ENHANCING SEAMLESS SUPPLY CHAIN CONNECTIVITY AND ACHIEVING OPERATIONAL EXCELLENCE

by

Dorin Cosmin Iacoban, MBA

DISSERTATION

Presented to the Swiss School of Business and Management Geneva

In Partial Fulfillment

Of the Requirements

For the Degree

DOCTOR OF BUSINESS ADMINISTRATION

SWISS SCHOOL OF BUSINESS AND MANAGEMENT GENEVA

April,2025.

ENHANCING SEAMLESS SUPPLY CHAIN CONNECTIVITY AND ACHIEVING OPERATIONAL EXCELLENCE

by

Dorin Cosmin Iacoban

APPROVED BY

[illegible]

Vasiliki Grougiou

Chairperson

RECEIVED/APPROVED BY:

Admissions Director

Dedication

I dedicate this dissertation to my parents, whose support, encouragement, and sacrifices have been the foundation of my journey. They have instilled in me the values of perseverance and hard work, shaping the person I am today. To my mother, for her boundless love and unwavering faith in my abilities, and to my father, whose wisdom and guidance continue to inspire me every step of the way.

To my mentor, Aleksandar Erceg, Ph.D., whose invaluable insights, patience, and dedication have played a crucial role in shaping this research. Your guidance has been instrumental in helping me navigate this academic journey, and I am deeply grateful for that.

Acknowledgments

I would like to express my deepest gratitude to **Aleksandar Erceg, Ph.D.,** for his invaluable support and guidance throughout my research. His expertise, insightful feedback, and encouragement have been instrumental in shaping this dissertation. He helped me refine my research and guided me in the right direction, ensuring that I stayed focused and made meaningful contributions to the field.

His patience, dedication, and willingness to share his knowledge have been truly inspiring, and I am incredibly grateful for his mentorship. This work would not have been possible without his guidance, and I sincerely appreciate the time and effort he invested in helping me reach this milestone.

ABSTRACT

ENHANCING SEAMLESS SUPPLY CHAIN CONNECTIVITY AND ACHIEVING OPERATIONAL EXCELLENCE

Dorin Cosmin Iacoban
<year>

Dissertation Chair: <Chair’s Name>
Co-Chair: <If applicable. Co-Chair's Name>

In today's increasingly digital and data-intensive environment, supply chain systems are challenged to process vast amounts of real-time data generated by the Internet of Things (IoT) devices. Traditional cloud-based architectures often struggle with high latency and bandwidth constraints, hindering timely decision-making in critical supply chain applications. The research aims to develop and evaluate a dynamic, priority-based task allocation framework that leverages Edge/Fog computing to address these limitations. The core framework is a *Priority Equation* that integrates principles from Queuing Theory (Willig, 1999) and dynamic thresholding, enabling the efficient distribution of tasks processing load across Edge/Fog and Cloud layers.

The research adopts a quantitative approach to analyze the impact of the Priority Equation on latency, resource allocation, and task prioritization in a simulated IoT supply chain environment. Key metrics include task processing time and system scalability under varying data loads by dynamically adjusting task priorities according to real-time demands. The Equation was designed to ensure that critical tasks are processed with minimal delay while optimizing the distribution of non-critical tasks. The results demonstrate that the priority-based framework reduces latency and improves operational efficiency compared to traditional cloud-centric models. Additionally, Edge/Fog computing integration shows improved system scalability, maintaining performance as data volume and task arrival rates increase.

Research contributes to Supply Chain Management by introducing a scalable, adaptive framework that enhances real-time responsiveness and resilience in complex, data-driven environments. Priority Equation offers a practical solution for industries where timely high-frequency data processing is essential, such as logistics, healthcare, and manufacturing. This study lays the groundwork for a more efficient, agile, and customer-centric supply chain model by optimizing task allocation and leveraging decentralized computing, positioning organizations to navigate the digital age's demands better.

TABLE OF CONTENTS

Acknowlegements iv
Abstract v

CHAPTER I: INTRODUCTION 1

1.1 Introduction 1
1.2 Research Problem 3
1.3 Purpose of Research 4
1.4 Significance of the Study 5
1.5 Research Purpose and Questions 5

CHAPTER II: REVIEW OF LITERATURE 6

2.1 Theoretical Framework 30
2.2 Key Benefits 34
2.3 Nomenclature 35
2.4 Priority Equation 36
2.5 Priority_Latency_Optimizer.py (Python) 37
2.6 Summary 40

CHAPTER III: METHODOLOGY 41

3. Overview of the Research Problem 41
3.1 System Arhitecture 41
3.2 Computational Model 42
3.3 Experimental Setup 52
3.3.1 Hardware & Software Environment 56
3.4 Conditions of Applying the Equation 57
3.5 Justification of Parameters 58
3.6 Data Sources & Acquisition Methods 68
3.7 Data Analysis 73
3.8 Research Design and Limitations 84
3.9 Conclusion 89

CHAPTER IV: RESULTS 91

4.1 Research Question One 91
4.2 Research Question Two 93
4.3 Research Question Three 95
4.4 Summary of Findings 97
4.5 Conclusion 97

CHAPTER V: DISCUSSION 99

5.1 Discussion of Results 99
5.2 Discussion of Research Question One 100
5.3 Discussion of Research Question Two 100
5.4 Discussion of Research Question Three 101

CHAPTER VI: SUMMARY, IMPLICATIONS, AND RECOMMENDATIONS 103

6.1 Summary 103
6.2 Implications 104
6.3 Recommendations for Future Research 105
6.4 Conclusion 108

REFERENCES 111

LIST OF TABLES

Figure 1: Three-Tier Architecture for M/G/c Queuing in Real-Time Supply Chain........ 32
Table 2: Calculated E[s^2] vs Approximated Values in Task Allocation Models............. 60
Table 3: Latency Requirements Across Different Industries.. 61
Table 4: Measured Cloud Responses Times Across Major Service Providers................. 63
Table 5: Task Arrival Rates and Batch Size Considerations Across IoT Environments.. 65
Table 6: Scaling Factor and Task Prioritization in IoT Systems 66
Table 7: Utilization Factor (ρ) Interpretation and Applications 67
Figure 8: Task Rates with Standard Deviation Intervals 73

LIST OF FIGURES

Scenario A: Baseline Evaluation of $T_{real\text{-}Time}$ Thresholds and Processor Scaling 76
Scenario B: Alternative Configuration with Similar Scaling 77
Scenario C: Fine-Tuning Local Processing Parameters.. 78
Scenario D: Introducing Cloud Dependence in Varied Workloads.................................. 79
Scenario E/E1: Adjusting Service Rate and Batch Size for Real-Time Processing 80
Scenario F: Bell Curve Simulation and Interpretation... 82

CHAPTER I: INTRODUCTION

1.1 Introduction

In the dynamic landscape of today's digital era, Cloud computing has evolved into a fundamental asset for businesses, which can spur numerous benefits for organizations, such as capital investment savings, simplified operations, scalability, improved information visibility, sustainability, and faster deployment (Attaran and Woods, 2018). Renowned for its scalability, flexibility, and cost-effectiveness, the Cloud has become the preferred choice of most enterprises (Nai-meng et al., 2019). However, with the escalating adoption of Cloud technology, the issue of *network latency* has surfaced as a crucial challenge requiring proactive attention for optimal Cloud performance. *Edge computing* is considered, in most cases, the option for fast processing of vast amounts of data based on proximity or in-house capabilities. Edge computing is the practice of processing (Ferreira, 2021) and computing client data closer to the data source rather than on a centralized server or a cloud-based location (Mohanan, 2022). In supply chain operations, real-time decision-making is crucial, and the latency associated with cloud-centric models poses significant constraints on operational efficiency.

Past research (Yousefpour et al., 2019) indicates that Edge computing effectively addresses privacy, latency, and connectivity issues, primarily attributable to its proximity to users. The meticulous design and implementation of a supply chain infrastructure necessitate careful consideration of critical factors, including processing speed and bandwidth.

However, there is still a need for additional investigation into bandwidth savings through the utilization of Edge computing (Yousefpour et al., 2019). The study by (Yousefpour et al., 2019) highlights that as data velocity and volume increase, moving the big data from the IoT devices to the Cloud might not be efficient or might even be infeasible in some cases due to bandwidth constraints. Based on the study observation, bringing Edge and Cloud computing closer to the user (Ahmed et al., 2017) will foster an environment of enhanced operational efficiency throughout the entire business process.

Traditionally, Cloud computing has been the cornerstone for managing the vast array of geographically dispersed IoT devices and their associated applications (Tuli et al., 2019). The physical separation between Cloud data centers and IoT devices often results in increased communication delays, affecting both data transmission and the delivery of services. Such latency is particularly problematic for critical healthcare and smart city infrastructure applications, which can significantly diminish Quality of Service (QoS) (Tuli et al., 2019). Moreover, the sheer volume of data generated by IoT devices in a short span can lead to substantial network congestion, primarily when numerous devices concurrently transmit data to Cloud data centers via the global internet (Tuli et al., 2019). Edge computing paradigms discussed by Bonomi *et al.* (2012) have been developed to address challenges inherent in a cloud-centric IoT model. These paradigms prioritize the use of local computing resources at the Edge of the network to execute real-time IoT applications, effectively reducing latency and network congestion; still, network planning and design of Edge/Fog networks is an important research topic, and yet not many studies have been performed in this area (Berenberg and Calder, 2022).

Edge computing brings computation and data storage closer to the location where it is needed, minimizing the latency traditionally associated with Cloud computing. This enhancement in computational efficiency and the reduction in request-response times significantly improve the processing overhead, thereby streamlining supply chain operations (Perumal et al., 2022).

Integrating advanced technologies into Supply Chain Management practices addresses current limitations and sets the stage for a more resilient, agile, and customer-centric supply chain ecosystem (Kaur et al., 2024). The evolution is decisive for industries to remain competitive in a rapidly changing economic landscape, where the efficiency of logistics and supply chain operations directly impacts business success and customer satisfaction.

1.2 Research Problem

In the contemporary, highly interconnected Supply Chain landscape, latency, task allocation, and resource efficiency management have become increasingly complex, particularly with the widespread integration of the Internet of Things (IoT), Edge/Fog, and Cloud computing (Nguyen et al., 2024). The continuous influx of data from sensors, Radio Frequency Identification **(**RFID) tags, barcodes, robotics, and other smart devices requires real-time processing to maintain operational efficiency (Nguyen et al., 2024). Developing a dynamic, priority-based task allocation equation is key to addressing these challenges. Such a model would facilitate low-latency processing for time-sensitive tasks, optimize computational resource distribution across Edge/Fog and Cloud layers, and enhance overall cost efficiency within Supply Chain operations. Based on these observations, the following question arises:

"How can a dynamic, priority-based task allocation equation be developed to improve latency, manage task load, and enhance resource efficiency across Edge/Fog and Cloud computing layers in Internet of Things-driven Supply Chain Environments?"

The increased need for low-latency processing reveals that traditional cloud-centric supply chains struggle with latency, especially as IoT data sources multiply and demand near-instantaneous processing (Oliveira and Handfield, 2019) and that Real-Time Supply Chain applications, latency can significantly impact decision-making and, most notably the operational efficiency.

Challenges of task allocation in multi-layered systems create a complex data flow across Edge and Cloud layers, and the question of *how to prioritize tasks* to ensure critical, time-sensitive data is processed immediately while less urgent data is handled in a way that does not overburden the system. The question led to exploring models beyond simple cloud-based architectures, where tasks could be allocated dynamically based on urgency.

At this point, the role of Queuing Theory (Willig, 1999) directed attention to formalize the task prioritization process. Traditional queuing models provide insight into managing tasks in a multi-processor environment. However, they are limited in handling real-time prioritization needs, which led to integrating *Queuing Theory* with *task thresholding* to manage high-velocity data and demand peaks in supply chains and a *dynamic prioritization* that will adjust based on real-time load, prioritizing tasks by urgency thresholds and managing load across processors. The Equation aims to optimize task allocation across Edge layers, allowing the supply chain to function efficiently and highly responsive.

1.3 Purpose of Research

The research seeks to tie the gap between traditional cloud-based supply chain models and the emerging need for low-latency, proximity-based processing by developing a Priority Equation that enables responsive, prioritized data handling across Edge/Fog and Cloud layers. Through this framework, the study aims to contribute to upcoming supply chains operating with enhanced agility, efficiency, and responsiveness, benefiting industries that depend on timely data processing and decision-making.

1.4 Significance of the Study

The study is significant since it addresses critical challenges in modern supply chain management, particularly latency and real-time decision-making in data-heavy IoT-driven environments. By introducing and validating a *Priority Equation* that integrates Edge/Fog computing with Queuing Theory and task thresholding, the research contributes to a scalable and adaptive framework for task prioritization that meets the evolving needs of global supply chains and also offers practical solutions for industries where responsiveness and operational efficiency are paramount.

1.5 Research Purpose and Questions

The research aims to develop and evaluate a priority-based task allocation framework using Edge computing guided by a dynamic Priority Equation. The framework aims to reduce latency, optimize resource allocation, and enhance real-time decision-making within IoT-enabled supply chain systems by addressing key questions:

- Research Question One: *How does a priority-based task allocation equation affect latency and resource utilization in a multi-layered supply chain system?*
- Research Question Two: *What role do dynamic thresholding and Queuing Theory play in optimizing task prioritization within Edge/Fog computing environments?*
- Research Question Three: *How does integrating Edge/Fog and Cloud computing improve the scalability and responsiveness of supply chains in real-time, high-data environments?*

Chapter II:
REVIEW OF LITERATURE

The increasing complexity of computational workloads in Edge, Fog, and Cloud environments has led to the development of various task scheduling and resource allocation methodologies. The literature reviewed in this section encompasses heuristic and mathematical evaluating contributions and limitations in comparison to the Priority Equation, which leverages Queuing Theory and real-time prioritization techniques.

The study "*A Method Based on the Combination of Laxity and Ant Colony System for Cloud-Fog Task Scheduling"* by (Xu et al., 2019) introduces a hybrid algorithm, combining laxity-based prioritization with an ant colony optimization system (LBP-ACS) to address these challenges. This section examines the contributions of this approach and evaluates its strengths and limitations. The LBP-ACS algorithm focuses on two primary objectives: enhancing the sensitivity of task prioritization to delay requirements and minimizing energy consumption during task scheduling. The laxity-based priority algorithm computes task urgency by calculating the allowable delay before the task's deadline is breached. Tasks with shorter laxity are assigned a higher priority, prioritizing delay-sensitive operations. While this approach effectively handles interdependent tasks modeled as directed acyclic graphs (DAGs), its reliance on static laxity metrics limits its adaptability to dynamic workloads. In contrast, the Priority Equation incorporates real-time system metrics, enabling dynamic task prioritization and offloading decisions that adapt to fluctuating system conditions.

The LBP-ACS algorithm's ant colony optimization component ensures global resource allocation optimization by iteratively improving task scheduling decisions. By incorporating energy consumption and task deadlines as key parameters, this approach achieves a balance between energy efficiency and task completion time. However, the iterative nature of the ant colony optimization introduces computational overhead, which may limit its scalability in large-scale IIoT deployments. The Priority Equation, designed for Edge-Fog-Cloud architectures, provides a lightweight alternative by minimizing per-task latency and dynamically distributing tasks between local and remote resources. It achieves this without requiring extensive heuristic optimization, leveraging real-time task prioritization to optimize system efficiency.

The researchers (Xu et al., 2019) highlight the benefits of their approach in reducing task scheduling failure rates and optimizing energy consumption. Experimental results demonstrate that LBP-ACS outperforms traditional scheduling algorithms such as HEFT and Greedy for Energy (GfE) in scenarios with mixed task deadlines. However, the study primarily addresses energy consumption and deadline constraints without explicitly accounting for the impact of network latency in Edge-Fog-Cloud architectures. In contrast, the Priority Equation addresses this gap by integrating latency as a core parameter in task prioritization, ensuring that real-time tasks are processed locally whenever possible to minimize transmission delays.

The study "*Composition-Driven IoT Service Provisioning in Distributed Edges"* by (Deng et al., 2018) presents an optimized service cache policy designed to enhance the performance of service provisioning in mobile edge computing (MEC) systems. This study highlights the importance of caching strategies that leverage the composability of services to reduce latency and improve resource utilization. By introducing a heuristic algorithm for average service response time (ASRT) minimization, the authors aim to improve the efficiency of MEC architectures. Deng et al. (2018) address the challenges of latency and resource constraints in IoT systems by proposing a cache optimization framework that classifies services as composite and atomic. They utilize service composition graphs

(SCGs) to model the hierarchical relationships between services and optimize caching decisions based on service popularity, resource consumption, and composition structures. The proposed heuristic algorithm effectively reduces the ASRT by prioritizing frequently invoked services while accounting for the dependencies within SCGs. However, the reliance on service popularity and static composition structures limits the adaptability of this approach in dynamic IIoT environments where task priorities and workloads fluctuate.

A significant contribution of the "*Composition-Driven"* study by Deng et al. (2018) is its incorporation of composite services, which are constructed from atomic services, into the caching framework. This approach enhances resource reuse and minimizes redundant computations by prioritizing caching decisions that maximize service reusability. While effective in improving resource utilization, the method assumes that task execution times on Edge and Cloud servers are equivalent, disregarding the impact of varying resource capacities and network conditions on latency. The Priority Equation explicitly accounts for latency differences by prioritizing tasks with stringent timing requirements for local processing, ensuring compliance with IIoT latency constraints.

The experimental results presented by Deng et al. (2018) demonstrate that their heuristic algorithm significantly reduces ASRT compared to enumeration and other evolutionary algorithms, such as genetic algorithms and particle swarm optimization. Despite these advancements, the heuristic approach does not incorporate dynamic offloading thresholds or account for real-time edge-cloud collaboration. The Priority Equation addresses these gaps by dynamically balancing local and Cloud processing based on system states and task priorities, achieving latency performance in real-time IIoT applications.

The subsequent literature review is "*Optimization of Task Scheduling in Fog-Based Regions and Cloud (FBRC)"* by Hoang and Dang (2017), which introduces a novel concept of fog-based regions combined with cloud computing to address task scheduling in latency-sensitive environments. This framework, termed FBRC, focuses on reducing task completion times and optimizing resource utilization by efficiently distributing computational tasks between local Fog regions and Cloud servers. The contributions of this work are significant in their development of an area-specific architecture and a heuristic algorithm for task scheduling.

In their work, Hoang and Dang (2017) conceptualize regions as dynamic clusters of Fog nodes that are geographically distributed and can exchange resources to handle computational tasks. The authors highlight the inherent trade-offs between using local resources in regions, which minimize data transmission latency, and relying on Cloud servers, which offer higher computational power at the cost of increased transmission delays. This duality forms the foundation of their FBRC framework, where tasks are dynamically allocated based on resource availability and location proximity. While the FBRC framework effectively leverages regional resource sharing to reduce task completion times, its reliance on heuristic algorithms introduces challenges in adapting to real-time workload fluctuations. The heuristic algorithm developed for FBRC employs a multi-step process to allocate tasks across regions and Cloud servers. This algorithm minimizes task completion time by iteratively assigning resources to tasks based on latency constraints and resource availability. Still, its computational complexity increases significantly with the number of regions and tasks, potentially limiting scalability in large-scale IIoT deployments. The Priority Equation addresses this limitation by using lightweight, real-time computations that prioritize latency-sensitive tasks for local processing while offloading non-critical tasks to the Cloud, which ensures that the system remains efficient even under high task loads. At the same time, the researchers Hoang and Dang (2017) validate the FBRC framework through simulations that compare its performance with cloud-only and region-only scheduling schemes. The results demonstrate that FBRC consistently outperforms these alternatives in minimizing task completion times, especially at higher task arrival rates. The study does not explicitly account for the dynamic nature of task priorities, which can vary significantly in IIoT environments. The Priority Equation builds on this gap by directly integrating task prioritization into its framework, ensuring critical tasks are processed on time while maintaining overall system efficiency.

Another strength of the FBRC framework lies in its emphasis on optimizing resource allocation across heterogeneous Fog and Cloud environments by considering varying service rates and transmission latencies, and the framework provides a balanced approach to resource utilization. Nevertheless, its static configuration of task scheduling parameters may hinder its ability to adapt to sudden workload or resource availability changes. In this illustration, the Priority Equation enhances this adaptability by dynamically adjusting offloading thresholds (s) based on real-time system states, enabling more responsive and efficient resource allocation.

For the following study done by Sohani and Jain (2021), "*A Predictive Priority-Based Dynamic Resource Provisioning Scheme With Load Balancing in Heterogeneous Cloud Computing,"* presents a novel approach to improving cloud-based task scheduling and resource provisioning. The research introduces the Predictive Priority-based Modified Heterogeneous Earliest Finish Time (PMHEFT) algorithm, which enhances load balancing and scheduling efficiency in heterogeneous cloud environments. The core idea is to dynamically allocate computing resources based on predictive modeling to minimize makespan, reduce energy consumption, and optimize Quality of Service (QoS).

Researchers Sohani and Jain focus on optimizing task scheduling by leveraging a priority queue that predicts resource demand and workload distribution. The PMHEFT algorithm builds on existing task scheduling strategies such as HEFT (Heterogeneous Earliest Finish Time) and CPOP (Critical Path on a Processor), modifying them to account for dynamic workload fluctuations. Their approach significantly improves makespan minimization and enhances load balancing across virtual machines (VMs). However, while PMHEFT optimizes scheduling efficiency in Cloud environments, it does not explicitly address task offloading in multi-cloud and Edge computing. Still, their study introduces a predictive model for estimating future workloads and adjusting resource allocation accordingly, which ensures that Cloud resources are optimally provisioned to prevent Service Level Agreement Violation (SLA) violations and minimize resource wastage. However, the predictive approach relies on historical workload patterns, which may not always reflect real-time conditions.

Another key contribution of the PMHEFT algorithm is its priority-based queuing mechanism, which ensures that high-priority tasks are executed first while balancing the load across multiple Cloud nodes. The approach enhances load balancing and reduces the probability of resource bottlenecks. However, PMHEFT does not explicitly consider geographical distance-based latency variations in multi-cloud environments. The Priority Equation does not currently integrate Critical Path Analysis. However, this could be a potential enhancement for dynamically assigning tasks to the fastest available Cloud provider, ensuring minimal transmission delay based on processing needs and latency requirements. The concept might present an optimal task execution path across multiple cloud regions, ensuring critical computations are processed with minimal transmission latency.

The experimental evaluation of the PMHEFT algorithm demonstrates superior performance compared to traditional HEFT, MHEFT, and Dynamic HEFT algorithms, particularly in terms of makespan reduction and resource utilization. However, the study primarily focuses on cloud-only architectures, whereas the Priority Equation is designed for Edge-Fog-Cloud hierarchies, making it more suitable for latency-sensitive IIoT/IoT applications.

Another reference used is the study done by Chakraborty et al. (2022), "*Intelligent Latency-Aware Tasks Prioritization and Offloading Strategy in Distributed Fog-Cloud of Things,"* which presents a multi-layered latency-aware offloading model for Fog-Cloud computing, incorporating fuzzy logic-based task prioritization and an Elitism-based Multipopulation Jaya (EMPJ) algorithm for task scheduling. The research highlights the importance of deadline constraints and priority-aware scheduling in distributed environments, focusing on minimizing offloading time and computational latency while ensuring efficient resource utilization.

One of the primary contributions of this work is the classification of tasks based on priority levels and their deadline sensitivity. The proposed fuzzy logic-based task classifier categorizes tasks into high-priority, medium-priority, and low-priority groups, assigning them to different computational layers accordingly. This classification ensures that high-priority and deadline-sensitive tasks are processed in the Cloud, while moderate-priority tasks are assigned to Fog nodes with mixed computational resources, and low-priority tasks are executed at local Fog nodes to prevent unnecessary transmission delays.

The EMPJ algorithm proposed in their study extends traditional metaheuristic approaches by incorporating multi-population scheduling, which enhances exploration-exploitation balance in computing resource allocation. The strategy effectively reduces waiting time and service latency by dynamically mapping tasks to heterogeneous computing nodes. However, the Priority Equation ensures that critical tasks are always prioritized without requiring computationally intensive metaheuristic optimization algorithms.

A key area of comparison lies in latency management and scheduling efficiency. The fuzzy-based offloading strategy in the Chakraborty et al. (2022) study ensures that tasks with hard deadlines are executed in the most resource-efficient nodes, whereas the Priority Equation ensures that local processing capacity is fully utilized before offloading occurs. Moreover, the Priority Equation is inherently adaptable, making it suitable for real-time IIoT environments, whereas the EMPJ-based approach may introduce computational overhead due to its reliance on iterative metaheuristic optimization.

The analysis performed by Bali et al. (2023), "*An Effective Technique to Schedule Priority-Aware Tasks to Offload Data on Edge and Cloud Servers,"* proposes the Priority-Aware Task Scheduling (PaTS) algorithm, which prioritizes tasks based on urgency and assigns them to Edge or Cloud servers accordingly. The PaTS approach formulates a multi-objective optimization problem, integrating a four-queue model where very-urgent and urgent tasks are processed at the Edge to minimize latency, whereas moderate and non-urgent tasks are offloaded to the Cloud, where computational resources are plentiful. The scheduling is further optimized using the NSGA-II genetic algorithm, improving average queue delay, computation time, and energy efficiency compared to benchmark models.

