing moderate scalability, but latency distribution analysis revealed higher processing times in some cases. This suggests bottlenecks in execution, possibly due to inter-process communication overhead in the microkernel design.

2) Latency Distribution

Latency distribution showed 95% of requests completed under 1.27 seconds, but 1% exceeded 1.58 seconds, indicating some outlier delays. The following latency breakdown highlights system efficiency.

TABLE VIII: Latency Distribution

Percentile	Latency
10%	165.62 ms
25%	943.70 ms
50% (Median)	989.38 ms
75%	1.00 s
90%	1.02 s
95%	1.27 s
99%	1.58 s

The longer execution times in upper percentiles suggest that optimizations in inter-process communication (IPC) and plugin execution could enhance performance.

3) CPU Profiling Insights

TABLE IX: CPU profiling

Metric	Value
Total Profiling Duration	60.01 seconds
Total CPU Samples	440ms
Percentage of Total Runtime	0.73%
Top CPU Consumer Function	runtime.cgocall
CPU Time of Top Function	310ms
Percentage of CPU in Top Function	70.45%
Possible Cause	C-based function calls (MongoDB, gRPC, or external libraries)
Next Steps	Optimize database calls, disable gRPC compression, check for unnecessary C bindings

Fig. 5

he CPU profile (Figure 5) analysis revealed significant computational bottlenecks, with runtime.cgocall consuming 70.45% of the total execution time (310ms). This heavy reliance on CGo calls suggests that optimizing these interactions could enhance overall system performance. Additionally, the profile displayed a high depth in function call stacks, indicating potential inefficiencies in recursive executions or dependency-heavy operations that may be causing unnecessary computational overhead. Furthermore, frequent inter-process communication (IPC) interactions were observed, which could contribute to execution delays, aligning with the findings from the latency distribution analysis. To address these challenges, several optimizations can be explored, such as reducing CGo dependencies, refining IPC mechanisms, and restructuring execution workflows to enhance system responsiveness and improve efficiency under high workload conditions.

V. Discussion

A. Strengths of the Proposed Architecture

The proposed microkernel-inspired architecture offers several advantages that make it well-suited for a transparent and adaptive supply chain management system. One of its key strengths lies in its high modularity and adaptability. The plugin-based architecture allows for easy customization of supply chain workflows, enabling businesses to modify or expand their operations dynamically. The system supports real-time loading and unloading of plugins, ensuring that updates or modifications can be incorporated without disrupting ongoing processes. This flexibility is crucial in supply chain environments where market demands, and operational requirements frequently change.

Another significant strength of this architecture is its faulty isolation and high availability. Since each plugin operates independently of the core system, failures in individual plugins do not affect the overall system stability. This enhances fault tolerance, preventing localized failures from cascading into system-wide downtime. Additionally, the use of Kubernetes (K3S) ensures that services remain available even in the event of plugin failures. Kubernetes manages redundant services, automatically restarting failed components and dynamically reallocating resources to maintain system performance and resilience under varying workloads.

The scalability and integration capabilities of the proposed architecture further reinforce its effectiveness in supply chain management. The incorporation of IoT sensors allows for real-time monitoring of supply chain processes, providing accurate data for better decision-making. Additionally, blockchain integration enhances traceability and transparency, ensuring that supply chain transactions remain secure, immutable, and compliant with industry regulations. This transparency is particularly valuable for tracking goods, maintaining ethical sourcing standards, and ensuring accountability across the supply chain network.

B. Performance Limitations

Despite its strengths, the architecture exhibits certain performance limitations that require further optimization. One of the primary challenges is inter-process communication (IPC) overhead. Since the microkernel-based system relies heavily on IPC for communication between plugins and the core, there is a slight performance impact, particularly under high concurrent request loads. The need for frequent message passing between system components introduces processing delays, affecting response times. Optimizations in gRPC request handling and message-passing efficiency could help reduce IPC overhead and improve system responsiveness.