Their approach shares similarities with the Priority Equation, particularly in task prioritization and offloading strategies. Both methods aim to enhance latency-sensitive task processing by ensuring critical tasks receive priority execution at the most suitable computational layer. The Priority Equation is distinct in its reliance on Queuing Theory and mathematical modeling, dynamically adjusting offloading thresholds based on real-time system parameters, unlike PaTS, which classifies tasks into discrete priority groups and assigns them accordingly.

Another key difference lies in the optimization methodology. The PaTS algorithm leverages bio-inspired genetic optimization (NSGA-II) to refine scheduling, which introduces computational overhead due to iterative search and selection processes. On the other hand, the Priority Equation is lighter in computation, as it uses a closed-form mathematical expression to dynamically determine task priority, making it more suitable for real-time IIoT applications with high task arrival rates. Furthermore, while PaTS effectively categorizes tasks based on urgency, it does not explicitly model the queuing dynamics and processing delays in the same structured manner as the Priority Equation, which integrates latency estimations directly into task prioritization calculations. This difference gives the Priority Equation an advantage in real-time adaptability, as it accounts for dynamic fluctuations in workload and processing speed without requiring constant re-evaluation through genetic algorithms.

The examination conducted by You and Tang (2021), "*Efficient Task Offloading Using Particle Swarm Optimization Algorithm in Edge Computing for Industrial Internet of Things,"* presents a Particle Swarm Optimization (PSO)-based task offloading strategy aimed at optimizing latency and energy consumption in Industrial IoT (IIoT) environments. The authors emphasize the need for an efficient task allocation strategy in mobile edge computing (MEC) systems, where limited processing capabilities and network constraints demand intelligent decision-making. The proposed multi-objective optimization problem considers time delay, energy consumption, and execution cost to achieve optimal task placement across heterogeneous Edge servers.

The PSO-based offloading approach introduced in their research enables dynamic and adaptive task scheduling, improving system efficiency by balancing computational loads across multiple MEC servers. The fitness function of PSO determines the most cost-effective task placement by evaluating total offloading costs under energy constraints. Through extensive simulations, the study demonstrates that PSO-based offloading outperforms traditional heuristic approaches, such as the Genetic Algorithm (GA) and

Simulated Annealing (SA), by reducing execution delay, optimizing energy consumption, and improving overall system throughput. However, the study acknowledges challenges in tuning PSO parameters, as optimal performance relies on selecting appropriate acceleration coefficients and penalty functions.

Comparing You and Tang (2021) approach to the Priority Equation, both models aim to enhance task allocation efficiency and latency reduction in distributed computing architectures. The Priority Equation, however, relies on Queuing Theory and real-time prioritization metrics rather than heuristic optimization techniques, unlike the PSO-based approach, which relies on global search heuristics. While PSO excels at finding near-optimal task placement solutions by exploring multiple execution paths, it introduces computational overhead due to its iterative nature. The Priority Equation offers a more lightweight and deterministic approach, ensuring that high-priority tasks are processed first based on system state conditions rather than evolutionary search techniques. Additionally, the PSO model does not explicitly incorporate priority-based queuing mechanisms, whereas the Priority Equation ensures task prioritization by dynamically allocating computing resources according to service demands.

The research conducted by Bali et al. (2021), "*Smart Architectural Framework for Symmetrical Data Offloading in IoT,"* introduces a structured approach to optimizing data offloading strategies in IoT networks. The work emphasizes the increasing burden of data traffic on cloud-based architectures and explores the integration of Edge and Fog computing as alternatives to mitigate network congestion and latency. The authors categorize offloading methodologies into data offloading, computation offloading, and task offloading, aiming to reduce system overload and improve real-time response efficiency. The proposed smart architectural framework ensures symmetrical data offloading, evenly distributing computational and storage resources across Edge and Fog nodes to maintain balance and efficiency.

Their approach aligns with the goals of the Priority Equation, which also seeks to reduce latency and optimize resource allocation in real-time environments. However, the two methodologies differ significantly in their implementation. The Priority Equation operates based on Queuing Theory and dynamic prioritization, where task arrival rates, service rates, and utilization factors govern the offloading decisions. In contrast, the study *"Smart Architectural Framework"* focuses on symmetry in data distribution rather than adaptive prioritization. The Priority Equation prioritizes tasks dynamically, ensuring low-latency processing of critical tasks, while the *"Smart Framework"* seeks to balance computational loads across all available nodes, which may not always prioritize the most time-sensitive operations. Another key difference is that the *"Smart Architectural Framework"* relies on heuristic-based decision-making for offloading, reducing bandwidth consumption, and maintaining even distribution across resources. While this can be effective for long-term stability, it may not provide the real-time adaptability that latency-sensitive applications require.

Additionally, Bali et al. (2021) propose a workflow in which data filtering occurs at the Edge before offloading to the Cloud, preventing unnecessary transmissions and optimizing bandwidth utilization. This structured filtering process is helpful in managing congestion but lacks the dynamic prioritization inherent in the Priority Equation, which adjusts task allocation in real-time based on system conditions. The Priority Equation ensures that processing capacity is always utilized efficiently, offloading only when necessary to prevent saturation.

The study by Bali et al. (2021) presents a well-structured offloading model that effectively balances data distribution and resource allocation; however, its reliance on static heuristic-based methodologies may limit its adaptability to real-time fluctuations in network traffic. The Priority Equation, in contrast, is more adaptive and latency-focused, making it more suitable for high-priority industrial IoT applications where response times are critical. Future research could explore hybrid approaches that integrate dynamic prioritization with structured resource balancing, leveraging the strengths of both methodologies for optimal task scheduling in IoT networks.

The work "*An Efficient Algorithm for Data Transmission Certainty in IIoT Sensing Networks: A Priority-Based Approach"* by Nalbant et al. (2024) presents a novel caching algorithm to optimize data transmission certainty in industrial IoT (IIoT) environments. The study introduces a periodic popularity prediction and size-based caching (PPPS) algorithm to enhance cache hit rates and minimize latency, particularly in industrial scenarios with strict timeliness requirements.

The PPPS algorithm centers on caching strategies tailored for IIoT applications, emphasizing the prediction of content popularity based on recent request sequences. By integrating metrics such as content size, timeliness, and historical popularity, the algorithm dynamically evaluates the caching value of each item and determines optimal content replacement strategies. Unlike conventional caching methods like LRU and LFU, which often fail in dynamic industrial scenarios, the proposed algorithm achieves better cache hit rates and lower transmission delays. However, while effective for improving caching efficiency, the static prediction model employed by PPPS may struggle to adapt to real-time workload fluctuations.

One of the significant contributions of the study by Nalbant et al. (2024) is the introduction of a shot noise model (SNM) for user request prediction. This model captures temporal variations in content popularity, allowing the caching algorithm to anticipate future requests based on historical trends. However, the SNM model does not account for the variability in real-time task priorities or the dynamic nature of IIoT workloads. The Priority Equation addresses this limitation by dynamically adjusting offloading thresholds (s) to optimize task distribution between local and remote resources, ensuring compliance with latency-sensitive IIoT requirements.

The experimental results of the PPPS algorithm demonstrate substantial improvements over traditional caching strategies in terms of cache hit rates and latency reductions. For instance, under dynamic content scenarios, the proposed algorithm achieved a 15.3% higher hit rate compared to Greedy Dual-Size (GDS) and a 24.8% higher rate than Least Frequently Used (LFU). However, these improvements are contingent upon pre-defined popularity patterns of user patterns and content size distributions, which may not reflect the diverse and unpredictable workloads encountered in IIoT systems. The Priority Equation complements these advancements by incorporating real-time adaptability, making it more suitable for environments where workload characteristics change frequently. The PPPS algorithm's reliance on static thresholds for determining content popularity and caching decisions limits its scalability in large-scale IIoT deployments. Furthermore, the authors note the limitations of the algorithm's high data replacement rates and lack of synchronization for multi-dimensional feature analysis. In contrast, the Priority Equation employs lightweight computations and scalable dynamic metrics, ensuring efficient resource utilization and scalability across diverse IIoT applications.

The research conducted by Tao et al. (2021), "*Content Popularity Prediction in Fog-RANs: A Bayesian Learning Approach,"* suggests a novel strategy for predicting content popularity in Fog Radio Access Networks (F-RANs) using a Bayesian learning framework. This method addresses the challenges of limited caching capacity in Fog Access Points (F-APs) by developing a Gaussian process-based Poisson regression model that leverages content features and request probabilities. The research introduces a hierarchical probabilistic model designed to predict content popularity while accounting for the nonlinear relationships between content features and request probabilities. The research model enables F-APs to optimize caching strategies by predicting the popularity of existing and newly added content using Bayesian learning. The approach is robust to overfitting, especially in scenarios with sparse data. However, its focus on caching efficiency is limited to static and semi-dynamic environments, where the request probabilities and features are assumed to follow relatively stable patterns. The Priority Equation, in contrast, offers a dynamic framework that adjusts task prioritization and offloading thresholds in real-time, ensuring adaptability to highly dynamic IIoT workloads.

A key contribution of Tao et al. (2021) study is the integration of stochastic variance reduced gradient Hamiltonian Monte Carlo (SVRG-HMC) into the Bayesian learning model. This innovation improves the convergence rate of the model, allowing faster and more accurate predictions of content popularity. While this method demonstrates computational efficiency, it does not directly address latency-sensitive tasks or their prioritization in IIoT systems. The evaluation of the Bayesian model using the MovieLens dataset highlights its effectiveness in reducing root mean square error (RMSE) and improving cache hit rates compared to other methods. However, the experimental framework relies on historical data and pre-defined feature sets, which may not adequately reflect the dynamic requirements of real-time IIoT systems. In contrast, the Priority Equation prioritizes tasks based on current system states and dynamically allocates resources, ensuring optimal performance in latency-sensitive applications without dependence on historical trends.

Another limitation of the study by Tao et al. (2021) is its focus on content popularity prediction rather than task-level prioritization or scheduling. The proposed model optimizes caching decisions based on popularity but does not address the distribution of computational tasks across Edge and Cloud resources. The Priority Equation fills this gap by integrating dynamic offloading thresholds (s) to balance local and Cloud processing, ensuring that critical tasks are handled with minimal latency while optimizing resource use across the system.

The research presented with the title "*An MCDM Optimization-Based Dynamic Workflow Scheduling Used to Handle Priority Tasks for Fault Tolerance in IIoT"* by Jamal and Muqeem (2023) introduces a multi-criteria decision-making (MCDM) approach to address these challenges. The MCDM approach proposed by Jamal and Muqeem integrates the Best-Worst Method (BWM) and the VIKOR technique for dynamic task prioritization and resource allocation in IIoT systems. This method emphasizes reducing the makespan, improving throughput, and ensuring fault tolerance by dynamically allocating resources based on task importance, deadlines, and system capacity. The authors demonstrate that their approach outperforms traditional algorithms, such as Min Connection and Latent Regression Topic Model (LRTM), in terms of resource utilization and fault tolerance. However, the reliance on centralized decision-making and increased computational complexity in the MCDM model poses limitations in highly dynamic environments where real-time adaptability is critical. The Priority Equation addresses these limitations by leveraging lightweight real-time parameters to ensure dynamic adaptability and efficient task prioritization without significant computational overhead.

One of the key contributions of Jamal and Muqeem's (2023) study is its focus on fault tolerance for priority tasks in IIoT systems. Their methodology incorporates redundant systems and robust fault-detection mechanisms to enhance system reliability. While this aligns with the goals of the Priority Equation, their approach does not explicitly address latency-sensitive task prioritization in Edge computing environments. The Priority

Equation distinguishes itself by integrating dynamic offloading thresholds (s) and real-time metrics to optimize task distribution between Edge and Cloud resources, ensuring lower latency for critical tasks and better resource scalability.

The use of the VIKOR technique in the MCDM model highlights the importance of multi-criteria optimization in IIoT task scheduling. By balancing trade-offs among multiple objectives, such as cost, time, and resource utilization, the proposed method offers a comprehensive framework for decision-making; however, the VIKOR model's dependency on pre-defined criteria, which limits adaptability to sudden workload changes. In contrast, the Priority Equation dynamically adjusts task prioritization and offloading decisions based on current system states, providing a more flexible and responsive solution for IIoT environments.

Another significant aspect of the MCDM approach is its emphasis on reducing makespan and improving throughput by utilizing optimization techniques such as parallel computing. While these improvements are notable, the approach does not fully explore the potential of decentralized task scheduling in Edge computing architectures. The Priority Equation, explicitly designed for Edge-Fog-Cloud architectures, ensures efficient task allocation by prioritizing latency-critical tasks locally while offloading less critical tasks to the Cloud, thereby achieving a balance between system performance and resource utilization.

The research "*Dynamic Multi-Level Auto-Scaling Rules for Containerized Applications"* by Taherizadeh and Stankovski (2019) presents a novel approach to managing resource allocation in cloud environments using dynamic multi-level (DM) auto-scaling. The method introduces dynamic thresholds for container-based applications, integrating both infrastructure-level and application-level metrics to optimize resource utilization and performance. Researchers Taherizadeh and Stankovski focus on finegrained resource scaling to adapt to dynamic workload fluctuations. Their method combines CPU, memory, and bandwidth monitoring with application-specific metrics such as response time and throughput. By introducing dynamic thresholds instead of static ones, the DM method ensures a more adaptive response to changing conditions, preventing both over-provisioning and under-provisioning of resources. However, while this method is effective for containerized applications, it does not explicitly address task prioritization, particularly for latency-sensitive tasks. The DM auto-scaling approach leverages a rule-based algorithm that launches or terminates container instances based on real-time monitoring data. This algorithm balances response time requirements with resource availability, achieving significant improvements in resource utilization and system performance compared to static scaling methods. However, the rule-based nature of the DM method introduces limitations in highly dynamic environments where task arrival patterns and system loads change unpredictably.

Experimental results from the DM method show superior performance compared to traditional scaling approaches. The authors report a reduction in response time violations and improved resource efficiency across diverse workload scenarios, including real-world use cases. However, these experiments focus primarily on container-level resource adjustments and do not explore the implications of task-level scheduling. Another strength of the DM method is its ability to integrate multi-level monitoring systems, such as the SWITCH monitoring system, which combines container-level and application-level metrics. This integration provides a holistic view of system performance, enabling precise scaling decisions. However, the approach relies heavily on pre-defined thresholds and static adaptation intervals, which may hinder its responsiveness in scenarios with rapid workload fluctuations.

The research conducted by Kafle and Al Muktadir (2020), "*Intelligent and Agile Control of Edge Resources for Latency-Sensitive IoT Services,"* presents a resource adjustment scheme for virtualized Edge environments using machine learning (ML). This work addresses the dynamic allocation of computational resources for latency-sensitive virtual network functions (VNFs), employing multiple regression models for resource demand prediction and agile adjustments. Kafle and Muktadir focus on reducing resource utilization while maintaining service latency below specified thresholds. Their proposed system incorporates both offline training and online retraining of regression models, enabling continuous resource allocation optimization based on workload variations. The models dynamically predict resource demands by analyzing input metrics such as workload intensity, resource utilization, and system latency. While this method ensures efficient resource use and scalability, its reliance on periodic retraining introduces potential delays in highly dynamic environments. The Priority Equation addresses this limitation by utilizing lightweight computations that dynamically adjust task priorities and offload thresholds (s) in real time, achieving immediate adaptability without relying on retraining cycles.

A key innovation of Kafle and Al Muktadir's (2020) work is the use of regression-based models to predict CPU demand, allowing resource adjustments to occur at one-second intervals. This scheme significantly improves resource utilization and reduces latency violations compared to conventional threshold-based algorithms. However, the approach prioritizes CPU-intensive tasks and does not extend to holistic resource management, such as bandwidth or memory allocation. The Priority Equation offers a broader perspective by optimizing task distribution across Edge, Fog, and Cloud layers, ensuring that latency-sensitive tasks are processed locally while non-critical tasks are offloaded to remote servers. The authors validate their approach through extensive simulations using real-world workload patterns, such as proportional, step, and Poisson distributions. They report a 58.2% reduction in latency violations and a 21.9% decrease in CPU resource demand compared to baseline algorithms. These results highlight the efficacy of their regression models in resource optimization. However, the experimental framework is limited to pre-defined workloads and static latency thresholds, which may not fully capture the dynamic nature of IIoT environments. Another strength of their research lies in its agile resource control mechanism, which adapts resource allocation every second, and this aspect is particularly beneficial for latency-sensitive applications such as automated driving and telemedicine. However, relying on supervised learning for offline training requires extensive dataset preparation, which may limit scalability in environments with rapidly changing conditions. The Priority Equation mitigates this challenge by employing a simplified, heuristic approach to task prioritization and offloading, reducing dependency on historical data and enabling immediate deployment in dynamic IIoT scenarios.

The research paper titled "*Performance Interference-Aware Vertical Elasticity for Cloud-Hosted Latency-Sensitive Applications*" by Shekhar et al. (2018) introduces a proactive vertical scaling framework designed to address the challenges of performance interference and workload variability in cloud-hosted latency-sensitive applications. This study optimizes resource allocation by leveraging Gaussian Processes (GP)-based machine learning to predict real-time workload and performance interference levels.

In their research, Shekhar et al. (2018) highlight the limitations of traditional horizontal elasticity methods, such as initialization delays and suboptimal resource utilization, and argue for the advantages of vertical elasticity in reducing latency and improving resource efficiency. By dynamically resizing application containers or virtual machines, their approach ensures better adherence to service-level objectives (SLOs) while minimizing interference from co-located batch applications. However, while the framework effectively models performance interference and predicts resource requirements, it introduces computational overhead associated with online training and prediction using GP models. The GP-based predictive model developed by Shekhar et al. (2018) accounts for both workload variability and performance interference by clustering system-level metrics using K-Means and building separate GP models for each cluster. This aspect enables precise predictions of latency and resource needs for latency-sensitive applications. While this approach significantly reduces tail latency and resource contention, it assumes a static set of interference profiles and clustering thresholds, limiting its adaptability to dynamic IIoT environments. Experimental results (Shekhar et al., 2018) demonstrate that their proactive vertical scaling framework achieves a 39.46% reduction in tail latency compared to reactive approaches, even at the cost of higher resource utilization. By focusing on task-level optimization and dynamic offloading, the Priority Equation ensures lower latency without significant increases in resource consumption, achieving a more balanced trade-off between performance and efficiency. Furthermore, while the vertical elasticity approach excels in managing latency-sensitive applications in Cloud environments, its reliance on centralized resource control limits its scalability for distributed architectures such as Edge-Fog-Cloud systems, which are central to IIoT applications. Another key contribution of the Shekhar et al. (2018) framework is its use of predictive modeling to enable proactive scaling decisions, mitigating the limitations of reactive threshold-based approaches. However, this framework primarily targets CPU-intensive applications and does not address the challenges associated with other resources, such as memory, bandwidth, or network latency. The Priority Equation extends these contributions by providing a comprehensive framework that dynamically distributes tasks across local and Cloud resources, optimizing CPU, network, and storage resources.

Last but not least, the study "*Priority-Based Task Scheduling and Resource Allocation in Edge Computing for Health Monitoring Systems"* by Sharif et al. (2023) presents a Priority-Based Task Scheduling and Resource Allocation (PTS-RA) system to optimize computational efficiency in mobile edge computing (MEC) for healthcare applications. The work focuses on task prioritization based on emergency levels derived from real-time patient monitoring data. The authors emphasize the importance of reducing task execution latency and minimizing bandwidth consumption, particularly in emergency healthcare scenarios, by assigning priority levels to different tasks. The PTS-RA model dynamically determines whether a task should be processed locally at hospital workstations (HWs) or offloaded to the Cloud, ensuring that critical healthcare tasks receive immediate computational resources. Comparing this approach to the Priority Equation, both models aim to reduce latency and improve task scheduling efficiency in computationally constrained environments; however, methodologies differ significantly. The PTS-RA system relies on heuristic-based task prioritization using urgency levels computed from patient health metrics, which operates primarily in a healthcare monitoring setting. In contrast, the Priority Equation is a more generalized framework applicable across diverse industrial IoT applications where real-time task prioritization is essential.

A key distinction between these approaches is their task execution decision mechanisms. The PTS-RA system uses a pre-defined heuristic model, where tasks are classified based on emergency levels, and decisions are made according to available bandwidth and computational resources. In contrast, the Priority Equation employs a dynamic queuing-based approach, continuously updating task allocation thresholds based on system load and latency constraints. The Equation allows a more adaptive and real-time response to varying task loads, ensuring system resources are optimally allocated without pre-defined emergency classifications. Another fundamental difference lies in the resource management strategy. The PTS-RA system distributes tasks between hospital workstations (acting as Edge servers) and the Cloud to prioritize critical tasks. The Priority Equation accounts for queue dynamics and processor availability, making it more robust for scalable and high-throughput environments, particularly in Supply Chain Management and industrial automation, where computational demand fluctuates dynamically.

Both models contribute to the advancement of real-time task scheduling in MEC environments, but their applicability differs. The PTS-RA system is well-suited for structured healthcare applications, where emergency-driven prioritization is necessary for medical decision-making. However, it may struggle to generalize to broader Edge computing environments with diverse computational loads. The Priority Equation, in contrast, provides a more flexible and scalable solution, making it ideal for real-time industrial IoT deployments, where task prioritization and latency optimization are essential.

Lastly, the study "*AI-Based Sustainable and Intelligent Offloading Framework for IIoT in Collaborative Cloud-Fog Environments"* by Kumar et al. (2023) introduces an artificial intelligence-driven framework for optimizing resource allocation in multi-layered Cloud-Fog architectures. The research emphasizes the significance of real-time offloading decisions to enhance Quality-of-Service (QoS) metrics such as execution time, energy consumption, and cost. The framework incorporates fuzzy-based offloading controllers and the Whale Optimization Algorithm (WOA) to intelligently assign tasks to the most suitable computational resources, minimizing latency and improving overall system efficiency. One of the primary contributions of this study is the introduction of an AI-enabled decision-making system that dynamically determines whether tasks should be executed locally, in a Fog node, or in a Cloud data center. This hierarchical decision-making model is particularly relevant for Industrial Internet of Things (IIoT) applications, where real-time responsiveness is crucial. The authors demonstrate that their approach improves execution performance, reducing makespan by 37.17%, energy consumption by 27.32%, and execution costs by 13.36% compared to traditional offloading techniques. However, the study does not explicitly address task prioritization mechanisms beyond QoS-driven optimization, leaving a gap where more refined dynamic prioritization techniques could be applied.

Comparing Kumar et al. (2023) work with the Priority Equation, a key distinction is the method of task prioritization and resource allocation. While the AI-based offloading framework uses metaheuristic algorithms (near-optimal solutions within a reasonable timeframe) and fuzzy logic (a mathematical framework that enables reasoning with imprecise, uncertain, or vague data) to enhance decision-making, the Priority Equation employs mathematical modeling rooted in Queuing Theory to dynamically adjust task offloading thresholds (s) based on real-time system conditions. The Priority Equation ensures that high-priority tasks receive preferential processing while balancing local Fog and Cloud computing capacities. Another key distinction is decision granularity. The AI-based framework primarily focuses on finding optimal resource allocations for individual task batches, making it effective for long-term system optimization. In contrast, the Priority Equation continuously adjusts offloading decisions in real time based on task arrival rates, service rates, and system utilization factors, ensuring dynamic adaptability in highly variable IIoT environments. The Priority Equation is more suitable for applications that require instantaneous decision-making and latency-sensitive operations, whereas the AI-based method is better suited for periodic adjustments driven by metaheuristic optimization algorithms.

While the Whale Optimization Algorithm (WOA) used in the AI framework significantly enhances task-to-resource mapping, it introduces computational overhead, requiring continuous model updates and convergence cycles. The Priority Equation avoids this computational burden by relying on lightweight mathematical computations, making it more efficient for real-time IIoT applications. However, integrating machine learning-based prediction models into the Priority Equation could allow for more adaptive threshold tuning, leveraging historical workload trends to anticipate system fluctuations.

The literature reviewed in this study highlights various methodologies for task scheduling, resource allocation, and latency optimization in Edge, Fog, and Cloud computing environments. A wide range of approaches, including heuristic models, artificial intelligence-driven frameworks, multi-criteria decision-making (MCDM) techniques, and optimization algorithms, have been examined in comparison to the Priority Equation. While these studies offer substantial contributions to computational efficiency and task management, each approach has distinct advantages and limitations. Across all methodologies examined, a key differentiator of the Priority Equation is its reliance on Queuing Theory and real-time prioritization rather than static optimization models or computationally intensive AI-based approaches. The Priority Equation ensures optimal latency-sensitive decision-making without requiring heavy computation by dynamically recalibrating task scheduling thresholds based on task arrival rates, service rates, and system utilization factors. Furthermore, while heuristic and AI-driven methods provide enhanced predictive capabilities, the Priority Equation lightweight mathematical foundation makes it more suitable for real-time Industrial IoT and supply chain applications where immediate processing decisions are necessary. Future research could explore hybrid models that integrate machine learning for dynamic threshold tuning while maintaining the real-time adaptability of queuing-based prioritization and/or incorporating Critical Path Analysis for cloud-based task execution, which could further refine the Priority Equation ability to allocate tasks to the most efficient computational resources across multi-cloud environments. By integrating queuing theory, real-time prioritization, and dynamic offloading thresholds, the Priority Equation provides a balanced solution that meets the needs of low-latency, high-throughput computing environments.

2.1 Theoretical Framework

Priority-based heuristics have been widely studied as effective strategies for task scheduling in latency-sensitive environments. According to Alatoun et al. (2022), in *"Advancements in Heuristic Task Scheduling for IoT Applications in Fog-Cloud Computing: Challenges and Prospects,"* extensive research has been conducted on priority-based task allocation. Researchers such as Fahad et al. (2022) and Tang et al. (2023) have explored both static and dynamic priority scheduling models to enhance the efficiency of task execution. Static priority scheduling assigns fixed priorities based on predefined criteria such as deadlines or importance, ensuring simplicity but lacking adaptability in dynamic environments. In contrast, Shi et al. (2020) described that dynamic priority scheduling adjusts task priorities in real-time, responding to fluctuating workloads and system conditions, thus offering increased flexibility at the cost of computational complexity. Another conceptual similarity of the Priority Equation is that it intends to optimize task allocation in real-time and resource-constrained environments. However, it diverges in methodology by employing a queuing-theoretic approach rather than heuristic-based task assignment. Unlike Multi-Queue Priority (MQP) scheduling proposed by Fahad et al. (2022), which dynamically adjusts preemption time slots to manage task starvation issues, the Priority Equation incorporates Queuing Theory principles to determine the likelihood of local versus offloaded task execution probabilistically. Task allocation is mathematically structured rather than relying on predefined heuristics or manually assigned priority levels.

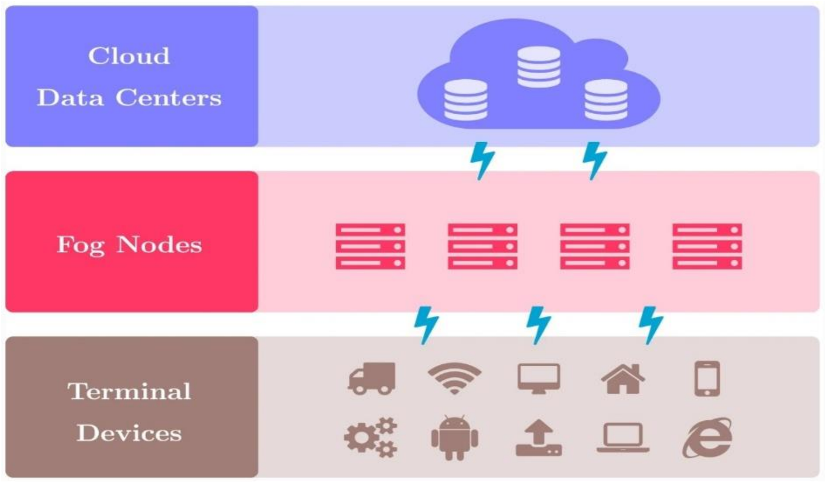
A key difference between heuristic-based models and the Priority Equation lies in their treatment of task execution dependencies. Studies such as Madhura et al. (2021) introduced directed acyclic graphs (DAGs) to account for task precedence constraints, ensuring that dependent tasks are scheduled efficiently. Similarly, heuristic methods such as the Opposition-Based Chaotic Whale Optimization Algorithm (OppoCWOA) approach proposed by Movahedi et al. (2021) employ optimization techniques such as the Whale Optimization Algorithm to balance computational workloads across fog computing nodes. While effective, these approaches require extensive computational tuning and iterative adjustments to optimize performance. In contrast, the Priority Equation integrates Poisson-based probability distributions and M/G/c queuing models, ensuring that latency-sensitive tasks are prioritized dynamically without requiring heuristic parameter fine-tuning.

Many heuristic approaches, such as those proposed by Hoseiny et al. (2021) and Choudhari et al. (2018), rely on task categorization based on pre-assigned priority levels. While effective in structured environments, this method risks inefficiencies when faced with unpredictable task arrivals and varying processing delays. The Priority Equation mitigates these inefficiencies by integrating real-time computational load monitoring, dynamically adjusting the offloading threshold based on workload conditions rather than fixed classification levels.

An enhancement introduced by the Priority Equation is its ability to integrate seamlessly into hybrid processing environments. Whereas heuristic models often operate within either Fog or Cloud computing paradigms, the Priority Equation dynamically shifts processing between Fog and Cloud nodes based on real-time latency constraints. This makes it particularly well-suited for applications such as Supply Chain Management, where fluctuating task loads necessitate an adaptive scheduling framework.

The Priority Equation derives from key Queuing Theory principles, probability distributions, and multi-server systems, building upon the mathematical framework established by foundational works such as Rényi (2007) and Willig (1999). The formula is designed to address real-time task prioritization and dynamic offloading in supply chain systems, particularly within Fog and Edge computing environments. Specifically, the **M/G/c** model accounts for multi-server systems with general service time distributions, while the task arrival process is modeled using Poisson processes (Last and Penrose, 2017), which provide a mathematically tractable representation of stochastic task arrivals. The integration of Poisson probability distributions into the Priority Equation ensures accurate modeling of task distributions in Fog and Cloud environments. The stationary and memoryless properties of the Poisson process enable efficient calculation of expected delays and prioritization logic in real-time decision-making. Furthermore, the integration of Campbell (1965) allows for precise computation of the expected number of tasks processed locally versus those offloaded, reinforcing the model's validity.

Figure 1: Three-Tier Architecture for **M/G/c** Queuing in Real-Time Supply Chain



(Ibrahim et al., 2022)

Similar to the work of Alatoun et al. (2022) in *"A Novel Low-Latency and Energy-Efficient Task Scheduling Framework for Internet of Medical Things in an Edge Fog Cloud System,"* the Equation shares foundational principles in system architecture. Both models utilize a three-layer structure, where the bottom layer consists of IoT sensors, the intermediate layer comprises Fog devices, and the top layer represents the Cloud. However, a key distinction lies in the prioritization mechanism. While Alatoun et al. (2022) classify tasks into *"normal, moderate, and critical"* based on medical parameters, the Priority Equation defines critical response times according to supply chain efficiency requirements. Supply chain operations require rapid data processing with optimal resource utilization, making real-time task prioritization essential. Alatoun et al. (2022) reference the earlier work of Cortés et al. (2015), which examined large-scale information flow in healthcare-related IoT applications, reporting that such systems can reach a processing rate of 25,000 records per second. While healthcare applications focus on patient monitoring and medical emergencies, Supply Chain Systems emphasize real-time logistics and operational efficiency. Both domains handle vast volumes of data where tasks must be efficiently distributed between local processing units and Cloud resources, ensuring time-sensitive computations are handled with minimal delay.

The Priority Equation aligns with existing research on edge-focused task offloading in resource-constrained environments. The work of Deng et al. (2018) in "Optimal Application Deployment in Resource-Constrained Distributed Edge" highlights the necessity of dynamically offloading tasks based on the real-time availability of Edge resources. Unlike conventional task distribution models that rely on pre-configured thresholds, the Priority Equation leverages real-time system conditions, particularly the utilization factor, to adjust offloading probabilities dynamically. This enables a probabilistic task distribution that prevents local node saturation while minimizing overall task response time.

The approach presented in *"Latency Minimization with Optimum Workload Distribution and Power Control for Fog Computing"* by Atapattu et al. (2020) contrasts with the Priority Equation's more fluid offloading mechanism. While the three-layer IoT-Fog-Cloud computing model assumes predefined computational capacities for each layer, the Equation adapts dynamically based on real-time task arrival rates, service rates, and utilization factors.

Summarizing, the Equation provides a structured, mathematically grounded alternative to heuristic-based scheduling models by integrating Queuing Theory, Poisson probability distributions, and real-time workload assessment. Unlike heuristic methods that require frequent tuning and optimization, the Equation ensures real-time task execution while maintaining low computational overhead, positioning it as an alternative to traditional heuristic-based scheduling methods in dynamic, high-demand computing environments. Its application in Supply Chain Systems could demonstrate the synergy between theoretical mathematics and practical implementation, offering improved latency management and task prioritization in fog-enabled IoT environments.

2.2 Key Benefits

The Priority Equation *reduces latency* by processing critical tasks closer to the data source, minimizing response times, and enabling immediate action for high-priority events, thereby supporting real-time decision-making. It *enhances resource efficiency* by offloading non-urgent tasks to the Cloud, optimizing resource utilization at the Edge, preventing local systems from overloading, and ensuring efficient task processing. Its dynamic prioritization and thresholding mechanisms enable *scalability*, allowing the system to adapt to growing data volumes and expand seamlessly within IoT-driven supply chain environments.

2.3 Nomenclature: Priority Formula for Real-Time Supply Chain

The following table defines the key parameters used in the Priority Equation, each playing a decisive role in optimizing task allocation and processing efficiency across Edge/Fog and Cloud computing layers. These parameters help determine whether tasks should be processed locally or offloaded, ensuring minimal latency, balanced workload distribution, and efficient resource utilization.

Symbol	Description
$T_{real\text{-}time}$	Real-time processing threshold identifies tasks that require immediate local processing (based on task urgency).
T	Task processing time. The time required to complete a given task determines if it should be processed locally or offloaded.
s	Offloading threshold for non-critical tasks. Tasks with $T < s$ are processed locally, while tasks with $T \geq s$ are offloaded.
λ	Task arrival rate. Represents the frequency at which tasks arrive in the system.
$\boldsymbol{E}[\boldsymbol{s^2}]$	Second moment of service time distribution. Reflects the variability in service times, with higher values indicating more variability.
c	Number of processors - providing the capacity to handle tasks per millisecond.
scaling_factor	A factor optimizes resource allocation. It can be adjusted when dynamic resource scaling is implemented.
ρ	Utilization factor, defined as $\boldsymbol{\rho} = \frac{\boldsymbol{\lambda}}{\boldsymbol{c}\mu}$. Measures the load on local processors; values close to 1 (one) indicate high utilization.
μ	Local service rate. The rate at which tasks are processed locally by each processor.
W_R	Expected remote response time. Accounts for the additional delay when tasks are offloaded to Cloud layers.
B	Batch size factor. Used in remote processing to handle larger batches of tasks efficiently.

2.4 The Priority Equation for Real-Time Supply Chain – Equation as per task

The Priority Equation serves as the foundation for dynamic task allocation in an IoT-driven supply chain environment, ensuring real-time processing efficiency while balancing computational loads across Edge, Fog, and Cloud layers. It operates by assessing key system parameters – such as task urgency, processing time, and resource availability – to determine whether a task should be processed locally or offloaded to a remote server. The Equation optimizes resource allocation, reduces latency, and prevents system overload by integrating factors like task arrival rate, service rate, processor utilization, and offloading thresholds. The variables parameters influence task prioritization, computational efficiency, and decision-making for real-time and non-critical workloads, forming the mathematical basis for optimizing supply chain operations.

$$\boldsymbol{Priority} = \boldsymbol{IF}(\boldsymbol{T} < \boldsymbol{Treal\_time}, \left(\frac{\lambda E[s^2]}{2(c \times scaling_{factor})(1-\rho)} + \frac{1}{\mu}\right), \boldsymbol{IF}\left(\boldsymbol{T} < \boldsymbol{s}, \left(\frac{\lambda E[s^2]}{2(c \times scaling_{factor})(1-\rho)} + \frac{1}{\mu}\right), \lambda \times \frac{WR}{B}\right)$$

2.5 Priority_Latency_Optimizer.py (Python)

As implemented in the pseudocode, the Priority Equation establishes a systematic approach to task prioritization and distribution within real-time supply chain environments. The algorithm dynamically determines whether a task should be processed locally at the Edge/Fog layer or offloaded to the Cloud to minimize latency while optimizing resource utilization. At its core, the model relies on fundamental principles of Queuing Theory, real-time thresholding, and dynamic workload distribution.

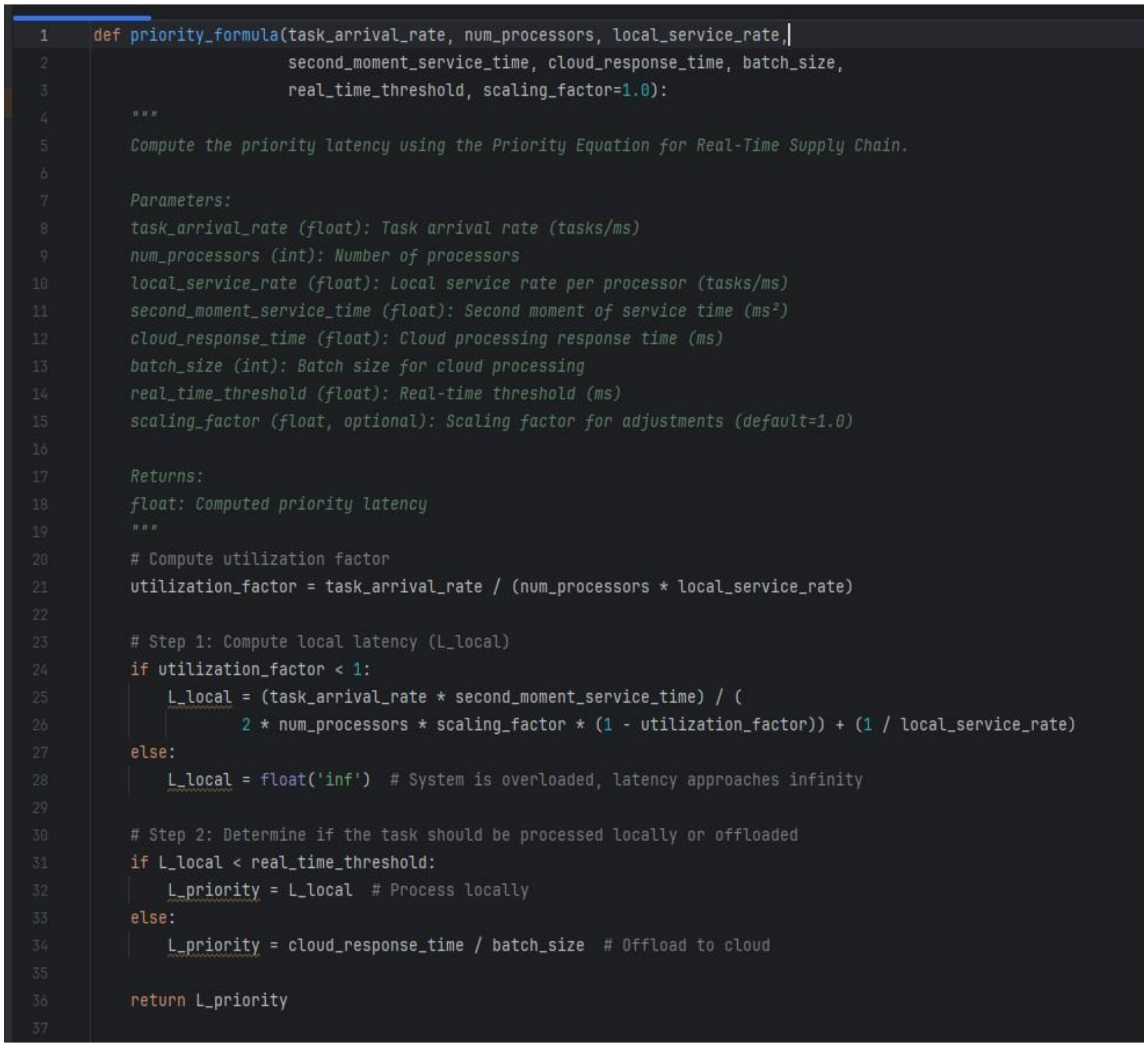
The first step involves computing the system's utilization factor, denoted as ρ, which represents the ratio of the task arrival rate to the total processing capacity of the available computational resources. This factor indicates system load, where higher values approaching one suggest that the system is nearing full capacity. If the system is not overloaded, the local processing latency, denoted as L_{local}, is computed using an equation derived from the M/G/c queuing model. This formulation incorporates the task arrival rate, the second moment of service time, the number of available processors, and a scaling factor to estimate the expected waiting time for task execution. If the utilization factor reaches or exceeds one, it signifies an overloaded state where processing delays become excessive, necessitating the offloading of tasks to external computing resources.

Following the computation of local processing latency, the algorithm determines whether a task should be retained within the local system or offloaded to the Cloud. This decision is based on a comparison between L_{local} and the predefined real-time threshold, $T_{real\text{-}time}$. If the estimated local processing latency is below this threshold, the task is processed locally to ensure timely execution. Conversely, if the latency exceeds the threshold, the task is offloaded to the Cloud, where the latency is governed by the Cloud response time divided by the batch size (B). This formulation ensures that Cloud offloading remains efficient, with larger batch sizes reducing per-task latency by distributing processing overhead across multiple tasks.

The final output of the algorithm is the computed priority latency, which reflects the task's expected processing time based on its assigned computational pathway. By dynamically adjusting processing decisions in response to system load conditions, the algorithm enhances overall system performance by maintaining a balance between local execution and Cloud offloading. This mechanism ensures that latency-sensitive tasks are given priority for immediate execution while non-critical workloads are efficiently managed to prevent system congestion.

This particular approach is relevant mainly in high-velocity data environments like IoT-enabled supply chains, where RFID sensors, barcode readers, and robotic systems generate thousands of computational tasks per second. The adaptive decision-making process embedded within the Priority Equation ensures that these tasks are processed in an optimized manner, preventing bottlenecks while maintaining real-time responsiveness. The implications extend beyond supply chain management to domains such as Smart Cities, Healthcare, and Autonomous Systems, where efficient workload distribution plays a critical role in ensuring system reliability and performance.

From a broader perspective, the Priority Equation introduces a structured methodology for balancing computational loads in distributed architectures, mitigating the limitations associated with cloud-dependent models. Intelligently leveraging Edge and Fog computing resources enables organizations to achieve lower latency, improved scalability, and reduced reliance on centralized infrastructures. Furthermore, future research should explore the integration of AI-driven predictive analytics into the equation to enhance its adaptability in complex and highly dynamic processing environments.


```python
def priority_formula(task_arrival_rate, num_processors, local_service_rate,
                     second_moment_service_time, cloud_response_time, batch_size,
                     real_time_threshold, scaling_factor=1.0):
    """
    Compute the priority latency using the Priority Equation for Real-Time Supply Chain.

    Parameters:
    task_arrival_rate (float): Task arrival rate (tasks/ms)
    num_processors (int): Number of processors
    local_service_rate (float): Local service rate per processor (tasks/ms)
    second_moment_service_time (float): Second moment of service time (ms²)
    cloud_response_time (float): Cloud processing response time (ms)
    batch_size (int): Batch size for cloud processing
    real_time_threshold (float): Real-time threshold (ms)
    scaling_factor (float, optional): Scaling factor for adjustments (default=1.0)

    Returns:
    float: Computed priority latency
    """
    # Compute utilization factor
    utilization_factor = task_arrival_rate / (num_processors * local_service_rate)

    # Step 1: Compute local latency (L_local)
    if utilization_factor < 1:
        L_local = (task_arrival_rate * second_moment_service_time) / (
                2 * num_processors * scaling_factor * (1 - utilization_factor)) + (1 / local_service_rate)
    else:
        L_local = float('inf')  # System is overloaded, latency approaches infinity

    # Step 2: Determine if the task should be processed locally or offloaded
    if L_local < real_time_threshold:
        L_priority = L_local  # Process locally
    else:
        L_priority = cloud_response_time / batch_size  # Offload to cloud

    return L_priority
```

2.6 Summary

The research explores the integration of Edge/Fog and Cloud computing to address challenges in IoT-driven supply chain systems. It highlights the limitations of traditional cloud-centric models, particularly concerning latency and resource efficiency, and proposes a Priority Equation for dynamic task allocation. Equation's primary purpose is to leverage Queuing Theory and task thresholding to optimize task prioritization, reduce latency, and improve scalability and resource utilization. The framework enhances real-time decision-making and operational efficiency by processing critical tasks locally and offloading non-urgent ones to the Cloud, particularly in industries requiring timely data handling, such as logistics, healthcare, and manufacturing. The study lays a foundation for scalable, responsive supply chains that align with the demands of modern digital environments.

CHAPTER III:
METHODOLOGY

3. Overview of the Research Problem

In the rapidly evolving landscape of IoT-driven supply chains, the increasing volume of data generated by connected devices presents significant challenges in task allocation, latency reduction, and resource management. Traditional task scheduling approaches struggle to balance computational loads efficiently across Edge/Fog and Cloud computing layers, often leading to processing bottlenecks and increased response times. In order to address these inefficiencies, it is essential to develop an adaptive mechanism that dynamically prioritizes and distributes tasks based on system conditions, workload demands, and real-time processing requirements, which leads to the central research problem:

"How can a dynamic, priority-based task allocation equation be developed to improve latency, manage task load, and enhance resource efficiency across Edge/Fog and Cloud computing layers in the Internet of Things-driven supply chain environments?"

3.1 System Architecture

Based on recent research on integrating Edge/Fog and Cloud computing for optimized task allocation in IoT environments by Nguyen et al. (2024), a new proposed system architecture enhances task prioritization and offloading by dynamically balancing computational workloads across multiple layers. The proposed system architecture for task prioritization and offloading consists of three main layers: *the IoT Devices or Data Sources, the Edge/Fog Layer, and the Cloud Layer*. This architecture is designed to dynamically prioritize and offload tasks based on urgency and processing requirements, optimizing latency reduction and resource utilization. The IoT Devices or Data Sources generate tasks in real-time, such as inventory tracking, fleet monitoring, or predictive maintenance alerts. These tasks vary in priority based on urgency (e.g., critical alerts versus non-urgent data). Each device is equipped to transmit data directly to the Edge/Fog layer for initial processing and prioritization. The Edge/Fog layer consists of multiple processing nodes (Fog nodes) positioned close to IoT devices to minimize latency. Fog nodes handle high-priority tasks requiring immediate processing, such as critical alerts. A local task queue is maintained at each Fog node, where tasks are initially evaluated based on predefined Real-Time Threshold ($T_{real\text{-}time}$) and Offloading Threshold (s) parameters. Tasks are processed locally at the Fog layer if their urgency and resource demands meet the local processing criteria; otherwise, they are offloaded to the Cloud.

The Cloud layer is for non-urgent tasks, where more extensive computational resources are available. The Cloud processes tasks that do not require real-time responses, such as trend analyses or data aggregation. The Cloud layer also acts as a backup in cases where Fog nodes reach their processing capacity, ensuring that tasks are still handled without compromising the overall system's efficiency.

3.2 Computational Model

The computational model relies on a queuing-based priority system with parameters that guide task distribution and prioritization based on urgency, processing capacity, and latency requirements.

A - The task arrival rate (*λ-lambda*)

The task arrival rate, denoted as λ, is a fundamental parameter in Queuing Theory and is used to model the frequency at which tasks arrive for processing within a system. In the context of fog-enabled IoT networks, the task arrival rate represents the number of tasks generated per unit of time by terminal devices (TDs), such as sensors, radio frequency identification (RFID), and embedded IoT devices. This rate is essential in determining system performance, as it directly affects latency, resource allocation, and overall efficiency of task processing within Fog and Cloud computing environments.

A widely accepted assumption in modeling task arrival rates within IoT-based systems is that task generation follows a Poisson process (Ibrahim et al., 2022). The Poisson distribution is commonly employed because it models independent arrivals over time, which aligns well with the behavior of IoT devices that generate tasks randomly based on user interactions, environmental changes, or scheduled triggers. The Poisson-based characterization of task arrival rates has been validated in various simulation studies (Bukhari et al., 2022; Ibrahim et al., 2022), demonstrating that dynamic offloading mechanisms can optimize task scheduling and minimize latency. These findings emphasize the importance of selecting an appropriate arrival rate λ based on real-time monitoring of system load and computational capacity. Furthermore, in intelligent task offloading models, such as the logistic regression-based framework proposed by (Bukhari et al., 2022), the arrival rate is dynamically adjusted based on historical data, ensuring that resource allocation remains optimal even under fluctuating workloads.

Within the framework of the Priority Equation, λ plays a critical role in determining system load and influencing task prioritization. A higher arrival rate indicates a more frequent influx of tasks, necessitating rapid assessment and prioritization to ensure efficient processing. An increase in λ may also demand greater utilization of local processing resources or more aggressive task offloading strategies to prevent congestion and maintain low latency. Effectively managing λ is essential for optimizing computational resource allocation and sustaining real-time performance in IoT-driven environments.

B - Processing Rates *(μ_{edge}, μ_{cloud})*

Each computational layer in a fog-enabled IoT system is characterized by its own service rate, denoted as μ, which defines the rate at which tasks are processed and completed. The service rate is crucial in determining the efficiency of task execution and directly influences system latency, queuing times, and overall computational performance. The value of μ varies across different computational layers, with the Cloud layer typically exhibiting the highest processing rate due to its extensive computing resources, while Fog and Edge layers prioritize real-time task execution at lower but more immediate processing capacities (Ibrahim et al., 2022).