Another notable limitation is the heavy reliance on CGo function calls, which has been identified as a major computational bottleneck. The CPU profile analysis revealed that 70.45% of the total execution time was spent on runtime.cgocall, indicating that significant time is spent interfacing with C-based system functions. This reliance on CGo introduces latency and additional computational overhead, making it an area that requires improvement. Moving critical functions to native Go implementations or optimizing CGo interactions could significantly enhance execution efficiency.

Furthermore, latency variation in higher percentiles suggests that the system experiences performance inconsistencies under certain workloads. While most requests are handled efficiently, 1% of the requests exceeded 1.58 seconds, indicating potential bottlenecks in request scheduling and execution workflows. These outliers suggest that some requests may be queuing for longer durations, leading to processing inefficiencies. Addressing these inconsistencies is crucial for maintaining consistent and predictable system performance.

C. Potential Optimizations

To improve overall system performance, several optimizations can be implemented. One of the primary improvements would be to reduce reliance on CGo calls. Since a substantial portion of the execution time is spent on runtime.cgocall, rewriting critical components in native Go would eliminate costly transitions between Go and C

functions, improving execution speed and reducing computational overhead.

Another critical optimization involves improving IPC mechanisms. The current system relies on standard message-passing techniques, which introduce processing delays. Adopting zero-copy shared memory techniques could significantly reduce IPC overhead, allowing for faster data exchange between plugins and the core system. This approach would minimize unnecessary data copying and context switching, improving the efficiency of inter-process communication.

Finally, optimizing plugin execution workflows through parallel execution strategies could enhance performance under high workloads. Implementing asynchronous execution models for non-blocking plugin operations would reduce processing bottlenecks and ensure that the system scales efficiently as more plugins are dynamically added. Additionally, improving scheduling algorithms to prioritize time-sensitive tasks could help mitigate latency variations, ensuring a more consistent response time across all system operations.

By implementing these optimizations, the proposed microkernel-based supply chain management system can achieve higher efficiency, reduced latency, and improved scalability, making it a more robust and reliable solution for modern supply chain challenges.

VI. Conclusion

This research introduced scalable microkernel-inspired architecture tailored for transparent and adaptive supply chain management, particularly within the coconut peat production industry. By leveraging a modular, plugin-based design, the architecture provides high adaptability, fault isolation, and seamless scalability, making it well-suited for dynamic and distributed supply chain environments. The integration of IoT sensors ensures real-time monitoring, while blockchain technology enhances traceability and transparency, addressing key challenges in ethical sourcing and accountability. Additionally, the use of gRPC for efficient communication and K3S for lightweight orchestration further strengthens the system's ability to handle scalability and performance demands.

Through a comprehensive static and runtime evaluation, the architecture demonstrated notable advantages over traditional monolithic and microservices approaches. The Architectural Trade-off Analysis (ATAM) and Cost-Benefit Analysis Method (CBAM) confirmed that the microkernel model balances modularity and system complexity, ensuring both flexibility and maintainability. Performance profiling using CPU analysis and latency distribution tests provided deeper insights into bottlenecks and optimization opportunities, such as high inter-process communication (IPC) overhead and reliance on CGo function calls. While the system showed moderate scalability with an ability to process 53.52 requests per second, higher latencies in upper percentiles indicated potential execution inefficiencies. These findings highlight the need for further optimizations, such as reducing CGo dependencies, refining IPC mechanisms, and restructuring execution workflows.

Despite some performance limitations, the research successfully validates the feasibility and effectiveness of using a novel architecture which is inspired by microkernel architecture in supply chain management. By ensuring fault tolerance, adaptability, and high availability, the proposed model presents a viable alternative to conventional supply chain management systems. Future work should focus on enhancing execution efficiency, exploring parallel plugin execution strategies, and reducing computational overhead to further optimize the system's responsiveness and scalability. This study paves the way for broader adoption of novel architectures in real-world industrial applications, fostering a new era of transparent, efficient, and data-driven supply chain ecosystems.

References

[1] M. H. Hugos, *Essentials of Supply Chain Management*, 5th ed. Hoboken, NJ, USA: Wiley, 2024.