Mathematically, the service rate μ is defined as the reciprocal of the average service time, expressed in tasks per second, which provides a quantitative measure of processing speed (Taha, 1998). A higher service rate corresponds to faster processing and lower queuing delays, making it a critical factor in performance optimization (Bukhari et al., 2022). The processing rate at the Edge layer (μ_{edge}) is often optimized to handle latency-sensitive tasks that require immediate responses, whereas the cloud processing rate (μ_{cloud}) is better suited for batch processing and computationally intensive tasks that do not have stringent real-time constraints.

The impact of μ on system performance is significant, mainly when evaluated alongside the task arrival rate λ. When μ is high relative to λ, tasks are processed efficiently with minimal delays. Conversely, if μ is low compared to λ, the system may experience congestion, leading to increased queuing times and performance bottlenecks. As demonstrated in queuing models applied to Fog computing, such as M/G/c systems, an optimal balance between λ and μ is essential to maintaining system stability and preventing excessive task accumulation (Ibrahim et al., 2022).

Moreover, dynamic adaptation of μ through virtualization techniques, such as adjusting the number of processors or virtual machines (VMs) in Fog environments, enhances flexibility and resource utilization. Research has shown that incorporating speedup factors in Fog computing, where each VM can scale its processing rate dynamically, improves task allocation efficiency and mitigates latency concerns (Ibrahim et al., 2022). The ability to allocate resources dynamically ensures that system performance remains optimal even under fluctuating task loads.

C - Processor count *(c)*:

The processor count, denoted as c, is a fundamental parameter in Queuing models used to characterize task response time in Fog-enabled IoT networks. In the context of Fog and Cloud computing, c represents the number of parallel processing units or virtual machines (VMs) available to handle incoming tasks. This parameter directly influences the system's capacity to process tasks efficiently, reducing latency and mitigating congestion in high-load scenarios.

The inclusion of c in priority-based task scheduling models aligns with the M/G/c queuing framework, where multiple servers operate in parallel to handle incoming tasks (Ibrahim et al., 2022). The performance of a fog-enabled system is significantly impacted by c, as increasing the number of processing units enhances system throughput and decreases waiting times. The mathematical relationship between the number of processors and task latency is evident in Queuing Theory (Willig, 1999), where the expected waiting time is inversely proportional to c. Specifically, in M/G/c systems, the expected delay for tasks depends on both the arrival rate λ and the processing rate μ, adjusted by c, ensuring that task allocation remains efficient under varying loads (Ibrahim et al., 2022).

In practical implementations, the choice of c depends on the computational demands of the system. For instance, in Fog environments where latency-sensitive applications require immediate responses, increasing c ensures that tasks are processed with minimal queuing delays. The study by (Bukhari et al., 2022) emphasizes the role of dynamic scaling, where the number of active processors is adjusted based on real-time system load, optimizing resource utilization while maintaining service quality. This dynamic adjustment prevents resource underutilization during low-load periods and ensures adequate processing power during peak loads.

From a system optimization perspective, balancing c, λ, and μ is critical for maintaining an efficient computing environment. If c is too low relative to λ, task queues accumulate, leading to increased response times and potential system congestion. Conversely, an excessively high c can lead to underutilization of resources, reducing overall efficiency. Therefore, dynamic provisioning of processing units, guided by real-time monitoring and predictive analytics, is essential in achieving an optimal balance in Fog and Cloud-integrated architectures.

D - Utilization Factor *(ρ-rho)*:

The utilization factor, denoted as ρ, is a critical parameter in queuing models that determines the efficiency and stability of task processing in Fog-enabled IoT networks. It is defined as the ratio of the task arrival rate λ to the total service capacity, given by the product of the number of processors c and the service rate μ, expressed mathematically as:

$$\rho = \frac{\lambda}{c\mu}$$

This metric quantifies the fraction of time that processing resources are occupied. When ρ is low, the system experiences minimal congestion, leading to reduced queuing delays. However, as ρ approaches unity, the system reaches saturation, resulting in increased waiting times and potential task loss if buffer capacities are exceeded (Ibrahim et al., 2022).

In priority-based task scheduling, ρ plays a crucial role in determining the feasibility of real-time processing. A well-balanced system maintains ρ at an optimal level to prevent bottlenecks while ensuring efficient resource utilization. In M/G/c queuing models widely used in Fog computing architectures, a system is considered stable if $\rho<1$, meaning that the service rate is sufficient to handle incoming tasks without indefinite queuing (Willig, 1999). Furthermore, dynamic task offloading mechanisms utilize ρ as a decision metric to distribute workloads efficiently between local Fog nodes and Cloud resources. If ρ exceeds a predefined threshold, indicating high congestion, tasks are offloaded to Cloud servers to prevent excessive queuing delays. Conversely, when ρ is low, more tasks are processed locally to optimize real-time performance (Bukhari et al., 2022).

The significance of ρ extends to latency optimization strategies in Fog computing. Research has demonstrated that adjusting c and μ in response to fluctuations in λ can enhance system adaptability, maintaining an optimal ρ range that minimizes queuing time while maximizing throughput (Ibrahim et al., 2022). By integrating ρ into Priority Equations, modern scheduling algorithms can dynamically adjust processing thresholds, ensuring efficient task allocation across computational layers.

An example from the logistics network is parcels arriving at a distribution center, which needs sorting, labeling, and dispatching tasks. If parcels arrive frequently (λ is high), the distribution center's local processors (e.g., five handling stations with a rate μ) might struggle to keep up, resulting in a high utilization factor ρ. For instance, if ρ reaches close to 1 (indicating full utilization of resources), parcels may face delays in processing. In order to avoid an overload situation, the system may offload lower-priority tasks (e.g., handling non-urgent packages) to a remote processor (like cloud-based planning systems) or delay them until resources become available. Conversely, if ρ is low (indicating under-utilization), more parcels can be processed locally, allowing for faster turnaround and enhanced real-time decision-making.

E - Threshold *(s)*:

The offloading threshold, denoted as s, is a fundamental parameter in fog-enabled IoT network priority-based task scheduling model. It defines the boundary between locally processed tasks and those offloaded to a remote processing unit, such as a Fog node or Cloud server. The threshold s is crucial for managing computational loads efficiently and minimizing latency by ensuring real-time and latency-sensitive tasks are processed closer to the source while less urgent tasks are offloaded for remote execution (Ibrahim et al., 2022).

Mathematically, the offloading decision is modeled using a Queueing-Theoretic approach, where tasks arrive following a Poisson process (Last and Penrose, 2017) and have service times that follow a generally distributed. The probability that a task is processed locally is given by:

$$a = 1 - e^{-\lambda s}$$

where λ represents the task arrival rate, the equation quantifies the proportion of tasks handled locally based on the defined threshold “s” (Ibrahim et al., 2022). If the execution time of a task exceeds s, it is offloaded to the Cloud, thereby balancing computational resources and avoiding excessive congestion at local processing units.

The choice of s directly impacts system performance, as a higher threshold increases the number of locally processed tasks, reducing the overhead associated with task transmission. However, an excessively high s value can lead to local resource saturation, resulting in increased queuing delays. Conversely, a lower s value promotes task offloading, alleviating local congestion but potentially increasing network transmission delays and cloud queuing times (Ibrahim et al., 2022).

Empirical studies in Fog computing have demonstrated that an optimal s value must be dynamically adjusted based on real-time system conditions. It is particularly relevant for adaptive offloading mechanisms considering varying workload conditions, resource availability, and energy consumption constraints (Bukhari et al., 2022). Research has shown that dynamically optimizing s improves response times and enhances overall system efficiency by ensuring critical tasks are processed within latency constraints while optimizing resource utilization across Fog and Cloud layers (Bukhari et al., 2022).

An example from the supply chain is the monitoring system of warehouse temperatures with IoT sensors. If the temperature spikes to a critical level (e.g., due to a cooling failure), an urgent alert needs immediate action to prevent spoilage. Here, the processing time $T_{real\text{-}time}$ for the alert is short and falls below *s*, so it is processed locally at the Fog level for a quick response. In contrast, regular temperature readings, which are not as time-sensitive, are sent to the Cloud for analysis and storage since their processing time exceeds "*s*".

F - Priority Parameters:

The parameter $T_{\text{real-time}}$ represents the upper time limit within which tasks must be processed locally to meet real-time constraints. It plays a role in determining whether a task is retained for local execution or offloaded to a higher processing tier, such as the Cloud. The selection of $T_{\text{real-time}}$ directly impacts system latency, ensuring that critical tasks receive immediate processing while balancing computational loads across different layers of the network (Ibrahim et al., 2022).

In fog-enabled IoT systems, task arrival follows a Poisson process (Last and Penrose, 2017), and service times are generally modeled using exponential or truncated exponential distributions. Tasks exceeding $T_{real\text{-}time}$ are offloaded to avoid local congestion, whereas tasks within the $T_{real\text{-}time}$ threshold are prioritized for local execution to minimize end-to-end latency. The decision to process a task locally or offload it is governed by a Queuing Theoretic approach, where real-time tasks are retained at the Edge while non-real-time tasks are distributed across Fog and Cloud resources – the threshold $T_{real\text{-}time}$ must be set dynamically to adapt to fluctuating workloads and varying system conditions. If the threshold is too high, excessive tasks remain in the local queue, potentially causing bottlenecks and increasing processing delays. Conversely, if it is set too low, tasks may be unnecessarily offloaded, incurring additional transmission delays and increasing reliance on Cloud resources. The study by (Ibrahim et al., 2022) emphasizes the importance of adaptive thresholding, where machine learning or predictive models are employed to adjust $T_{real\text{-}time}$ based on system congestion and task urgency dynamically.

G - Latency and Resource Allocation:

The local latency component, L_{local}, is crucial in optimizing resource allocation within Fog/Edge computing environments. Local latency represents the time required to process tasks within the local computational layer, significantly impacting real-time applications. Efficient resource allocation in Fog and Edge computing minimizes L_{local} by dynamically distributing computational workloads between local servers and offloading mechanisms. According to research on Fog-based resource allocation, the efficient assignment of computing resources ensures that tasks with strict latency constraints remain within the local processing environment, while less urgent tasks can be offloaded to the Cloud (Rezaee et al., 2024). The allocation of processing capacity at the Edge requires an optimal balance between task processing speed, energy consumption, and system throughput, which directly influences the efficiency of local latency management (Alenizi and Rana, 2021).

H - Task Prioritization Mechanism:

The prioritization of tasks is primarily governed by their urgency, execution requirements, and resource constraints, and it is implemented through structured queuing and scheduling strategies. In Fog and Cloud computing architectures, priority-based scheduling is a well-established method to manage latency-sensitive applications, ensuring that critical tasks receive processing priority over less urgent ones (Alatoun et al., 2022).

One approach to task prioritization involves a hierarchical scheduling mechanism where tasks are categorized into different priority levels. High-priority tasks, such as real-time monitoring applications in IoT systems, are processed at the Edge/Fog layer to minimize latency, while lower-priority tasks are deferred to Cloud computing resources, where processing power is superior but response times are typically longer (Jamal and Muqeem, 2023). The execution of tasks in such environments is often governed by a queueing model where tasks are dynamically assigned to available processors based on their priority rank and resource availability.

A significant enhancement in task prioritization mechanisms is the integration of adaptive scheduling algorithms, which allow real-time adjustments based on system load, network congestion, and processing capability. These adaptive systems employ decision-making algorithms that dynamically adjust the priority weight of tasks depending on network conditions and workload distribution (Ibrahim et al., 2022). Additionally, machine learning-based techniques, such as reinforcement learning, have been explored to optimize task scheduling by predicting task execution times and adjusting priorities accordingly (Jamal and Muqeem, 2023).

Furthermore, the effectiveness of task prioritization is also influenced by resource availability and utilization. Studies have shown that efficient resource mapping strategies, such as priority-based dynamic resource allocation, can significantly improve system performance by balancing workload distribution across Fog and Cloud nodes (Alatoun et al., 2022). By incorporating such mechanisms, the Priority Equation ensures that critical tasks are processed with minimal latency while optimizing computational efficiency for all other tasks. The core idea behind this approach is to prioritize tasks based on urgency and execution constraints. Tasks that must meet strict time constraints are processed locally, ensuring low latency, while tasks with more flexible deadlines are allocated to remote computing resources. The Equation provides a scalable and adaptive solution for modern IoT-based supply chain environments where fluctuating workloads require dynamic resource allocation. The model is particularly beneficial for latency-sensitive applications such as real-time inventory management, predictive maintenance, and automated logistics, where processing delays can significantly impact supply chain performance. Combining queuing models with adaptive prioritization ensures higher system responsiveness, improved efficiency, and balanced resource utilization across Fog and Cloud layers.

3.3 Experimental Setup

The latency associated with offloading IoT tasks to the Cloud is critical in Supply Chain Operations and real-time applications. In traditional cloud-based models, tasks generated by IoT devices are transmitted to the Cloud for processing, introducing significant transmission delays due to network congestion, bandwidth limitations, and cloud server response times. The transmission latency increases as more tasks are offloaded to centralized cloud servers, which can negatively impact applications requiring real-time processing (Bukhari et al., 2022).

The offloading processing time is given by:

$$P_{offload} = T_{transmit} + T_{queue} + T_{execute}$$

where $T_{transmit}$ represents the transmission delay from IoT devices to Cloud servers, T_{queue} accounts for queuing delays in the Cloud environment, and $T_{execute}$ is the actual task execution time on the Cloud server. Studies indicate that Cloud offloading results in higher latency due to network dependencies, as Cloud servers are often geographically distant from IoT devices, leading to increased transmission and queuing delays (Bukhari et al., 2022; Malik et al., 2022). Transmission delays depend on factors such as network bandwidth, data packet size, and routing complexity (Almutairi and Aldossary, 2021). Furthermore, once offloaded, tasks experience queuing delays before execution, which are influenced by:

- λ (task arrival rate) – the frequency at which new tasks arrive at Cloud processing units.
- W_R (expected remote response time) – the total delay from network transmission ($T_{transmit}$) and Cloud execution delays.
- B (batch size factor) – task are often processed in batches to optimize system throughput, reducing per-task latency.

The batch-based queuing delay due to the Cloud processing effect is modeled as follows:

$$\frac{\lambda WR}{B}$$

where dividing by B accounts for task parallelization benefits in bulk processing (Bukhari et al., 2022). Fog computing has emerged as a viable alternative to mitigate Cloud-induced delays, enabling proximity-based task execution that significantly reduces transmission time (Almutairi and Aldossary, 2021). Fog nodes allow the processing of time-sensitive tasks locally, minimizing dependency on remote Cloud infrastructures. The incorporation of dynamic offloading thresholds further optimizes latency-sensitive applications, ensuring real-time responsiveness in supply chain networks (Bukhari et al., 2022).

In supply chain networks, a larger capacity processor *(c)* enables parallel task execution, reducing queuing congestion before offloading to the Cloud (Khinchin et al., 2013). Maintaining an optimized processor-to-task ratio is essential in real-time applications, where delays must remain within strict operational constraints.

The *scaling_factor* functions as an adaptive control mechanism in task scheduling models. In Fog and Cloud computing, computational resources experience fluctuations in demand, necessitating dynamic workload balancing. The inclusion of scaling_factor in the denominator of queuing-based models reflects its role in enhancing local service efficiency (Khinchin et al., 2013). Higher values of scaling_factor improve computational performance, aligning with workload-aware scheduling frameworks.

Finally, the term $(1-\rho)$ represents idle capacity in queuing systems. As $\rho>1$, the system experiences congestion, causing exponential increases in latency (Ibe, 2013). The impact of high utilization necessitates dynamic resource allocation strategies, such as scaling processor availability (c) to accommodate increased workloads and adaptive offloading thresholds to optimize task distribution between local, Fog, and Cloud layers.

Little’s Law, which relates queue length to response time, underscores that high “ρ” leads to exponentially increasing waiting times, reinforcing the importance of balanced task scheduling (Ibe, 2013).

The Priority Equation for Real-Time Supply Chain offloads tasks to the Cloud only when tasks exceed the threshold “s”, which keeps non-critical tasks from adding latency to local critical tasks. Tasks are processed closer to the source, with critical tasks prioritized for real-time processing.

$$\boldsymbol{Priority} = (\frac{\boldsymbol{\lambda}\mathbf{E}[\boldsymbol{s}^{\mathbf{2}}]}{\mathbf{2}\boldsymbol{c} \times \boldsymbol{scaling\_factor}(\mathbf{1} - \boldsymbol{\rho})} + \frac{\mathbf{1}}{\mu})$$

By introducing $\frac{\mathbf{1}}{\boldsymbol{\mu}}$ in the Priority Equation, it ensures that real-time supply chain tasks are processed efficiently at the Fog/Edge layer, reducing unnecessary offloading to the Cloud. This term directly contributes to latency minimization, making it a key factor in balancing computational workloads between local and remote execution environments. The integration of $\frac{\mathbf{1}}{\boldsymbol{\mu}}$ enhances the model's adaptability, improving supply chain efficiency while maintaining real-time task execution requirements (Khinchin et al., 2013).

The differences between locally processed tasks and those sent to the Cloud affect latency. On the other hand, the Priority Equation calculates the priority level of tasks based on arrival rates and processing factors, which helps decide whether tasks should be handled locally or offloaded. It guides task prioritization to ensure that high-importance tasks receive processing attention when system resources are limited by including λ to scale the entire equation, emphasizing the role of arrival rates in determining its priority. High arrival rates inherently increase the calculated priority, which helps dynamically adjust the urgency of tasks based on the system load.

The comparison components of latency and task handling capture the delay incurred by offloading tasks from IoT devices to the Cloud, a standard approach in traditional, cloud-centric supply chains. It represents the time data travels from the Edge to the Cloud, whereas the Priority Equation uses similar Queuing and processing components but focuses on prioritizing tasks rather than directly calculating latency. The Equation does not focus on individual task delays but rather on dynamically prioritizing tasks based on load and processing conditions, adjusting priorities to avoid latency for high-priority tasks.

3.3.1 Hardware and Software Environment

The simulation experiments were conducted on a personal computing environment to evaluate task prioritization and latency optimization in an IoT-driven supply chain system. The specifications of the computing environment are as follows:

Hardware Configuration

- Operating System: Microsoft Windows 11 Home (Version 10.0.26100, 64-bit)
- Processor: Intel Family 6 Model 158 (Intel Core, ~1700 MHz)
- Cores & Threads: 1 Processor installed (limited to 1 core)
- Total RAM: 12 GB DDR4
- Available RAM: 4.45 GB at the time of testing
- Storage: Not explicitly mentioned, but assumed to be an SSD/HDD setup
- BIOS Version: AMI F.12 (06/24/2020)

Networking Setup

- Active Network Interface: Realtek RTL8821CE 802.11ac PCIe Adapter (Wi-Fi)
- VPN Adapters: NordVPN TAP & OpenVPN interfaces (disabled during simulations)
- Ethernet Adapter: Realtek PCI(e) (Media Disconnected)
- IPv4 Address: 192.16x.x.xx (for internal networking)
- DNS & DHCP: Managed by local router

The system runs on a 1-core processor at 1.7 GHz, which limits the ability to run highly parallelized simulations. However, it is sufficient for mid-scale task prioritization models. If larger datasets were used, computations may have been offloaded to an external server or cloud service. The available RAM of 4.45 GB at runtime means that simulations were optimized to avoid excessive memory usage. Large-scale experiments may require batch processing or cloud execution and the Wi-Fi connection was used for cloud-based simulations.

3.4 Conditions of Applying The Equation for Real-Time Supply Chain

The Priority Equation for Real-Time Supply Chain has the following conditions $IF(T<T_{real\text{-}time})$ and $IF(T<s)$ play crucial roles in determining the urgency of tasks and whether they should be processed locally or offloaded to the Cloud computing resources.

IF($T<T_{real\text{-}time}$) means real-time processing thresholds, and it is a strict threshold, typically set to a low value, e.g., 50 milliseconds, based on research done by Nawaz et al. (2021), identifying tasks that require immediate, near-instantaneous processing. Tasks deemed high priority and time-sensitive require local processing (at the Edge/Fog level) to meet the stringent time requirement. When $T<T_{real\text{-}time}$ meets the *"True"* criteria, the equation prioritizes immediate local processing, ensuring that critical, real-time tasks do not experience delays from offloading or queuing. This threshold is essential for tasks where any delay (such as transmission to the Cloud) could compromise system responsiveness or affect outcomes.

IF($T<s$) is a secondary threshold that identifies tasks suitable for local processing but less urgent than those needing real-time response. When a task's processing time T is less than s (but greater than $T_{real\text{-}time}$), the Formula directs the task to be processed locally rather than offloaded to the Cloud, as local processing minimizes latency. However, this

condition is less strict than the real-time condition, allowing for some flexibility. This threshold identifies tasks that are less time-sensitive than those under $T_{real\text{-}time}$ conditions but where local processing is still advantageous. If $T<s$ is *"Within,"* tasks are processed locally unless resources are highly constrained. Otherwise, tasks that do not meet this threshold are offloaded to the Cloud, where they can be managed with higher latency but greater resource availability.

3.5 Justification of Parameters

The task arrival rate λ represents the frequency at which IoT devices generate tasks or signals that require processing (sensors, tracking devices, machine monitoring systems, etc.) It directly influences system utilization, latency, and task distribution. Lambda (λ) affects the system load by calculating the utilization factor $\boldsymbol{\rho = \frac{\lambda}{c\mu}}$, which indicates the proportion of the system capacity being used. In $L_{priority}$ a higher λ increases the system workload, potentially leading to a higher latency. A low λ means fewer tasks arrive, and the system operates under capacity, leading to lower latency. When λ is high, more tasks arrive and potentially can exceed processing capacity, causing higher queue delays and offloading to the Cloud. In the healthcare IoT, the usual rate is between 100 to 500 tasks/second (Shukla et al., 2017) through wearable devices and patient monitoring systems (*for the research, all seconds are transformed into milliseconds*). In smart cities and pollution monitors, the rate is between 500 to 2,500 tasks/second (Shukla et al., 2017), whereas in the Industrial Internet of Things (IIoT), the rate increases to 5,000 to 10,000 tasks/second (Shukla et al., 2017), usually seen in assembly line sensors, robotics, and predictive maintenance.

The processor count *c* represents the number of processors available locally, such as at the Edge/Fog layer, that can handle tasks before they are offloaded to the Cloud. More processors increase the system's ability to process multiple tasks in parallel, improving performance and reducing latency. Based on the utilization factor $\boldsymbol{\rho = \frac{\lambda}{c\mu}}$ a higher c (processor count) decreases ρ (utilization factor), reducing the likelihood of system overload and allowing the system to handle a greater task load without queuing or delays. With more processors, the system can handle more tasks per unit of time, which means an improved throughoutput of the system. Another aspect is a decrease in latency as tasks can be distributed among processors (c – number of processors increase), directly reducing wait times and queuing. Lastly, adding more processors c allows the system to scale (system scalability) by increasing task arrival rate λ. In large-scale Industrial Internet of Things (IIoT) systems, the number of processors must grow in proportion to task load to maintain performance.

In the context of the Priority Equation, the number of processors (c) plays a crucial role in reducing queuing delays. However, diminishing returns occur as adding more processors provides lesser performance improvements beyond a certain point. If the task arrival rate (λ) is low or the system is under-utilized (ρ close to zero), increasing c (processor count) offers limited benefits. Since additional processors increase costs, the optimal value of c balances performance gains and cost efficiency. Scaling c becomes essential when the task arrival rate exceeds the system's processing capacity ($\mu \cdot c$), especially in high-throughput IIoT systems or environments handling large data volumes. The service rate μ represents the rate at which each processor can complete tasks.

The *second moment of service time* is often denoted as $\boldsymbol{E}[\boldsymbol{s}^{\mathbf{2}}]$ is a measure from Probability Theory (Rényi, 2007) and Queuing Theory (Ibrahim et al., 2022; Willig, 1999) that provides insight into the variability in service times. The service time s is the time required to process a task at a server (e.g., Edge/Fog nodes or Cloud). It captures both the average service time and its variability. A higher $\boldsymbol{E}[\boldsymbol{s}^{\mathbf{2}}]$ indicates greather variability in service time, leading to longer queues and higher latency whereas a lower $\boldsymbol{E}[\boldsymbol{s}^{2}]$ sugest consistent processing time, improving system predictability. The values chosen for the calculation are *0.1* and *1.0* since, in many practical applications (Willig, 1999), these values are used to simplify calculations and are reasonable for systems where tasks take longer to process or have more variability (network latency, hardware performance, resource contentions). For complex data tasks (data aggregation or analytics), service times are more variable, leading to larger $\boldsymbol{E}[\boldsymbol{s}^{2}]$ values. For low-variance workloads such as barcode scanning, RFID readings, and sensor data collection in IoT-based supply chain operations, using $\boldsymbol{E}[\boldsymbol{s}^{2}]$ is a reasonable approximation due to their near-deterministic service times with minimal variance. However, for systems handling mixed workloads that include high-variance tasks, a higher $\boldsymbol{E}[\boldsymbol{s}^{2}]$ value may be necessary to capture processing variability accurately. Early Queuing Theory (Willig, 1999) often uses $\boldsymbol{E}[\boldsymbol{s}^{2}]$=1.0 as default or baseline, reflecting systems with moderate variability.

Table 2: Calculated $\boldsymbol{E}[\boldsymbol{s}^{2}]$ vs Approximated Values in Task Allocation Models:

Scenario	Use Calculated $\boldsymbol{E}[\boldsymbol{s}^{2}]$	Use Approximate $\boldsymbol{E}[\boldsymbol{s}^{2}]$
High-precision modeling	✓	x
Known consistent service times	✓	x
High-task variability	x	✓
Lack of detailed service time data	x	✓
Complex tasks with outliers	x	✓
Early-stage system design	x	✓

The offloading threshold "*s*" is a based threshold, typically measured in milliseconds, which governs the decision to offload a task. The offloading threshold *s* determines whether a task should be processed locally (e.g., at the Edge/Fog layer) or offloaded to a remote processing resource like the Cloud. It represents the maximum tolerable local queuing and processing delay for tasks before they are considered for Cloud processing. The value of *s* for Process and Factory Automation is roughly 100ms (Ma et al., 2019), similar to telesurgery based on findings by Nankaku et al.(2022).

Table 3: Latency Requirements Across Different Industries

Industries	Milliseconds
Smart cities	50-150
Healthcare IoT	10-50
Autonomous robots	10-25
Drones	10-50
Industrial IoT (IIoT)	20-100
Smart agriculture	100-300

When $T<s$ tasks are processed locally on the Edge/Fog, and when $T \geq s$ are offloaded to the Cloud. This approach can use a fixed threshold of s=150 ms (Puleri et al., 2016) to determine task allocation. It ensures that tasks requiring shorter processing times are prioritized for local handling at the Edge/Fog, leveraging their proximity to the data source. Tasks exceeding this threshold are strategically offloaded to the Cloud, preventing overburdening local resources while maintaining availability. This fixed-threshold approach simplifies task distribution, optimizing system performance and effectively balancing the load across computing layers.

In many real-time systems, $T_{real\text{-}time}$ defines the maximum allowable delay for a task to be considered real-time. Setting $T_{\text{real-time}}$ between 1ms to 150ms means any task that requires completion within $T_{\text{real-time}}$ threshold will be classified as real-time (Puleri et al., 2016). This threshold is used in IoT industries like Supply Chains, Robotics, and Autonomous Systems. The Real-Time threshold ($T_{real\text{-}time}$) is a strict predefined time limit to identify tasks that require immediate processing to meet stringent latency requirements.

This threshold ensures that high-priority tasks critical to system performance or outcomes are processed locally (e.g., at the Edge or Fog layer) without delay. It means that tasks that meet the condition $T < T_{real\text{-}time}$ are handled immediately and locally to avoid any latency introduced by offloading (e.g., transmission delay or remote queuing), and it prioritizes the allocation of local resources (processors) for tasks that cannot tolerate delays, ensuring high-priority tasks are not affected by non-critical workloads. While $T_{real\text{-}time}$ is stricter and targets the most urgent tasks, the offloading threshold s serves as a secondary condition to identify tasks suitable for local processing but less critical than real-time tasks.

The remote response time, or W_R, is the average time required for a task to be offloaded to a remote computing resource, such as the Cloud or a centralized server. It captures the delay incurred due to the queuing, processing, and communication involved in handling tasks at a remote location. A higher W_R indicates longer delays for tasks offloaded to the Cloud, which can negatively impact time-sensitive operations. In contrast, a lower W_R indicates more efficient processing at the remote resource, which is favorable for handling non-critical tasks without significantly affecting overall system performance. In the Priority Equation, W_R is included in the remote delay, appearing in the second case of the Equation when task processing time exceeds the real-time threshold $T_{real\text{-}time}$. The offloading component $\frac{\boldsymbol{WR}}{\boldsymbol{B}}$ reflects the average response time per batch for tasks processed at the Cloud level. Lower W_R (via faster Cloud services or reduced transmission delays) improves offloading performance. W_R influences task distribution between local and Cloud processing, guiding the system to offload only when local processing is infeasible, which is optimal for dynamic adjustments. A higher W_R can create bottlenecks in the system, making monitoring and minimizing this parameter for high-throughput environments critical.

Transmission delay ($T_{transmit}$) is the time taken to send data to the remote system (Cloud server), and it is influenced by bandwidth, latency, and distance.

$$\boldsymbol{Ttransmit = \frac{Data\ size\ (bits)}{Network\ Bandwith\ (bits\ per\ second)}}$$

For this case, the Cloud or Remote response time ($T_{process}$) is the time taken for remote servers to process the task or batch, and it is modeled using Quesing Theory (Willig, 1999) $T_{process} = \frac{1}{\mu cloud}$; where μ_{cloud} is the service rate of the Cloud system.

Table 4: Measured Cloud Response Times Across Major Service Providers

Ping times	Milliseconds
Amazon Web Services	175
Microsoft Azure	229
Google Cloud	177
Digital Ocean	174
Linode	173
Alibaba Cloud	376

*For the purpose of this research, **Azure's ping** time is utilized as a benchmark to calculate latency, providing a consistent reference point for analysis.*

The Batch Size (*B*) refers to the number of tasks processed together as a single group or batch at a remote location, such as the Cloud or Fog computing node. Batching is a common optimization technique used to improve the efficiency of processing non-critical tasks (Zhang et al., 2019) by reducing the overhead associated with individual task handling. The batch size *B* reduces the impact of remote queuing delays by dividing the remote response time W_R across the number of tasks in the batch. Batch processing allows the system to handle multiple tasks simultaneously, improving efficiency in scenarios where individual task processing incurs significant delays or overhead. When tasks are offloaded to the Cloud $\frac{\boldsymbol{WR}}{\boldsymbol{B}}$ represents the average latency per task in the batch. A larger B (batch) reduces the per-task latency because W_R (the remote response time) is shared among tasks. Implementing batch processing improves efficiency by grouping tasks, which reduces the overhead of initiating multiple transmissions or processing requests. It helps with network optimization; fewer transmissions mean less network congestion, which is especially important in environments with limited bandwidth. Lastly, it reduces latency per task by dividing W_R by B, ensuring that the average latency per task decreases as batch size increases.

Selecting the appropriate batch size (B) involves optimizing system performance to meet the application's requirements. The decision is influenced by factors such as the task arrival rate, real-time constraints, and the system's throughput demands. For systems with strict real-time parameters, a smaller batch size is preferable to minimize delays associated with batch filling. Conversely, in scenarios where high throughput is prioritized, a larger batch size is advantageous as it enhances efficiency and reduces the overhead associated with processing individual tasks. In the context of the Priority Equation, the benchmark suite's stream workloads, derived from real-world IoT observations in Smart Cities, highlight task arrival rates ranging from 500 to 10,000 messages per second (Shukla et al., 2017) and analysis based on data collected mean of 17,817 batch size. These varying rates and diverse frequency distributions underscore the need for dynamic prioritization and adaptive processing strategies to ensure efficient resource allocation and real-time response, particularly in high-throughput environments.

Table 5: Task Arrival Rates and Batch Size Considerations Across IoT Environments

Dataset	Task Arrival Rate (tasks/ms)	Batch Size (B) (ms)	IoT Application	Rationale for Batch Size
FIT Dataset	500	50-250	Fitness tracker and health monitoring.	Small batches minimize delays for real-time health alerts while maintaining system efficiency.
NYC Taxi Dataset	4,000	1,000 – 2,000	Urban transportation and fleet tracking	Moderate batch size balance latency and throughoutput for near real-time tracking.
Sence Your City (CITY)	5,000	500 – 2,500	Smart city monitoring (air quality, noise levels).	Smaller batch sizes ensure real-time responsiveness while balancing throughoutput.
GRID Dataset	10,000	5,000 – 10,000	Smart grid energy management	Larger batches optimize throughoutput for high-volume data streams while reducing per-task overhead.

(Shukla et al., 2017)

The *scaling_factor* was determined using Pairwise Comparison Principles and Priority Weighting Methods found in Saaty's (1977) work on hierarchical structures *("A Scaling Method for Priorities in Hierarchical Structures").* According to the Engineering Optimization journal by Zhang et al. (2023) on job priority heuristics *("A New Job Priority Rule for the NEH-Based Heuristic"),* a scaling factor is introduced after matrix multiplication to prevent excessive variance when computing task priorities and the aspect is similar to Priority Equation, where scaling ensures that latency calculations remain within a feasible range. The scaling_factor in the Priority Equation serves as a dynamic multiplier that adjusts the system's processing efficiency in response to varying operational conditions and as a baseline for normalization rather than a direct indicator of system efficiency. This empirical parameter quantifies the system's ability to scale resources, such as processors, to accommodate fluctuating task loads. The scaling_factor plays a critical role in system optimization by adapting the processing capacity to current conditions – such as task arrival rates, resource availability, or processor utilization. In highly optimized systems equipped with advanced hardware, the scaling factor remains at 1.0, ensuring stable prioritization and minimal queuing overhead. In contrast, resource-constrained systems or those operating under significant load experience increased queuing delays and higher utilization (ρ), necessitating dynamic offloading and resource-aware task distribution to maintain efficiency.

Table 6: Scaling Factor and Task Prioritization in IoT Systems

Scaling_factor as resource Representation	Scaling_factor as prioritization of all tasks	Suitable for systems
1.0 – single resource performance (e.g., one type of processor or uniform task distribution)	1.0 – equal prioritization of all tasks	1.0 - IoT sensors (uniform workloads), predictable workloads, all tasks treated equally.
2.0 – double the capacity or efficiency, representing higher-performing resources or optimized local processing.	2.0 – higher emphasis on critical tasks, reflecting greater resource allocation.	2.0 - Edge-Cloud Systems (moderate variability). Edge processing is faster, but Cloud offloading introduces variability.
3.0 – triple the capacity or efficiency for sensors with significantly better resources (specialized Edge devices).	3.0 – strong prioritization; modeling systems where critical tasks dominate.	3.0 - Critical real-time systems, focused on prioritizing critical tasks, account for variability and real-time requirements.

The utilization factor ρ(rho) is a key parameter in Queuing Theory (Ibrahim et al., 2022) that measures how heavily the system's processing resources are utilized. It represents the proportion of the system's capacity that is currently in use, providing insight into the system's load and efficiency. The ρ is defined by $\frac{\boldsymbol{\lambda}}{\boldsymbol{c}\mu}$ λ-task arrival rate (tasks per second), c-number of processors available, μ service rate of each processor (tasks per second). The utilization factor can range from 0 (no load on the system) to 1 (full utilization of processing capacity), and values that exceed $\rho>1$ indicate an overloaded system (where the task arrival rate exceeds the system's total processing capacity). In cases where ρ increases, queuing delays grow because tasks spend more time waiting for resources, and in cases where $\rho=1$

or near 1, delays can increase exponentially as tasks compete for the limited remaining capacity. In order to keep ρ at a manageable level, resource allocation is needed by adding more processors to increase total processing capacity c, optimizing the service rate μ of each processor (e.g., faster hardware), or by reducing task arrival rate λ by offloading non-critical tasks.

Table 7: Utilization Factor (ρ) Interpretation and Applications

ρ value	Interpretation	Applications
ρ=1.0	System is fully utilized but stable	High-priority real-time systems
ρ=0.7 to 0.8	Efficient operation with buffer space	General IoT/IIoT systems with variability
ρ<0.5	Underutilized system	System with low task rate

The local service rate (μ) in the Priority Equation represents the speed at which an individual processor handles tasks, measured in tasks per second. Higher μ decreases the utilization factor $\boldsymbol{\rho = \frac{\lambda}{c\mu}}$, reducing queuing delays and enhancing system stability. Stability is achieved when $\mu \cdot c \geq \lambda$, ensuring $\rho \leq 1$. Conversely, if $\mu \cdot c < \lambda$, the system becomes unstable, with increasing queues and delays. Furthermore, higher μ reduces latency by shortening each task's time in service. Combined with the number of processors (c), μ determines the system's ability to scale with growing task arrival rates (λ) and supports efficient handling of increasing workloads.

The value of μ is influenced by hardware capabilities such as processor speed, memory bandwidth, and task complexity. Simpler, repetitive tasks yield higher μ, while complex operations, such as Artificial Intelligence-based processing or image analysis, lower it. The system environment also plays a role, as Edge devices typically exhibit lower μ due to resource limitations, whereas Cloud servers benefit from higher μ because of their advanced hardware.

In the Priority Equation, μ is essential for accurate modeling of system latency, reflecting real-world processing behavior. By balancing local and Cloud processing, higher μ enables more tasks to be handled locally, reducing dependency on Cloud offloading and minimizing transmission delays. Additionally, μ is integral to Queuing Theory (Willig, 1999) models like $M/G/c$ (Ibrahim et al., 2022), where it determines utilization (ρ), waiting times, and system capacity. In M/G/c models, μ influences queuing delays through the second moment of service time, $\boldsymbol{E}[\boldsymbol{s}^2]$, making it crucial for performance optimization. Overall, μ encapsulates the system's processing efficiency and is vital for optimizing performance, stability, and scalability.

3.6 Data Sources and Acquisition Methods

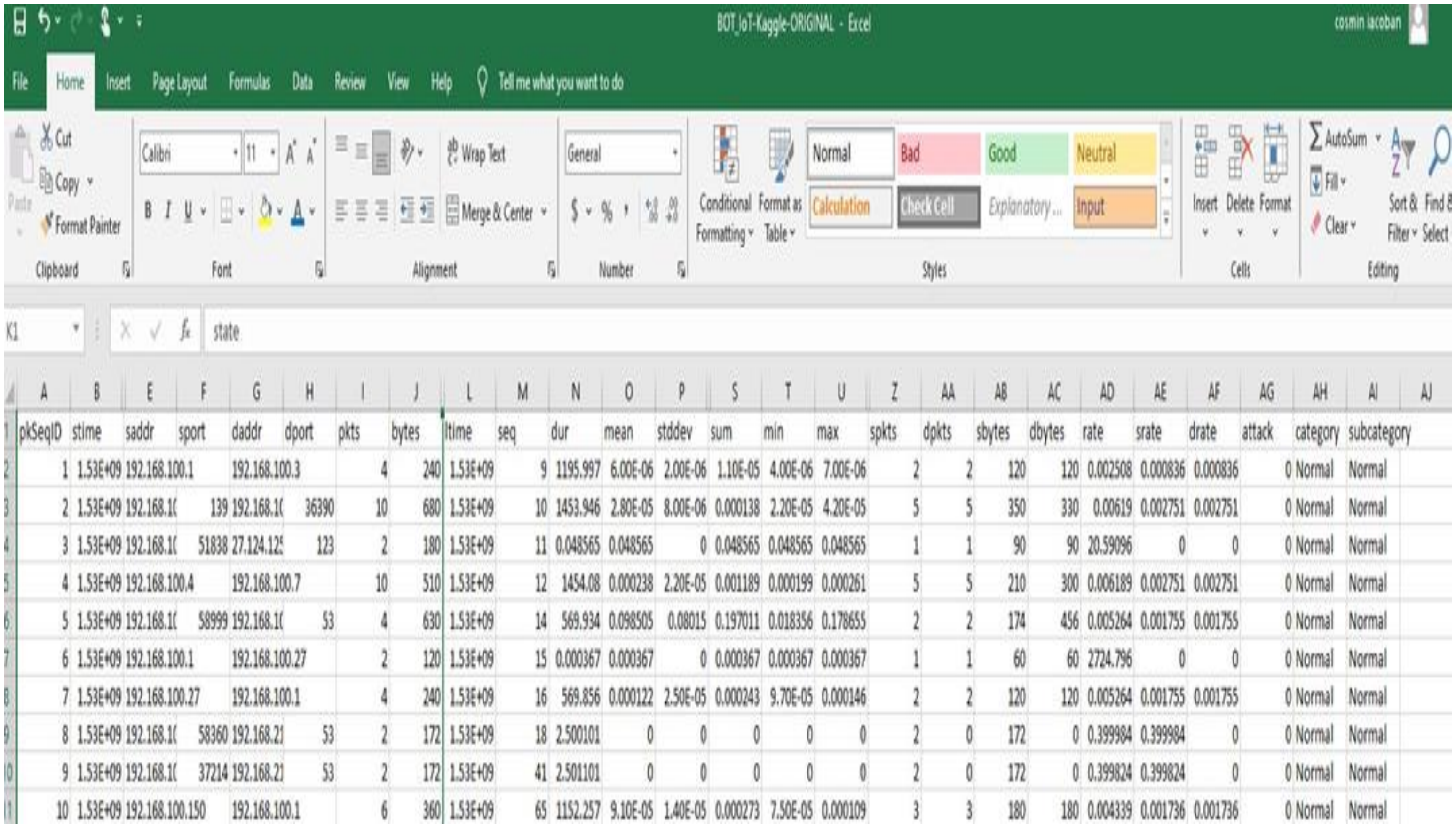
The datasets utilized in this research was downloaded from Kaggle.com. The Bot-IoT dataset is publicly available for academic research under a perpetual license for scholarly use by Koroniotis (2020), Koroniotis et al. (2020a, 2020b, 2019, 2018), Koroniotis and Moustafa (2020). However, any commercial application necessitates prior authorization from the dataset's creators. The authors have retained their rights under copyright law, and any utilization of the Bot-IoT dataset must include appropriate citations of the relevant publications that document its development and characteristics.

The dataset used in the research is extensive, necessitating a strategic selection of parameters most relevant to developing and implementing the Priority Equation. Given the focus on real-time supply chain operations, it is essential to identify key factors that directly impact task prioritization, resource allocation, and the decision-making process for local versus cloud processing. The selection of parameters was guided by their significance in determining system efficiency, latency, and adaptability to varying workload conditions.

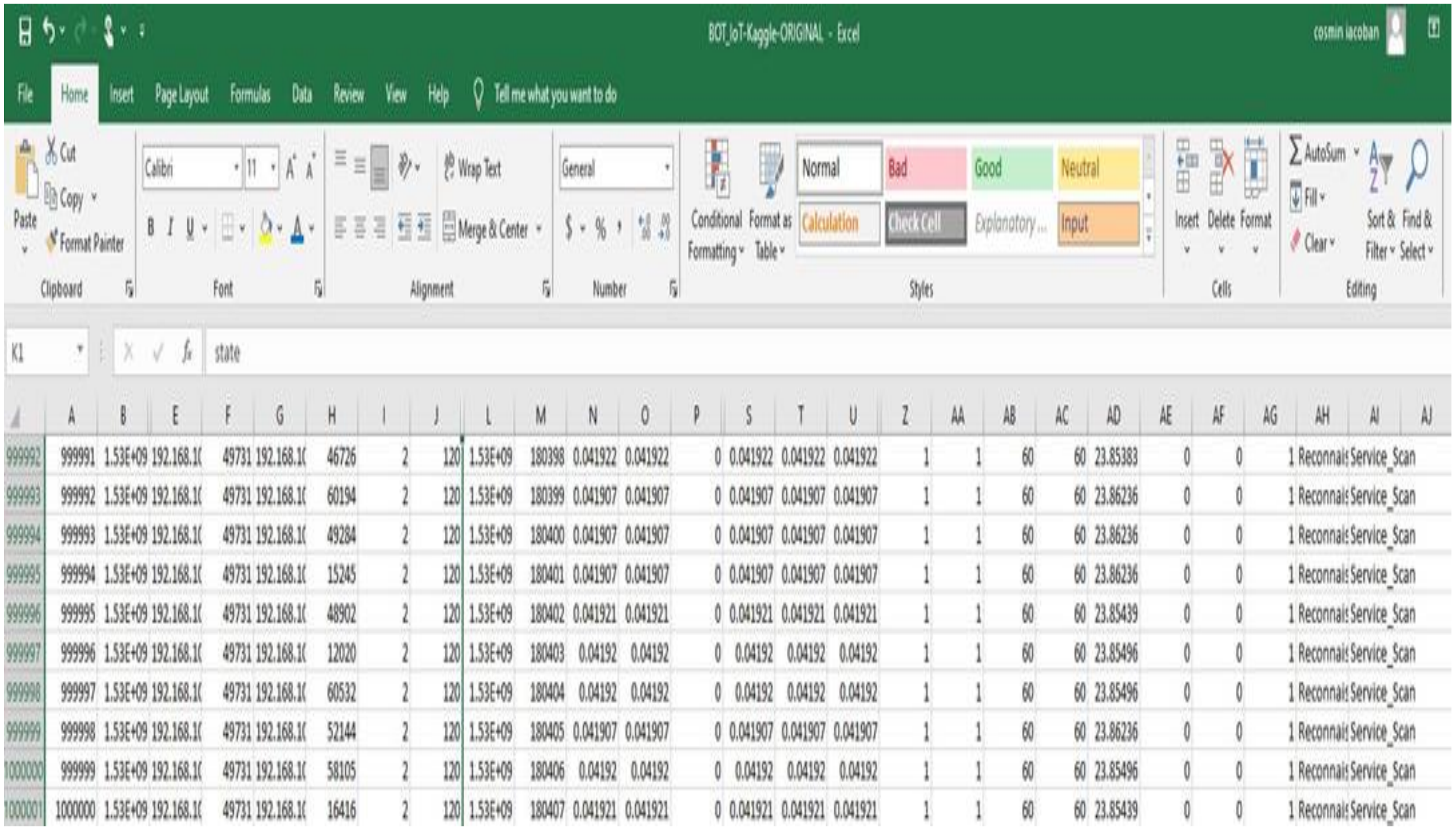
One of the most critical parameters chosen is the task arrival rate, denoted as lambda\λ. This value represents the rate at which tasks, such as temperature readings, RFID scans, and barcode processing, enter the system. Including this parameter is crucial for workload management, as it provides insight into how frequently the system must process incoming tasks. The service rate, denoted as mu\μ, is another fundamental component, as it defines the processing capability of local computational resources. By incorporating the utilization factor, rho\ρ, calculated as $\rho=\frac{\lambda}{c\mu}$, the formula ensures that system load is continuously monitored, preventing overload and dynamically adjusting the distribution of tasks between local processing and offloading to the cloud when necessary. The process also incorporates batch processing, represented as B, which determines how tasks are grouped for execution. This approach optimizes system performance by reducing the computational overhead associated with processing individual tasks separately. Furthermore, the cloud response time, W_R, ensures that when offloading occurs, the latency associated with remote processing is accounted for, thus maintaining the efficiency and responsiveness of the system.

These parameters were selected by the dataset's structure, which contains numerous attributes related to network traffic, packet transmission, and processing characteristics. Task arrival rate was derived from features such as ***stime*** (total tasks divided by average duration), which provide information on the number of tasks per second (λ). The batch size (B) was determined based on the ***bytes*** attribute (sum of bytes divided by total tasks) and local service (μ) derived from the duration (***dur***) column $\mu = \frac{1}{mean\ (dur)}$.

The choice of these parameters is particularly advantageous, given the large size of the dataset. The Priority Equation extracts only the most impactful attributes rather than processing all available columns, which would introduce unnecessary computational overhead. This selective approach enhances processing efficiency and ensures the system remains scalable as task volumes increase. Additionally, by dynamically adjusting task distribution based on real-time system conditions, the Priority Equation prevents bottlenecks and reduces latency. Through this structured selection of relevant parameters, the Priority Equation can balance computational workload, improve processing efficiency, and support the real-time decision-making requirements of Modern Supply Chain Systems.