[2] J. Liedtke, "On Microkernel Construction," in Proc. SIGOPS '95, ACM, 1995, pp. 237-249. [Online]. Available: http://crossmark.crossref.org/dialog/?doi=10.1145%2F224057.224075&domain=pdf&date_stamp=1995 12-03. [Accessed: Aug. 20, 2024].

[3] Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," ACM Queue, vol. 14, no. 1, pp. 70-93, Jan.-Feb. 2016. [Online]. Available: http://queue.acm.org/detail.cfm?id=2898444 . [Accessed: Aug. 22, 2024].

[4] Z. Qian, R. Xia, G. Sun, X. Xing, and K. Xia, "A measurable refinement method of design and verification for micro-kernel operating systems in communication network," Digital Communications and Networks, vol. 9, pp. 1070-1079, 2023.

[5] G. Wang and M. Xu, "Research on tightly coupled multi-robot architecture using microkernel-based, real time, distributed operating system," in Proc. 2008 International Symposium on Information Science and Engineering (ISISE), Shanghai, China, 2008, pp. 978-0-7695-3494-7. DOI: 10.1109/ISISE.2008.80.

[6] V. Benavente, C. Rodriguez, L. Yantas, R. Inquilla, I. Moscol, and Y. Pomachagua, "Comparative analysis of microservices and monolithic architecture," in Proc. 14th IEEE Int. Conf. on Computational Intelligence and Communication Networks (CICN), Lima, Peru, 2022, pp. 177-182. DOI: 10.1109/CICN.2022.30

[7] Mark Richards, "Microkernel Architecture," in Software Architecture Patterns, 2nd ed. Sebastopol, CA: O'Reilly Media, 2022. [Online]. Available: https://learning.oreilly.com/library/view/software-architecture patterns/9781098134280/ch04.html. [Accessed: Aug. 20, 2024].

[8] Raposo, D., Rodrigues, A., Sinche, S., Sá Silva, J., & Boavida, F. (2018). *Industrial IoT Monitoring: Technologies and Architecture Proposal*. **Sensors**, **18**(3568), 1–32. DOI: 10.3390/s18103568

[9] Jayawardena, C., Kuo, I. H., Broadbent, E., & MacDonald, B. A. (2016). *Socially Assistive Robot HealthBot: Design, Implementation, and Field Trials*. **IEEE Systems Journal**, **10**(3), 1056–1067. DOI: 10.1109/JSYST.2014.2337882

[10] Sri Lanka Export Development Board, "Sri Lankan Cocopeat on the Global Stage: A Strategic Exploration of Local Dynamics, Market Entry, and Global Opportunities," Sri Lanka Export Development Board, 2023. [Online]. Available: https://www.srilankabusiness.com/pdf/sri-lankan-cocopeat-on-the-global-stage a-strategic-exploration-of-local-dynamics-market-entry-and-global-opportunities.pdf. [Accessed: Aug. 20, 2024].

[11] T. Vajk, R. Kereskényi, T. Levendovszky and Á. Lédeczi , "Raising the Abstraction of Domain-Specific Model Translator Development," in *2009 16th Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS)*, San Francisco, CA, USA, 2009.

[12] Z. Hou and Z. Yu, "Research of the Workflow Management System Based on Microkernel," *Journal of Theoretical and Applied Information Technology*, vol. 47, no. 1, pp. 266-271, January 2013.

[13] S. Khanzadeh and M. H. Alalfi, "SolOSphere: A Framework for Gas Optimization in Solidity Smart Contracts," 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering - Companion (SANER-C), Rovaniemi, Finland, 2024, pp. 35-45, doi: 10.1109/SANER-C62648.2024.00010.

[14] D. Pritam and J. H. Dewan, "Detection of fire using image processing

techniques with LUV color space," 2017 2nd International Conference for Convergence in Technology (I2CT), Pune, India, 2017, pp. 1158-1162. DOI: 10.1109/I2CT.2017.8226182.