BOT_IoT-Kaggle-ORIGINAL - Excel
cosmin iacoban
File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do
K1 state

	A	B	E	F	G	H	I	J	L	M	N	O	P	S	T	U	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI
1	pkSeqID	stime	saddr	sport	daddr	dport	pkts	bytes	ltime	seq	dur	mean	stddev	sum	min	max	spkts	dpkts	sbytes	dbytes	rate	srate	drate	attack	category	subcategory
2	1	1.53E+09	192.168.100.1		192.168.100.3		4	240	1.53E+09	9	1195.997	6.00E-06	2.00E-06	1.10E-05	4.00E-06	7.00E-06	2	2	120	120	0.002508	0.000836	0.000836	0	Normal	Normal
3	2	1.53E+09	192.168.1(	139	192.168.1(	36390	10	680	1.53E+09	10	1453.946	2.80E-05	8.00E-06	0.000138	2.20E-05	4.20E-05	5	5	350	330	0.00619	0.002751	0.002751	0	Normal	Normal
4	3	1.53E+09	192.168.1(	51838	27.124.12!	123	2	180	1.53E+09	11	0.048565	0.048565	0	0.048565	0.048565	0.048565	1	1	90	90	20.59096	0	0	0	Normal	Normal
5	4	1.53E+09	192.168.100.4		192.168.100.7		10	510	1.53E+09	12	1454.08	0.000238	2.20E-05	0.001189	0.000199	0.000261	5	5	210	300	0.006189	0.002751	0.002751	0	Normal	Normal
6	5	1.53E+09	192.168.1(	58999	192.168.1(	53	4	630	1.53E+09	14	569.934	0.098505	0.08015	0.197011	0.018356	0.178655	2	2	174	456	0.005264	0.001755	0.001755	0	Normal	Normal
7	6	1.53E+09	192.168.100.1		192.168.100.27		2	120	1.53E+09	15	0.000367	0.000367	0	0.000367	0.000367	0.000367	1	1	60	60	2724.796	0	0	0	Normal	Normal
8	7	1.53E+09	192.168.100.27		192.168.100.1		4	240	1.53E+09	16	569.856	0.000122	2.50E-05	0.000243	9.70E-05	0.000146	2	2	120	120	0.005264	0.001755	0.001755	0	Normal	Normal
9	8	1.53E+09	192.168.1(	58360	192.168.21	53	2	172	1.53E+09	18	2.500101	0	0	0	0	0	2	0	172	0	0.399984	0.399984	0	0	Normal	Normal
10	9	1.53E+09	192.168.1(	37214	192.168.21	53	2	172	1.53E+09	41	2.501101	0	0	0	0	0	2	0	172	0	0.399824	0.399824	0	0	Normal	Normal
11	10	1.53E+09	192.168.100.150		192.168.100.1		6	360	1.53E+09	65	1152.257	9.10E-05	1.40E-05	0.000273	7.50E-05	0.000109	3	3	180	180	0.004339	0.001736	0.001736	0	Normal	Normal



BOT_IoT-Kaggle-ORIGINAL - Excel

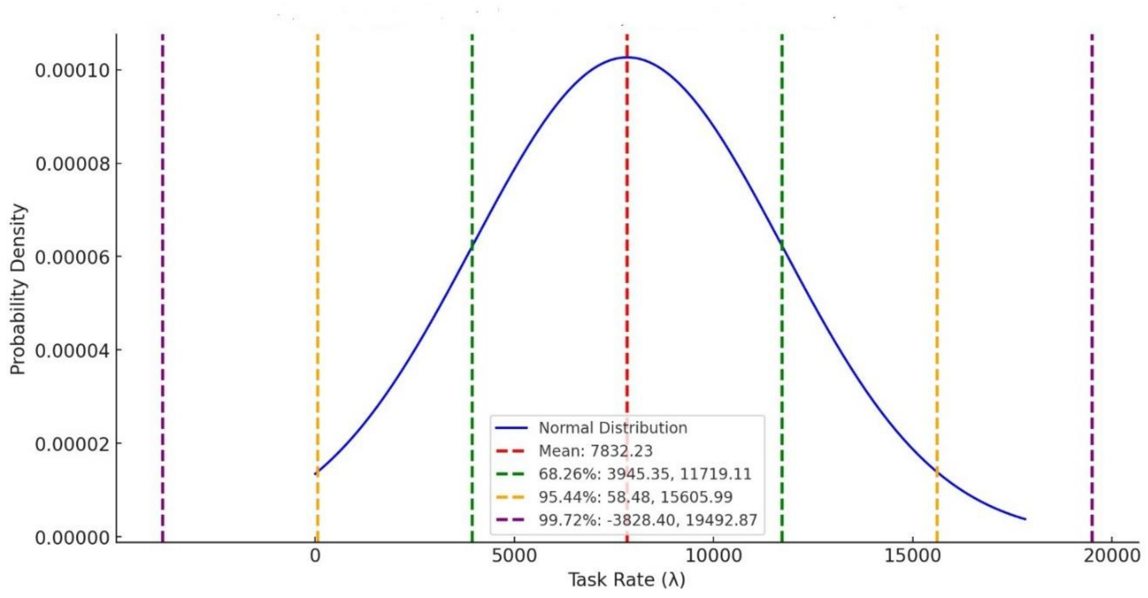
K1 state

	A	B	E	F	G	H	I	J	L	M	N	O	P	S	T	U	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ
999992	999991	1.53E+09	192.168.1(	49731	192.168.1(	46726	2	120	1.53E+09	180398	0.041922	0.041922	0	0.041922	0.041922	0.041922	1	1	60	60	23.85383	0	0	1	Reconnais	Service_Scan	
999993	999992	1.53E+09	192.168.1(	49731	192.168.1(	60194	2	120	1.53E+09	180399	0.041907	0.041907	0	0.041907	0.041907	0.041907	1	1	60	60	23.86236	0	0	1	Reconnais	Service_Scan	
999994	999993	1.53E+09	192.168.1(	49731	192.168.1(	49284	2	120	1.53E+09	180400	0.041907	0.041907	0	0.041907	0.041907	0.041907	1	1	60	60	23.86236	0	0	1	Reconnais	Service_Scan	
999995	999994	1.53E+09	192.168.1(	49731	192.168.1(	15245	2	120	1.53E+09	180401	0.041907	0.041907	0	0.041907	0.041907	0.041907	1	1	60	60	23.86236	0	0	1	Reconnais	Service_Scan	
999996	999995	1.53E+09	192.168.1(	49731	192.168.1(	48902	2	120	1.53E+09	180402	0.041921	0.041921	0	0.041921	0.041921	0.041921	1	1	60	60	23.85439	0	0	1	Reconnais	Service_Scan	
999997	999996	1.53E+09	192.168.1(	49731	192.168.1(	12020	2	120	1.53E+09	180403	0.04192	0.04192	0	0.04192	0.04192	0.04192	1	1	60	60	23.85496	0	0	1	Reconnais	Service_Scan	
999998	999997	1.53E+09	192.168.1(	49731	192.168.1(	60532	2	120	1.53E+09	180404	0.04192	0.04192	0	0.04192	0.04192	0.04192	1	1	60	60	23.85496	0	0	1	Reconnais	Service_Scan	
999999	999998	1.53E+09	192.168.1(	49731	192.168.1(	52144	2	120	1.53E+09	180405	0.041907	0.041907	0	0.041907	0.041907	0.041907	1	1	60	60	23.86236	0	0	1	Reconnais	Service_Scan	
1000000	999999	1.53E+09	192.168.1(	49731	192.168.1(	58105	2	120	1.53E+09	180406	0.04192	0.04192	0	0.04192	0.04192	0.04192	1	1	60	60	23.85496	0	0	1	Reconnais	Service_Scan	
1000001	1000000	1.53E+09	192.168.1(	49731	192.168.1(	16416	2	120	1.53E+09	180407	0.041921	0.041921	0	0.041921	0.041921	0.041921	1	1	60	60	23.85439	0	0	1	Reconnais	Service_Scan	

3.7 Data Analysis (Observations and Performance for Priority Equation)

In Queuing Theory (Willig, 1999) and performance analysis, calculating the task arrival rate λ and the utilization factor ρ is crucial for evaluating system efficiency. The following analysis demonstrates how these metrics are derived and interpreted using real-world data collected from the above datasets. The context involves assessing task processing efficiency in a computational system with one processor.

Figure 8: Task Rates with Standard Deviation Intervals



Calculating Mean Task Rate (λ) by quantifying the number of tasks arriving in the system (S_{time}):

$$Mean\ Task\ Rate\ (\lambda) = 7{,}832\ tasks/\sec = 7.832\ tasks/ms$$

The calculated task rate indicates that, on average, 7.832 tasks arrive every millisecond, which serves as the basis for subsequent performance metrics.

Calculating Service Time and Service Rate μ represents the system's ability to process tasks per millisecond. It is derived from the average task duration (*dur*) provided in the 74 million dataset.

Mean Task Duration (*dur*): 23.89 milliseconds or 24milliseconds =μ

The local service rate μ indicates that the probability for a single processor can handle approximately 24 simple tasks/inputs per millisecond, highlighting the system's processing capacity.

Calculating Utilization Factor ρ measures the fraction of a processor's capacity being used, and it is calculated by using the task arrival rate λ, service rate μ, and the number of processors (c).

- Task Arrival Rate (λ-lambda): 7.832 tasks/ms
- Local Service Rate (μ-mu): 24 tasks/ms
- Number of Processors (c): 1

$$\rho = \frac{\lambda}{c\mu} = \frac{7.832}{1 * 24} = 32.63\%$$

The utilization factor ρ of 32.63% indicates that the system is underutilized, and the calculations of λ, μ, and ρ provide a foundational analysis for evaluating system performance. The system's underutilized state highlights optimization opportunities, such as load balancing or task redistribution, to enhance efficiency in real-world scenarios. This methodology is critical for designing scalable and responsive computational systems in Supply Chain applications and beyond. Despite the underutilized state of the processor, tasks are offloaded to the Cloud when they exceed the predefined $\underline{T_{real\text{-}time}}$ or $\underline{s}$ threshold, ensuring that critical latency requirements are met. This approach balances local and Cloud processing to optimize performance while maintaining compliance with stringent real-time criteria.

Local latency (L_{local}) is a critical metric in Queuing Theory, measuring the time taken to process a task locally, including the queuing and service times. This calculation is based on the Priority Equation, which incorporates key system parameters such as task arrival rate (λ), service rate (μ), utilization factor (ρ), and the second moment of the service time $E[s^2]$.

The Formula for local latency is expressed as:

$$L_{local} = \left(\frac{\lambda * E[s^2]}{2(c * scaling\_factor\ )(1-\rho)} + \frac{1}{\mu}\right)$$

Where:

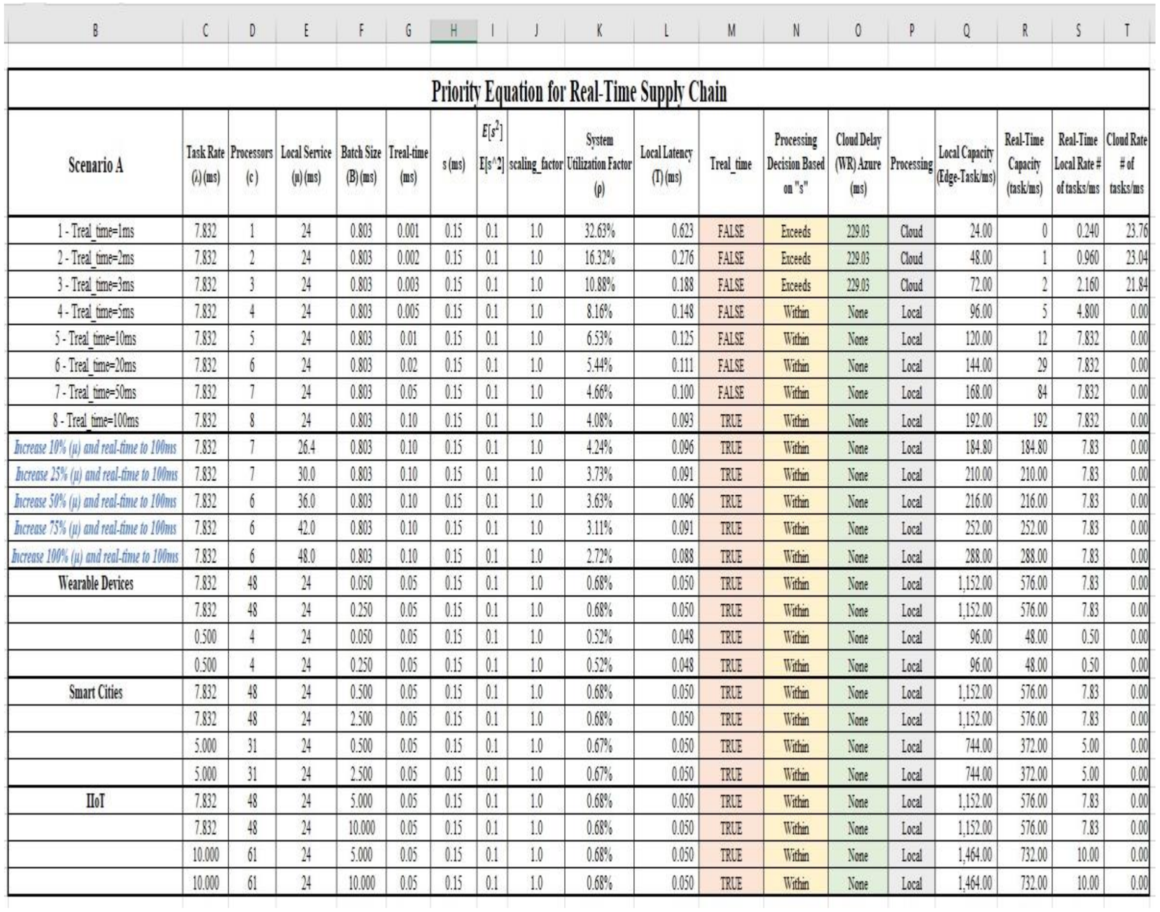
- λ: task arrival rate (tasks per second) – 7.832 tasks/ms
- $E[s^2]$: second moment of service time – 0.1 ms^2
- c: number of processors – 1 (single processor)
- scaling_factor: adjustment factor for system dynamics – 1.0 (default adjustment factor)
- ρ: utilization factor – 32.63%
- μ: local service rate (tasks per second) – 24 tasks/ms

The different scenario analysis of real-time task processing in a supply chain system involves evaluating the impact of various parameters, such as task arrival rates, processor counts, service rates, and real-time constraints, on system performance. Examining all five different scenarios (A, B, C, D, and E), each highlights how these parameters' variations influence the decision to process tasks locally or offload them to the Cloud. The primary objective is to determine the optimal configuration that minimizes Cloud dependency while ensuring low latency and efficient task execution.

Scenario A: Baseline Evaluation of $T_{real\text{-}time}$ Thresholds and Processor Scaling

Scenario A focuses on assessing the impact of increasing the real-time threshold (Treal-time) while scaling the number of processors (c) from 1 to 8. The task arrival rate (λ = 7.832 tasks/ms) and local service rate (μ = 24 ms) remain constant. At the lowest $T_{real\text{-}time}$ (1 ms), the local system is unable to process incoming tasks efficiently, leading to Cloud offloading with a significant delay of 229.03 ms. As $T_{real\text{-}time}$ increases, the system gains more time to process tasks locally, gradually reducing Cloud dependency. When $T_{real\text{-}time}$ reaches 5 ms, local processing becomes fully sufficient, eliminating Cloud offloading. Increasing $T_{real\text{-}time}$ to 100 ms significantly decreases system utilization (ρ=4.08%), indicating that the system can handle tasks efficiently with minimal congestion at this threshold. Beyond this point, adding more processors yields diminishing performance gains, as utilization remains low and local processing capacity is sufficient for the given workload.

The second part of Scenario A explores the effect of increasing the local service rate (μ) by 10% to 100% while keeping $T_{real\text{-}time}$ at 100 ms. The results show a steady decline in system utilization, further reinforcing that enhancing processing speed improves efficiency and reduces resource strain. However, beyond a 50% increase in μ, the additional gains become marginal, suggesting diminishing returns on performance improvements.

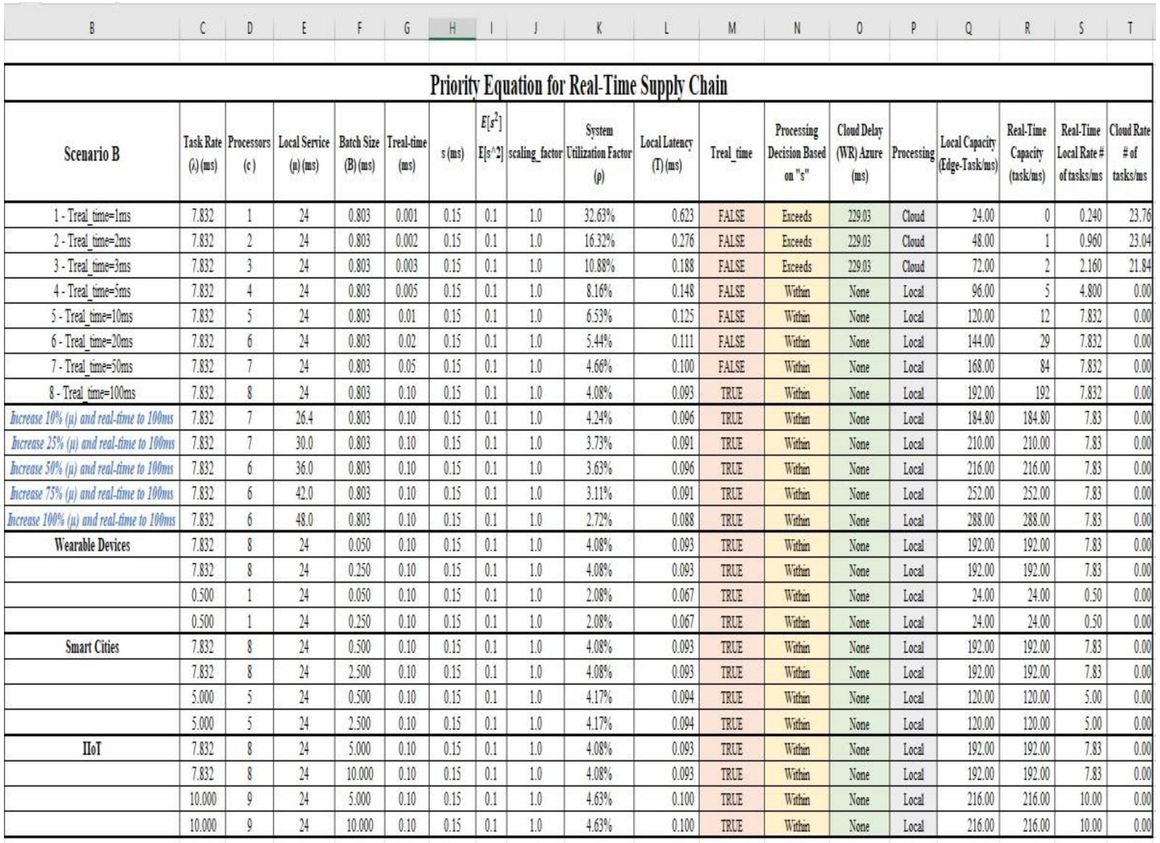


Priority Equation for Real-Time Supply Chain

B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Scenario A	Task Rate (λ) (ms)	Processors (c)	Local Service (μ) (ms)	Batch Size (B) (ms)	Treal-time (ms)	s (ms)	$E[s^2]$ E[s^2]	scaling_factor	System Utilization Factor (ρ)	Local Latency (T) (ms)	Treal_time	Processing Decision Based on "s"	Cloud Delay (WR) Azure (ms)	Processing	Local Capacity (Edge-Task/ms)	Real-Time Capacity (task/ms)	Real-Time Local Rate # of tasks/ms	Cloud Rate # of tasks/ms
1 - Treal_time=1ms	7.832	1	24	0.803	0.001	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.03	Cloud	24.00	0	0.240	23.76
2 - Treal_time=2ms	7.832	2	24	0.803	0.002	0.15	0.1	1.0	16.32%	0.276	FALSE	Exceeds	229.03	Cloud	48.00	1	0.960	23.04
3 - Treal_time=3ms	7.832	3	24	0.803	0.003	0.15	0.1	1.0	10.88%	0.188	FALSE	Exceeds	229.03	Cloud	72.00	2	2.160	21.84
4 - Treal_time=5ms	7.832	4	24	0.803	0.005	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	5	4.800	0.00
5 - Treal_time=10ms	7.832	5	24	0.803	0.01	0.15	0.1	1.0	6.53%	0.125	FALSE	Within	None	Local	120.00	12	7.832	0.00
6 - Treal_time=20ms	7.832	6	24	0.803	0.02	0.15	0.1	1.0	5.44%	0.111	FALSE	Within	None	Local	144.00	29	7.832	0.00
7 - Treal_time=50ms	7.832	7	24	0.803	0.05	0.15	0.1	1.0	4.66%	0.100	FALSE	Within	None	Local	168.00	84	7.832	0.00
8 - Treal_time=100ms	7.832	8	24	0.803	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192	7.832	0.00
Increase 10% (μ) and real-time to 100ms	7.832	7	26.4	0.803	0.10	0.15	0.1	1.0	4.24%	0.096	TRUE	Within	None	Local	184.80	184.80	7.83	0.00
Increase 25% (μ) and real-time to 100ms	7.832	7	30.0	0.803	0.10	0.15	0.1	1.0	3.73%	0.091	TRUE	Within	None	Local	210.00	210.00	7.83	0.00
Increase 50% (μ) and real-time to 100ms	7.832	6	36.0	0.803	0.10	0.15	0.1	1.0	3.63%	0.096	TRUE	Within	None	Local	216.00	216.00	7.83	0.00
Increase 75% (μ) and real-time to 100ms	7.832	6	42.0	0.803	0.10	0.15	0.1	1.0	3.11%	0.091	TRUE	Within	None	Local	252.00	252.00	7.83	0.00
Increase 100% (μ) and real-time to 100ms	7.832	6	48.0	0.803	0.10	0.15	0.1	1.0	2.72%	0.088	TRUE	Within	None	Local	288.00	288.00	7.83	0.00
Wearable Devices	7.832	48	24	0.050	0.05	0.15	0.1	1.0	0.68%	0.050	TRUE	Within	None	Local	1,152.00	576.00	7.83	0.00
	7.832	48	24	0.250	0.05	0.15	0.1	1.0	0.68%	0.050	TRUE	Within	None	Local	1,152.00	576.00	7.83	0.00
	0.500	4	24	0.050	0.05	0.15	0.1	1.0	0.52%	0.048	TRUE	Within	None	Local	96.00	48.00	0.50	0.00
	0.500	4	24	0.250	0.05	0.15	0.1	1.0	0.52%	0.048	TRUE	Within	None	Local	96.00	48.00	0.50	0.00
Smart Cities	7.832	48	24	0.500	0.05	0.15	0.1	1.0	0.68%	0.050	TRUE	Within	None	Local	1,152.00	576.00	7.83	0.00
	7.832	48	24	2.500	0.05	0.15	0.1	1.0	0.68%	0.050	TRUE	Within	None	Local	1,152.00	576.00	7.83	0.00
	5.000	31	24	0.500	0.05	0.15	0.1	1.0	0.67%	0.050	TRUE	Within	None	Local	744.00	372.00	5.00	0.00
	5.000	31	24	2.500	0.05	0.15	0.1	1.0	0.67%	0.050	TRUE	Within	None	Local	744.00	372.00	5.00	0.00
IIoT	7.832	48	24	5.000	0.05	0.15	0.1	1.0	0.68%	0.050	TRUE	Within	None	Local	1,152.00	576.00	7.83	0.00
	7.832	48	24	10.000	0.05	0.15	0.1	1.0	0.68%	0.050	TRUE	Within	None	Local	1,152.00	576.00	7.83	0.00
	10.000	61	24	5.000	0.05	0.15	0.1	1.0	0.68%	0.050	TRUE	Within	None	Local	1,464.00	732.00	10.00	0.00
	10.000	61	24	10.000	0.05	0.15	0.1	1.0	0.68%	0.050	TRUE	Within	None	Local	1,464.00	732.00	10.00	0.00

Scenario B: Alternative Configuration with Similar Scaling

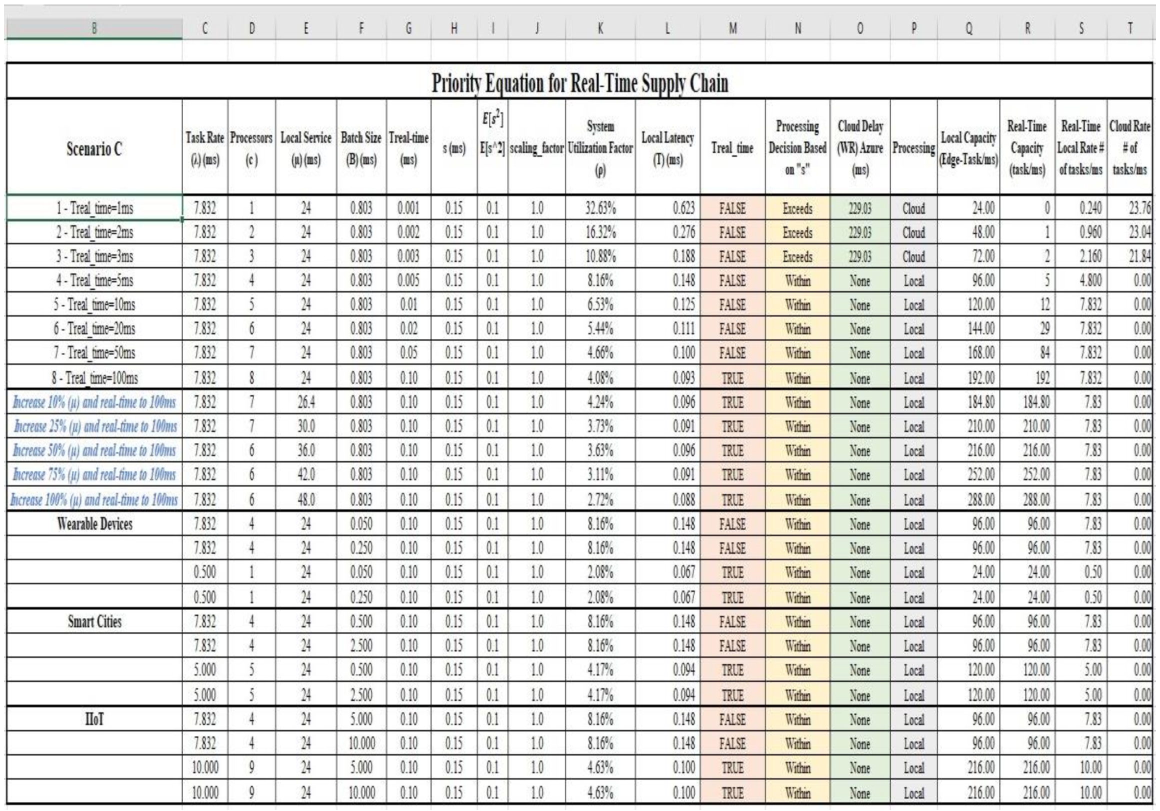
Scenario B replicates the conditions of Scenario A while evaluating alternative processor distributions. The results align closely with Scenario A, confirming that increasing Treal-time and c improves local processing capability while reducing Cloud dependency. The critical transition occurs at Treal-time = 5ms, where local processing becomes feasible, allowing a utilization factor below 10%. Similar to Scenario A, a 100% increase in μ yields a utilization factor as low as 2.72%, demonstrating efficient real-time task processing at optimal conditions.



B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Priority Equation for Real-Time Supply Chain																		
Scenario B	**Task Rate (λ) (ms)**	**Processors (c)**	**Local Service (μ) (ms)**	**Batch Size (B) (ms)**	**Treal-time (ms)**	**s (ms)**	**$E[s^2]$ E[s^2]**	**scaling_factor**	**System Utilization Factor (ρ)**	**Local Latency (T) (ms)**	**Treal_time**	**Processing Decision Based on "s"**	**Cloud Delay (WR) Azure (ms)**	**Processing**	**Local Capacity (Edge-Task/ms)**	**Real-Time Capacity (task/ms)**	**Real-Time Local Rate # of tasks/ms**	**Cloud Rate # of tasks/ms**
1 - Treal_time=1ms	7.832	1	24	0.803	0.001	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.03	Cloud	24.00	0	0.240	23.76
2 - Treal_time=2ms	7.832	2	24	0.803	0.002	0.15	0.1	1.0	16.32%	0.276	FALSE	Exceeds	229.03	Cloud	48.00	1	0.960	23.04
3 - Treal_time=3ms	7.832	3	24	0.803	0.003	0.15	0.1	1.0	10.88%	0.188	FALSE	Exceeds	229.03	Cloud	72.00	2	2.160	21.84
4 - Treal_time=5ms	7.832	4	24	0.803	0.005	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	5	4.800	0.00
5 - Treal_time=10ms	7.832	5	24	0.803	0.01	0.15	0.1	1.0	6.53%	0.125	FALSE	Within	None	Local	120.00	12	7.832	0.00
6 - Treal_time=20ms	7.832	6	24	0.803	0.02	0.15	0.1	1.0	5.44%	0.111	FALSE	Within	None	Local	144.00	29	7.832	0.00
7 - Treal_time=50ms	7.832	7	24	0.803	0.05	0.15	0.1	1.0	4.66%	0.100	FALSE	Within	None	Local	168.00	84	7.832	0.00
8 - Treal_time=100ms	7.832	8	24	0.803	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192	7.832	0.00
Increase 10% (μ) and real-time to 100ms	7.832	7	26.4	0.803	0.10	0.15	0.1	1.0	4.24%	0.096	TRUE	Within	None	Local	184.80	184.80	7.83	0.00
Increase 25% (μ) and real-time to 100ms	7.832	7	30.0	0.803	0.10	0.15	0.1	1.0	3.73%	0.091	TRUE	Within	None	Local	210.00	210.00	7.83	0.00
Increase 50% (μ) and real-time to 100ms	7.832	6	36.0	0.803	0.10	0.15	0.1	1.0	3.63%	0.096	TRUE	Within	None	Local	216.00	216.00	7.83	0.00
Increase 75% (μ) and real-time to 100ms	7.832	6	42.0	0.803	0.10	0.15	0.1	1.0	3.11%	0.091	TRUE	Within	None	Local	252.00	252.00	7.83	0.00
Increase 100% (μ) and real-time to 100ms	7.832	6	48.0	0.803	0.10	0.15	0.1	1.0	2.72%	0.088	TRUE	Within	None	Local	288.00	288.00	7.83	0.00
Wearable Devices	7.832	8	24	0.050	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192.00	7.83	0.00
	7.832	8	24	0.250	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192.00	7.83	0.00
	0.500	1	24	0.050	0.10	0.15	0.1	1.0	2.08%	0.067	TRUE	Within	None	Local	24.00	24.00	0.50	0.00
	0.500	1	24	0.250	0.10	0.15	0.1	1.0	2.08%	0.067	TRUE	Within	None	Local	24.00	24.00	0.50	0.00
Smart Cities	7.832	8	24	0.500	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192.00	7.83	0.00
	7.832	8	24	2.500	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192.00	7.83	0.00
	5.000	5	24	0.500	0.10	0.15	0.1	1.0	4.17%	0.094	TRUE	Within	None	Local	120.00	120.00	5.00	0.00
	5.000	5	24	2.500	0.10	0.15	0.1	1.0	4.17%	0.094	TRUE	Within	None	Local	120.00	120.00	5.00	0.00
IIoT	7.832	8	24	5.000	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192.00	7.83	0.00
	7.832	8	24	10.000	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192.00	7.83	0.00
	10.000	9	24	5.000	0.10	0.15	0.1	1.0	4.63%	0.100	TRUE	Within	None	Local	216.00	216.00	10.00	0.00
	10.000	9	24	10.000	0.10	0.15	0.1	1.0	4.63%	0.100	TRUE	Within	None	Local	216.00	216.00	10.00	0.00

Scenario C: Fine-Tuning Local Processing Parameters

Scenario C further refines the evaluation of real-time task processing by testing additional increments in Treal-time. The results validate Scenarios A and B observations, reinforcing that Cloud offloading becomes unnecessary once Treal-time exceeds 5 ms. The study also highlights that increasing c beyond a certain threshold (approximately eight processors) diminishes performance, as system utilization remains low regardless of additional processing power. This Scenario solidifies the claim that balancing λ, μ, and c is crucial for achieving optimal efficiency without over-provisioning computational resources.

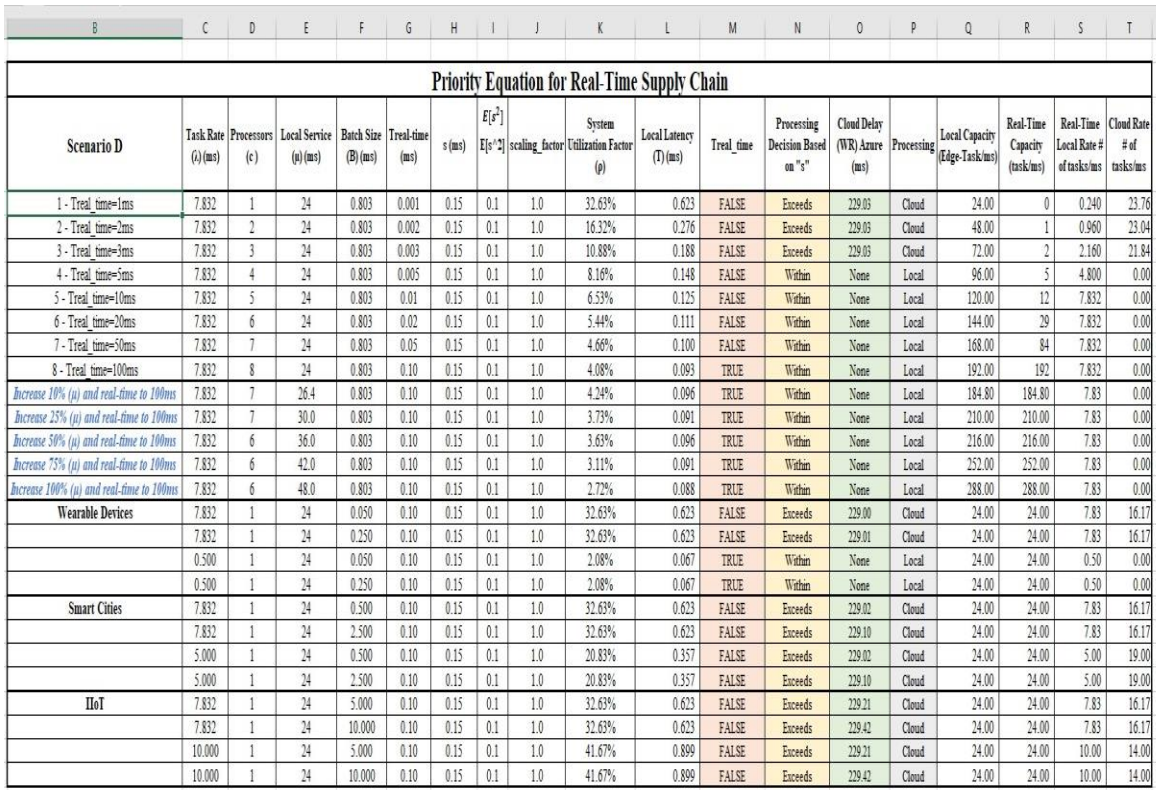


Priority Equation for Real-Time Supply Chain

B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Scenario C	Task Rate (λ) (ms)	Processors (c)	Local Service (μ) (ms)	Batch Size (B) (ms)	Treal-time (ms)	s (ms)	$E[s^2]$ E[s^2]	scaling_factor	System Utilization Factor (ρ)	Local Latency (T) (ms)	Treal_time	Processing Decision Based on "s"	Cloud Delay (WR) Azure (ms)	Processing	Local Capacity (Edge-Task/ms)	Real-Time Capacity (task/ms)	Real-Time Local Rate # of tasks/ms	Cloud Rate # of tasks/ms
1 - Treal_time=1ms	7.832	1	24	0.803	0.001	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.03	Cloud	24.00	0	0.240	23.76
2 - Treal_time=2ms	7.832	2	24	0.803	0.002	0.15	0.1	1.0	16.32%	0.276	FALSE	Exceeds	229.03	Cloud	48.00	1	0.960	23.04
3 - Treal_time=3ms	7.832	3	24	0.803	0.003	0.15	0.1	1.0	10.88%	0.188	FALSE	Exceeds	229.03	Cloud	72.00	2	2.160	21.84
4 - Treal_time=5ms	7.832	4	24	0.803	0.005	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	5	4.800	0.00
5 - Treal_time=10ms	7.832	5	24	0.803	0.01	0.15	0.1	1.0	6.53%	0.125	FALSE	Within	None	Local	120.00	12	7.832	0.00
6 - Treal_time=20ms	7.832	6	24	0.803	0.02	0.15	0.1	1.0	5.44%	0.111	FALSE	Within	None	Local	144.00	29	7.832	0.00
7 - Treal_time=50ms	7.832	7	24	0.803	0.05	0.15	0.1	1.0	4.66%	0.100	FALSE	Within	None	Local	168.00	84	7.832	0.00
8 - Treal_time=100ms	7.832	8	24	0.803	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192	7.832	0.00
Increase 10% (μ) and real-time to 100ms	7.832	7	26.4	0.803	0.10	0.15	0.1	1.0	4.24%	0.096	TRUE	Within	None	Local	184.80	184.80	7.83	0.00
Increase 25% (μ) and real-time to 100ms	7.832	7	30.0	0.803	0.10	0.15	0.1	1.0	3.73%	0.091	TRUE	Within	None	Local	210.00	210.00	7.83	0.00
Increase 50% (μ) and real-time to 100ms	7.832	6	36.0	0.803	0.10	0.15	0.1	1.0	3.63%	0.096	TRUE	Within	None	Local	216.00	216.00	7.83	0.00
Increase 75% (μ) and real-time to 100ms	7.832	6	42.0	0.803	0.10	0.15	0.1	1.0	3.11%	0.091	TRUE	Within	None	Local	252.00	252.00	7.83	0.00
Increase 100% (μ) and real-time to 100ms	7.832	6	48.0	0.803	0.10	0.15	0.1	1.0	2.72%	0.088	TRUE	Within	None	Local	288.00	288.00	7.83	0.00
Wearable Devices	7.832	4	24	0.050	0.10	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	96.00	7.83	0.00
	7.832	4	24	0.250	0.10	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	96.00	7.83	0.00
	0.500	1	24	0.050	0.10	0.15	0.1	1.0	2.08%	0.067	TRUE	Within	None	Local	24.00	24.00	0.50	0.00
	0.500	1	24	0.250	0.10	0.15	0.1	1.0	2.08%	0.067	TRUE	Within	None	Local	24.00	24.00	0.50	0.00
Smart Cities	7.832	4	24	0.500	0.10	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	96.00	7.83	0.00
	7.832	4	24	2.500	0.10	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	96.00	7.83	0.00
	5.000	5	24	0.500	0.10	0.15	0.1	1.0	4.17%	0.094	TRUE	Within	None	Local	120.00	120.00	5.00	0.00
	5.000	5	24	2.500	0.10	0.15	0.1	1.0	4.17%	0.094	TRUE	Within	None	Local	120.00	120.00	5.00	0.00
IIoT	7.832	4	24	5.000	0.10	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	96.00	7.83	0.00
	7.832	4	24	10.000	0.10	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	96.00	7.83	0.00
	10.000	9	24	5.000	0.10	0.15	0.1	1.0	4.63%	0.100	TRUE	Within	None	Local	216.00	216.00	10.00	0.00
	10.000	9	24	10.000	0.10	0.15	0.1	1.0	4.63%	0.100	TRUE	Within	None	Local	216.00	216.00	10.00	0.00

Scenario D: Introducing Cloud Dependence in Varied Workloads

Unlike previous scenarios, Scenario D introduces conditions where Cloud processing remains partially necessary even as $T_{\text{real-time}}$ increases. The results show that for certain configurations, even when c is increased, the local system remains unable to meet stringent real-time constraints, resulting in continued Cloud offloading. The utilization factor remains relatively high, suggesting that some real-world applications may require hybrid processing solutions. Additionally, due to the overwhelming demand on local processors, the Cloud remains a necessary component for Smart City and IIoT applications with higher task rates (e.g., 10,000 tasks/ms).

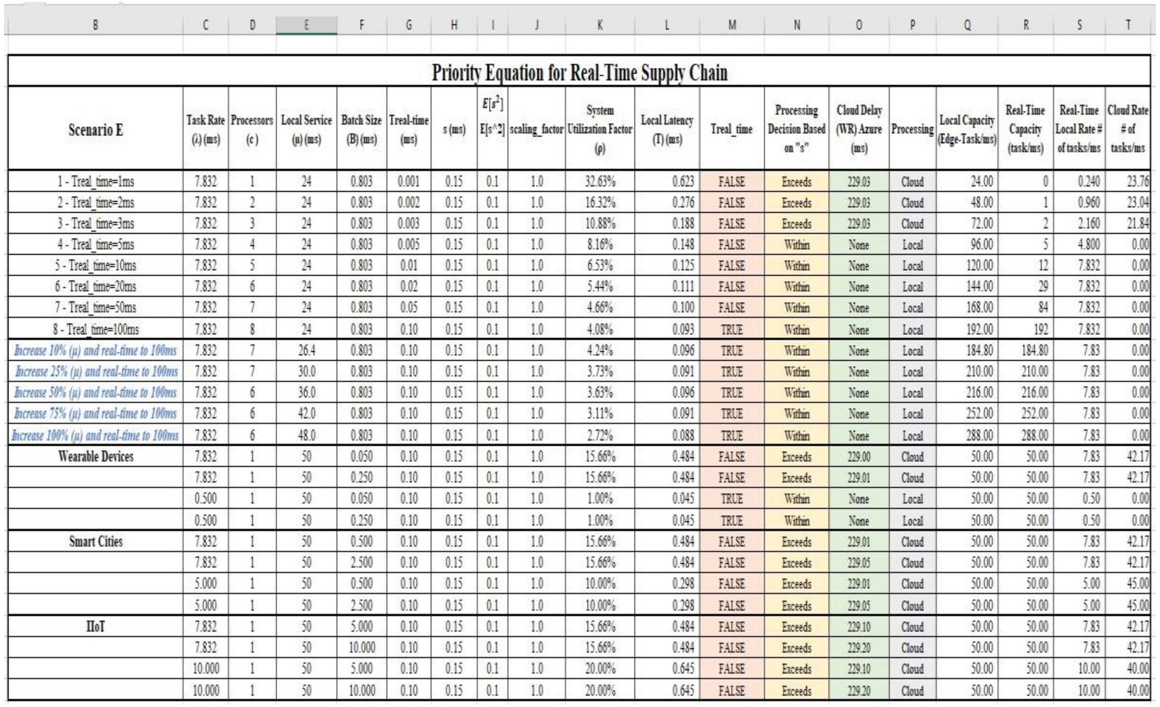


Priority Equation for Real-Time Supply Chain

Scenario D	Task Rate (λ) (ms)	Processors (c)	Local Service (μ) (ms)	Batch Size (B) (ms)	Treal-time (ms)	s (ms)	$E[s^2]$ E[s^2]	scaling_factor	System Utilization Factor (ρ)	Local Latency (T) (ms)	Treal_time	Processing Decision Based on "s"	Cloud Delay (WR) Azure (ms)	Processing	Local Capacity (Edge-Task/ms)	Real-Time Capacity (task/ms)	Real-Time Local Rate # of tasks/ms	Cloud Rate # of tasks/ms
1 - Treal_time=1ms	7.832	1	24	0.803	0.001	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.03	Cloud	24.00	0	0.240	23.76
2 - Treal_time=2ms	7.832	2	24	0.803	0.002	0.15	0.1	1.0	16.32%	0.276	FALSE	Exceeds	229.03	Cloud	48.00	1	0.960	23.04
3 - Treal_time=3ms	7.832	3	24	0.803	0.003	0.15	0.1	1.0	10.88%	0.188	FALSE	Exceeds	229.03	Cloud	72.00	2	2.160	21.84
4 - Treal_time=5ms	7.832	4	24	0.803	0.005	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	5	4.800	0.00
5 - Treal_time=10ms	7.832	5	24	0.803	0.01	0.15	0.1	1.0	6.53%	0.125	FALSE	Within	None	Local	120.00	12	7.832	0.00
6 - Treal_time=20ms	7.832	6	24	0.803	0.02	0.15	0.1	1.0	5.44%	0.111	FALSE	Within	None	Local	144.00	29	7.832	0.00
7 - Treal_time=50ms	7.832	7	24	0.803	0.05	0.15	0.1	1.0	4.66%	0.100	FALSE	Within	None	Local	168.00	84	7.832	0.00
8 - Treal_time=100ms	7.832	8	24	0.803	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192	7.832	0.00
Increase 10% (μ) and real-time to 100ms	7.832	7	26.4	0.803	0.10	0.15	0.1	1.0	4.24%	0.096	TRUE	Within	None	Local	184.80	184.80	7.83	0.00
Increase 25% (μ) and real-time to 100ms	7.832	7	30.0	0.803	0.10	0.15	0.1	1.0	3.73%	0.091	TRUE	Within	None	Local	210.00	210.00	7.83	0.00
Increase 50% (μ) and real-time to 100ms	7.832	6	36.0	0.803	0.10	0.15	0.1	1.0	3.63%	0.096	TRUE	Within	None	Local	216.00	216.00	7.83	0.00
Increase 75% (μ) and real-time to 100ms	7.832	6	42.0	0.803	0.10	0.15	0.1	1.0	3.11%	0.091	TRUE	Within	None	Local	252.00	252.00	7.83	0.00
Increase 100% (μ) and real-time to 100ms	7.832	6	48.0	0.803	0.10	0.15	0.1	1.0	2.72%	0.088	TRUE	Within	None	Local	288.00	288.00	7.83	0.00
Wearable Devices	7.832	1	24	0.050	0.10	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.00	Cloud	24.00	24.00	7.83	16.17
	7.832	1	24	0.250	0.10	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.01	Cloud	24.00	24.00	7.83	16.17
	0.500	1	24	0.050	0.10	0.15	0.1	1.0	2.08%	0.067	TRUE	Within	None	Local	24.00	24.00	0.50	0.00
	0.500	1	24	0.250	0.10	0.15	0.1	1.0	2.08%	0.067	TRUE	Within	None	Local	24.00	24.00	0.50	0.00
Smart Cities	7.832	1	24	0.500	0.10	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.02	Cloud	24.00	24.00	7.83	16.17
	7.832	1	24	2.500	0.10	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.10	Cloud	24.00	24.00	7.83	16.17
	5.000	1	24	0.500	0.10	0.15	0.1	1.0	20.83%	0.357	FALSE	Exceeds	229.02	Cloud	24.00	24.00	5.00	19.00
	5.000	1	24	2.500	0.10	0.15	0.1	1.0	20.83%	0.357	FALSE	Exceeds	229.10	Cloud	24.00	24.00	5.00	19.00
IIoT	7.832	1	24	5.000	0.10	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.21	Cloud	24.00	24.00	7.83	16.17
	7.832	1	24	10.000	0.10	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.42	Cloud	24.00	24.00	7.83	16.17
	10.000	1	24	5.000	0.10	0.15	0.1	1.0	41.67%	0.899	FALSE	Exceeds	229.21	Cloud	24.00	24.00	10.00	14.00
	10.000	1	24	10.000	0.10	0.15	0.1	1.0	41.67%	0.899	FALSE	Exceeds	229.42	Cloud	24.00	24.00	10.00	14.00

Scenario E: Adjusting Service Rate and Batch Size for Real-Time Processing

Scenario E introduces an increased local service rate (μ= 50 ms) while evaluating its effect on batch processing efficiency. The findings indicate that while a higher μ improves overall performance, it does not entirely eliminate Cloud dependency when task arrival rates exceed a critical threshold. The system remains self-sufficient mainly in Wearable Devices and Smart City applications, but offloading is still required in IIoT applications with extreme task loads. This scenario emphasizes that while increasing μ improves performance, hybrid processing strategies remain necessary for high-intensity workloads.

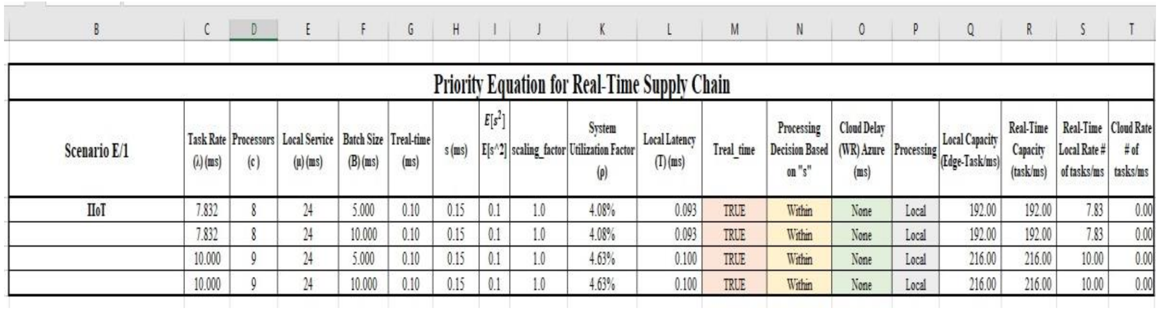


Priority Equation for Real-Time Supply Chain

B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Scenario E	**Task Rate (λ) (ms)**	**Processors (c)**	**Local Service (μ) (ms)**	**Batch Size (B) (ms)**	**Treal-time (ms)**	**s (ms)**	**$E[s^2]$ E[s^2]**	**scaling_factor**	**System Utilization Factor (ρ)**	**Local Latency (T) (ms)**	**Treal_time**	**Processing Decision Based on "s"**	**Cloud Delay (WR) Azure (ms)**	**Processing**	**Local Capacity (Edge-Task/ms)**	**Real-Time Capacity (task/ms)**	**Real-Time Local Rate # of tasks/ms**	**Cloud Rate # of tasks/ms**
1 - Treal_time=1ms	7.832	1	24	0.803	0.001	0.15	0.1	1.0	32.63%	0.623	FALSE	Exceeds	229.03	Cloud	24.00	0	0.240	23.76
2 - Treal_time=2ms	7.832	2	24	0.803	0.002	0.15	0.1	1.0	16.32%	0.276	FALSE	Exceeds	229.03	Cloud	48.00	1	0.960	23.04
3 - Treal_time=3ms	7.832	3	24	0.803	0.003	0.15	0.1	1.0	10.88%	0.188	FALSE	Exceeds	229.03	Cloud	72.00	2	2.160	21.84
4 - Treal_time=5ms	7.832	4	24	0.803	0.005	0.15	0.1	1.0	8.16%	0.148	FALSE	Within	None	Local	96.00	5	4.800	0.00
5 - Treal_time=10ms	7.832	5	24	0.803	0.01	0.15	0.1	1.0	6.53%	0.125	FALSE	Within	None	Local	120.00	12	7.832	0.00
6 - Treal_time=20ms	7.832	6	24	0.803	0.02	0.15	0.1	1.0	5.44%	0.111	FALSE	Within	None	Local	144.00	29	7.832	0.00
7 - Treal_time=50ms	7.832	7	24	0.803	0.05	0.15	0.1	1.0	4.66%	0.100	FALSE	Within	None	Local	168.00	84	7.832	0.00
8 - Treal_time=100ms	7.832	8	24	0.803	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192	7.832	0.00
Increase 10% (μ) and real-time to 100ms	7.832	7	26.4	0.803	0.10	0.15	0.1	1.0	4.24%	0.096	TRUE	Within	None	Local	184.80	184.80	7.83	0.00
Increase 25% (μ) and real-time to 100ms	7.832	7	30.0	0.803	0.10	0.15	0.1	1.0	3.73%	0.091	TRUE	Within	None	Local	210.00	210.00	7.83	0.00
Increase 50% (μ) and real-time to 100ms	7.832	6	36.0	0.803	0.10	0.15	0.1	1.0	3.63%	0.096	TRUE	Within	None	Local	216.00	216.00	7.83	0.00
Increase 75% (μ) and real-time to 100ms	7.832	6	42.0	0.803	0.10	0.15	0.1	1.0	3.11%	0.091	TRUE	Within	None	Local	252.00	252.00	7.83	0.00
Increase 100% (μ) and real-time to 100ms	7.832	6	48.0	0.803	0.10	0.15	0.1	1.0	2.72%	0.088	TRUE	Within	None	Local	288.00	288.00	7.83	0.00
Wearable Devices	7.832	1	50	0.050	0.10	0.15	0.1	1.0	15.66%	0.484	FALSE	Exceeds	229.00	Cloud	50.00	50.00	7.83	42.17
	7.832	1	50	0.250	0.10	0.15	0.1	1.0	15.66%	0.484	FALSE	Exceeds	229.01	Cloud	50.00	50.00	7.83	42.17
	0.500	1	50	0.050	0.10	0.15	0.1	1.0	1.00%	0.045	TRUE	Within	None	Local	50.00	50.00	0.50	0.00
	0.500	1	50	0.250	0.10	0.15	0.1	1.0	1.00%	0.045	TRUE	Within	None	Local	50.00	50.00	0.50	0.00
Smart Cities	7.832	1	50	0.500	0.10	0.15	0.1	1.0	15.66%	0.484	FALSE	Exceeds	229.01	Cloud	50.00	50.00	7.83	42.17
	7.832	1	50	2.500	0.10	0.15	0.1	1.0	15.66%	0.484	FALSE	Exceeds	229.05	Cloud	50.00	50.00	7.83	42.17
	5.000	1	50	0.500	0.10	0.15	0.1	1.0	10.00%	0.298	FALSE	Exceeds	229.01	Cloud	50.00	50.00	5.00	45.00
	5.000	1	50	2.500	0.10	0.15	0.1	1.0	10.00%	0.298	FALSE	Exceeds	229.05	Cloud	50.00	50.00	5.00	45.00
IIoT	7.832	1	50	5.000	0.10	0.15	0.1	1.0	15.66%	0.484	FALSE	Exceeds	229.10	Cloud	50.00	50.00	7.83	42.17
	7.832	1	50	10.000	0.10	0.15	0.1	1.0	15.66%	0.484	FALSE	Exceeds	229.20	Cloud	50.00	50.00	7.83	42.17
	10.000	1	50	5.000	0.10	0.15	0.1	1.0	20.00%	0.645	FALSE	Exceeds	229.10	Cloud	50.00	50.00	10.00	40.00
	10.000	1	50	10.000	0.10	0.15	0.1	1.0	20.00%	0.645	FALSE	Exceeds	229.20	Cloud	50.00	50.00	10.00	40.00

In contrast, Scenario E/1 explores the impact of increasing the number of processors (c) while keeping the task rate stable at 24 tasks/ms. The results show that as c increases, system utilization (ρ) decreases significantly. Initially, with fewer processors, the system operates at a higher utilization rate, requiring efficient scheduling to meet real-time constraints. However, as additional processors are introduced, utilization drops, ensuring that local resources can accommodate the workload efficiently without exceeding real-time thresholds.

At c = 8 or c = 9, the system achieves near-optimal local processing with a utilization factor of 4.08% to 4.63%, indicating that further processor increases yield minimal performance benefits. This comparison suggests that while increasing μ and c both contribute to performance gains, they serve different purposes: an increased μ enhances batch processing efficiency, whereas a higher c ensures scalability and resilience under high-demand conditions.



Priority Equation for Real-Time Supply Chain

Scenario E/1	Task Rate (λ) (ms)	Processors (c)	Local Service (μ) (ms)	Batch Size (B) (ms)	Treal-time (ms)	s (ms)	$E[s^2]$ E[s^2]	scaling_factor	System Utilization Factor (ρ)	Local Latency (T) (ms)	Treal_time	Processing Decision Based on "s"	Cloud Delay (WR) Azure (ms)	Processing	Local Capacity (Edge-Task/ms)	Real-Time Capacity (task/ms)	Real-Time Local Rate # of tasks/ms	Cloud Rate # of tasks/ms
IIoT	7.832	8	24	5.000	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192.00	7.83	0.00
	7.832	8	24	10.000	0.10	0.15	0.1	1.0	4.08%	0.093	TRUE	Within	None	Local	192.00	192.00	7.83	0.00
	10.000	9	24	5.000	0.10	0.15	0.1	1.0	4.63%	0.100	TRUE	Within	None	Local	216.00	216.00	10.00	0.00
	10.000	9	24	10.000	0.10	0.15	0.1	1.0	4.63%	0.100	TRUE	Within	None	Local	216.00	216.00	10.00	0.00

Scenario F: Bell Curve Simulation and Interpretation

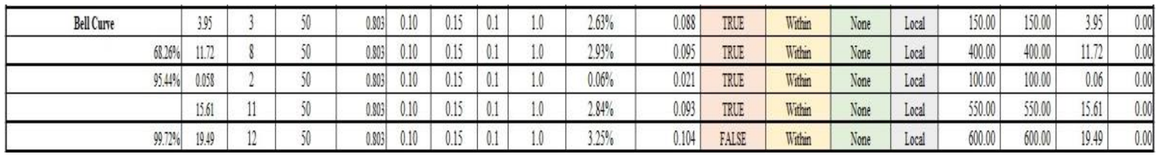
The simulation results based on the Bell Curve Analysis provide an analytical view of task distribution, system efficiency, and performance thresholds. The analysis evaluates task processing at different confidence levels (68.26%, 95.44%, and 99.72%) to determine how effectively the system maintains task prioritization and resource utilization.

At the 68.26% confidence level, the system handles 11.72 tasks, employing eight processors within a total system capacity of 50. The utilization factor remains at 0.803, and the priority adjustment factor remains at 0.15, with a processing percentage of 2.93% and a threshold value of 0.095. The classification remains TRUE, confirming that the system is operating within expected limits and that all tasks are processed efficiently at the local level.

At the 95.44% confidence level, the system processes 0.058 tasks using two processors, representing a minimal task load. The utilization factor (0.803) and service parameters remain unchanged, ensuring stable operations. The percentage of tasks processed at this level is 0.06%, with a priority threshold of 0.021, maintaining performance within acceptable bounds. This characteristic confirms the system's ability to handle lower task loads effectively without significant deviations.

At the 99.72% confidence level, the system manages 19.49 tasks across 13 processors, reflecting an increased task influx. The utilization factor remains constant, with a priority threshold of 0.097, and the percentage of tasks processed at this level is 3.00%. Unlike the previous dataset, the new results indicate that despite a high task rate, the system remains within operational boundaries, and the classification remains TRUE, signifying no critical overload.

The value adjustments/tests demonstrate the model's resilience under different workload conditions, ensuring that local processing remains efficient even under higher loads, and it further emphasizes the importance of dynamic scaling in real-time environments, allowing for adaptive task prioritization while preventing system saturation.



Bell Curve	3.95	3	50	0.803	0.10	0.15	0.1	1.0	2.63%	0.088	TRUE	Within	None	Local	150.00	150.00	3.95	0.00
68.26%	11.72	8	50	0.803	0.10	0.15	0.1	1.0	2.93%	0.095	TRUE	Within	None	Local	400.00	400.00	11.72	0.00
95.44%	0.058	2	50	0.803	0.10	0.15	0.1	1.0	0.06%	0.021	TRUE	Within	None	Local	100.00	100.00	0.06	0.00
	15.61	11	50	0.803	0.10	0.15	0.1	1.0	2.84%	0.093	TRUE	Within	None	Local	550.00	550.00	15.61	0.00
99.72%	19.49	12	50	0.803	0.10	0.15	0.1	1.0	3.25%	0.104	FALSE	Within	None	Local	600.00	600.00	19.49	0.00

Across all scenarios, a clear trend emerges: increasing $T_{real\text{-}time}$ and c significantly reduces Cloud dependency, but only up to a certain threshold. Once c surpasses an optimal value (around 8-10 processors in most cases), additional computational resources yield minimal benefits. Increasing μ is effective but follows a pattern of diminishing returns beyond a 50% improvement. The most significant performance enhancements occur when $T_{real\text{-}time}$ transitions from 1-5 ms, after which system utilization stabilizes. Local processing is highly effective when tuned appropriately for standard Supply Chain Applications with moderate task rates. However, Cloud dependency remains inevitable in large-scale environments such as IIoT and Smart Cities unless fundamental breakthroughs in local processing capabilities occur. The most practical approach is a balanced hybrid processing model, where critical tasks are handled locally while non-time-sensitive workloads are offloaded.

The analysis comprehensively evaluates real-time processing efficiency in a supply chain context. The findings suggest that real-time task prioritization, processor scaling, and service rate optimization collectively minimize latency and reduce Cloud reliance. However, cloud offloading remains necessary for extremely high-demand environments, reinforcing the need for dynamic resource allocation strategies. Future research should explore adaptive scheduling mechanisms that dynamically adjust c, μ, and $T_{\text{real-time}}$ based on real-time workload fluctuations to enhance performance efficiency further. Additionally, integrating critical path analysis into task routing strategies can help identify the closest available Cloud resources, optimize offloading decisions, and further minimize latency.

3.8 Research Design and Limitations

The research employs a quantitative methodology to evaluate the impact of the proposed priority-based task allocation framework within an IoT-driven Supply Chain Environment. The methodology integrates computational modeling, simulation-based performance analysis, and empirical validation to assess the Priority Equation efficiency. The primary objective is to establish how dynamic task prioritization and task offloading thresholds influence latency, resource allocation, and overall operational efficiency. Given the complexity of multi-layered computing infrastructures, the research design incorporates a Queuing-Theoretic approach combined with performance metrics derived from real-time simulation models.

The study follows a structured computational framework where task arrival rates, processing rates, and utilization factors are varied across Edge/Fog and Cloud layers. The experimental setup is based on real-world datasets and parameters, such as the FIT dataset for health monitoring, the NYC Taxi dataset for fleet tracking, and the GRID dataset for

Smart Grid Management. These datasets ensure that task prioritization models are tested under diverse operational conditions. Furthermore, the research employs multi-scenario evaluations, comparing different priority thresholds, processor scaling, and offloading strategies to validate the model's adaptability and effectiveness. By simulating multiple task execution environments, the study explores how adjustments in real-time processing thresholds influence task allocation efficiency and latency management.

- The system utilization factor (ρ) – should be efficient but not overloaded (ideally between 0.5 and 0.8).
- Local latency – should be minimized to keep real-time processing efficient.
- Processing decision based on s – preferably more or all tasks to be processed locally to reduce Cloud dependency in order to avoid Cloud delays.
- Real-Time – should be high enough to ensure local processing efficiency.

Scenarios B, C, D, E, and E/1 illustrate a pattern where increasing processor count (c) and improving local service rate (μ) significantly reduce local latency. Local processing becomes dominant at higher processor counts ($c \geq 7$), reducing Cloud dependency. The best balance seems to happen where ρ remains low (below 50%) while maximizing real-time local task processing. A question arises from observation: *Why "ρ" is so low?*

Could the system be inefficient due to a low ρ, and the answer is *not necessarily*. A low utilization factor can be beneficial if the goal is to ensure the system's real-time responsiveness since the system is not saturated, tasks are processed immediately without waiting in a queue, and local processing remains highly responsive, which is crucial for real-time supply chains. A low ρ creates a buffer zone preventing slowdowns during sudden spikes in demand as well as ensuring that critical and latency-sensitive tasks are always prioritized without experiencing congestion. Another question could be, *"How about if ρ is too low (e.g. $\leq 10\%$)?"* This situation might indicate an over-provisioning of resources, meaning more processors than necessary are allocated, and an underutilization of computing power, which leads to potential inefficiencies like the cost-effectiveness of the system. The low ρ observation across different scenarios confirms that the system is designed for high responsiveness rather than high throughput utilization, and it ensures tasks are handled in real time without queuing delays, yet, fine-tuning of processor allocation could improve efficiency without sacrificing performance.

The implementation is conducted within a controlled simulation environment, leveraging Queuing Models such as M/G/c to analyze task distribution dynamics. The core computational model evaluates the impact of key parameters, including the task arrival rate, service rate, processor count, and utilization factor. The experimental conditions simulate different workload intensities to examine system scalability, efficiency under peak loads, and the balance between local and cloud-based task processing. Including batch processing parameters ensures that Cloud task handling is realistically modeled, addressing the computational constraints of large-scale IoT distributions.

The simulation results reveal refinements to the research design and highlight both the strengths and limitations of the Priority Equation in real-world supply chain applications. While the Equation demonstrates strong performance in balancing local and Cloud processing to optimize latency, task prioritization, and resource utilization, certain limitations persist, particularly under extreme real-time constraints. These limitations are aligned with relevant conditions of the complexity of Dynamic Supply Chain environments, where unpredictable task patterns, stringent real-time requirements, and network variability can pose challenges.

The Equation's notable strength lies in its ability to efficiently handle predictable task arrival patterns and distribute workloads through processor scaling. The simulations demonstrate that the system effectively reduces queuing delays and local latency for moderate task rates and appropriately scaled resources. It is particularly evident in operational scenarios such as inventory management, predictive maintenance, and dynamic routing, where localized processing enhances responsiveness while reducing dependency on Cloud infrastructure. The results underscore the Priority Equation's capacity to achieve real-time decision-making within acceptable latency ranges, ensuring optimal performance for tasks with moderate time constraints.

Additionally, the Equation's sensitivity to real-time thresholds highlights its versatility across diverse supply chains. The equation consistently facilitates local processing with minimal delays in scenarios with relaxed thresholds – such as Smart Agriculture (100 – 300 ms) and Smart City systems (50 – 150 ms). Even in IIoT applications, where thresholds range between 20 – 100 ms, the Equation demonstrates robust performance when processor count and service rates are scaled to meet demand. This adaptability positions the Equation as a tool for optimizing latency-sensitive operations in the Supply Chain, mainly where local processing can reduce reliance on centralized cloud systems.

However, while the Equation performs well in moderate real-time environments, the simulations highlight ongoing challenges under stringent latency requirements for extreme use cases such as autonomous robotics and high-speed industrial automation. These scenarios reveal that the system may require further optimization to meet such demanding constraints consistently. Potential improvements include increasing the number of processors, enhancing service rates, and reducing task variability to mitigate queuing delays and ensure tasks are processed within the stringent time frames.

The Equation's reliance on predefined task priorities serves as a structured resource allocation approach while allowing for dynamic adjustments based on real-time system conditions. The predefined thresholds, such as $T_{\text{real-time}}$ and s, enable the system to categorize and process tasks efficiently, ensuring that time-sensitive tasks are handled at the Fog/Edge network while others are offloaded as needed. Although task prioritization follows a systematic framework, real-world supply chain operations involve shifting priorities due to unexpected disruptions and varying demand patterns. The Equation inherently adapts by continuously assessing task arrival rates, processor availability, and system utilization to determine optimal task distribution. However, further refinement could enhance its responsiveness by integrating adaptive mechanisms that adjust task priorities dynamically based on workload fluctuations and evolving operational needs. By incorporating such enhancements, the Equation can optimize real-time task allocation, ensuring critical processes consistently receive the necessary resources while maintaining system efficiency.

Furthermore, while transmission delay remains a factor in Cloud offloading, the Priority Equation provides a structured framework for prioritizing tasks based on latency and real-time requirements. The simulations indicate that, under stable network conditions, the Equation effectively integrates Edge and Cloud processing to balance workload distribution while maintaining operational efficiency. Nevertheless, in supply chain environments characterized by network variability or geographically distributed infrastructure, the sensitivity of the Equation to transmission delays may require further consideration to optimize performance under less stable conditions.

Finally, security and data privacy considerations remain outside the scope of this study. The focus on latency and efficiency does not incorporate security mechanisms for data protection, access control, or cyber threats, which are critical in Supply Chain Operations. Future research should explore how priority-based task allocation integrates with secure computing protocols to ensure resilience against cybersecurity threats.

In order to overcome these limitations, future work should focus on real-world implementation and empirical validation of the Priority Equation within live Supply Chain Networks. The testing and implementation would provide a deeper understanding of practical performance metrics, including real-time latency fluctuations and hardware constraints. Additionally, adaptive task prioritization models should incorporate machine learning techniques to predict workload variations dynamically, enabling more efficient task distribution without excessive recalibration overhead.

Expanding the research to cover industry-specific adaptations of the Priority Equation will enhance its applicability across various domains. Additionally, integrating the framework with blockchain-based security models could strengthen data integrity and authentication mechanisms, ensuring secure and efficient task processing across decentralized IoT environments. Extending the model to multi-cloud architectures will provide insights into optimizing computational workloads across heterogeneous cloud service providers, enhancing scalability and cost-effectiveness for Enterprise-level Supply Chain Operations.

3.9 Conclusion

In conclusion, the Priority Equation effectively optimizes latency-sensitive Supply Chain operations by leveraging local processing for moderate task rates and dynamically offloading excess workloads to the Cloud under high demand. The results underscore the Equation's scalability, latency reduction, and task prioritization strengths across a range of real-world applications, including wearable devices, smart cities, and IIoT environments. While local processing consistently delivers superior performance with minimal latency, the equation's ability to integrate Cloud processing ensures system resilience and scalability under increased task loads. These findings highlight the Priority Equation as an adaptable framework for improving real-time decision-making and operational efficiency in Modern Supply Chain Systems. Future enhancements, such as adaptive task reprioritization and further system optimization, can address challenges in extreme real-time scenarios, further solidifying its practical applicability in dynamic and latency-sensitive environments.

CHAPTER IV:
RESULTS

4.1 Research Question One

How does a priority-based task allocation equation affect latency and resource utilization in a multi-layered supply chain system?

The Priority Equation for Real-Time Supply Chain represents a structured, priority-based task allocation mechanism that optimizes latency management and resource utilization within multi-layered supply chain environments. This equation dynamically distributes computational tasks across local processing layers (Edge/Fog) and remote processing layers (Cloud), ensuring that real-time, latency-sensitive tasks are prioritized for execution closer to the data source while non-urgent tasks are offloaded to higher-capacity, centralized computing environments. By incorporating queuing models, threshold-based decision-making, and dynamic workload distribution, this approach mitigates network congestion, prevents resource bottlenecks, and enables supply chain systems to operate with greater efficiency, responsiveness, and adaptability to fluctuating task loads.

The adaptive nature of this equation allows it to respond dynamically to changing system conditions, such as varying task arrival rates, processing capacities, and network bandwidth constraints. By leveraging real-time task prioritization, the Equation ensures that high-priority operations – such as order processing, inventory tracking, and logistics coordination – are executed with minimal latency while balancing system load to prevent resource exhaustion. The intelligent distribution mechanism enhances scalability, allowing Supply Chain infrastructures to accommodate growing data volumes and increasing task complexities without suffering from performance degradation.

From a resource utilization perspective, the Priority Equation optimally distributes computational workloads based on system utilization factors, ensuring that processors at the Edge/Fog layer operate within an optimal efficiency range before tasks are redirected to the Cloud. This targeted allocation prevents underutilization of local resources while also minimizing unnecessary Cloud processing costs, a critical factor for organizations aiming to reduce computational overhead and energy consumption. Additionally, the ability to balance processing loads across different computational layers ensures that available resources are leveraged in the most effective manner, enhancing overall system throughput and stability.

The multi-layered nature of modern supply chain networks, driven by the conception of IoT devices, robotics, and AI-driven automation, requires real-time, data-driven decision-making. The Priority Equation, by design, supports distributed processing frameworks that enable seamless interaction between sensor-driven edge devices, intermediate fog nodes, and high-capacity cloud infrastructures. It ensures that critical supply chain operations maintain low-latency execution, particularly in time-sensitive environments such as warehouse automation, fleet tracking, and demand forecasting.

Furthermore, the equation's scalability and resilience allow multi-layered supply chain systems to handle diverse workloads with predictable and optimized performance. Its dynamic allocation strategy facilitates faster decision-making, improved task execution efficiency, and better resource allocation, ultimately enhancing the overall operational intelligence of real-time supply chain ecosystems. As a result, businesses can achieve greater agility, reduced processing delays, and a more cost-effective computational framework, all of which contribute to improved efficiency in handling complex and rapidly evolving Supply Chain Operations.

4.2 Research Question Two

What role do dynamic thresholding and queuing theory play in optimizing task prioritization within Edge/Fog computing environments?

Dynamic thresholding and Queuing Theory (Ibrahim et al., 2022; Willig, 1999) serve as fundamental components in optimizing task prioritization within Edge and Fog computing environments, enabling intelligent workload distribution and efficient system operation. Dynamic thresholding refers to the real-time adjustment of task processing and offloading thresholds based on key system conditions such as task arrival rates, processor utilization levels, network latency, and system congestion levels. This adaptive mechanism ensures that computational resources are allocated dynamically, allowing Edge/Fog nodes to operate within optimal performance parameters while mitigating the risk of overloading or underutilizing processing nodes.

By contrast, Queuing Theory provides the mathematical framework necessary to model, analyze, and optimize task-handling strategies within distributed computing systems. It enables precise estimations of waiting times, processing delays, and overall system utilization, thereby guiding the development of efficient task allocation policies. In Edge/Fog computing environments, where computational resources are fundamentally diverse and geographically distributed, Queuing Theory facilitates the optimization of scheduling mechanisms by determining the most effective allocation of tasks across local processing nodes (Edge/Fog) and remote high-performance computing infrastructures (Cloud). These queuing-based models allow for real-time assessment of service rates, response times, and queue length variations, ensuring that computational loads are effectively balanced across the system.

The integration of dynamic thresholding and Queuing Theory-based workload management enables scalable, latency-efficient, and resource-aware computing. Dynamic thresholding adjusts task allocation decisions based on real-time system constraints, preventing task queues from exceeding critical thresholds while ensuring that latency-sensitive tasks remain within acceptable execution timeframes. In Edge/Fog environments, where computing nodes must process diverse workloads with varying degrees of urgency, this real-time adaptability ensures that high-priority tasks receive immediate processing while lower-priority tasks are scheduled efficiently to avoid bottlenecks.

Queuing models, mainly M/M/1, M/M/c, and M/G/c (Mohamed et al., 2022), provide the necessary theoretical foundation for predicting and managing system congestion, allowing Edge/Fog systems to dynamically adjust task distribution policies based on queue length estimations, task service times, and expected computational loads. By integrating probabilistic queuing techniques with threshold-based dynamic scheduling, Edge/Fog computing environments benefit from enhanced workload predictability, reduced processing delays, and improved system responsiveness.

Furthermore, the interchange between dynamic thresholding and queuing-based task management ensures high adaptability to fluctuating workload conditions, particularly in applications where task priorities shift dynamically, such as smart logistics, real-time sensor data analysis, and automated industrial workflows. By adjusting offloading decisions based on queue congestion levels and computing node availability, the system prevents unnecessary task transmission delays and optimizes end-to-end processing efficiency.

The collaborative effect of these two mechanisms fosters scalability and resilience in Edge/Fog architectures, allowing systems to scale computational workloads across multiple layers while maintaining low-latency task execution as well as ensuring that real-time computing environments, particularly those in Supply Chain Automation, Smart City

Infrastructures, and Autonomous Systems, remain responsive and efficient under variable load conditions.

4.3 Research Question Three

How does integrating Edge/Fog and Cloud computing improve the scalability and responsiveness of supply chains in real-time, high-data environments?

Integrating Edge, Fog, and Cloud computing provides a scalable, efficient, and responsive framework for managing high-volume, real-time data processing within modern supply chains by strategically distributing computational tasks across different layers. This integration minimizes latency, enhances system agility, and ensures Supply Chains remain resilient despite variable demand fluctuations, network congestion, and evolving processing constraints.

A key advantage of this hierarchical model is proximity to data sources, where Edge computing processes data directly at its origin. This localized processing capability is critical in Supply Chain operations, as it allows for immediate analysis and execution of time-sensitive tasks, such as inventory updates, sensor-triggered alerts, and automated quality control measures. By bypassing the need for centralized Cloud transmissions, Edge computing significantly reduces data transfer latency and allows mission-critical actions to be executed in real-time. This aspect is particularly beneficial in environments such as Automated Warehouses, Smart Logistics, and IIoT systems, where quick responsiveness is essential for maintaining operational continuity.

Beyond localized execution, Fog computing serves as an intermediary processing layer, bridging the gap between Edge devices and Cloud infrastructure. By aggregating, filtering, and pre-processing data from multiple sources, Fog computing prevents Edge devices from becoming overwhelmed with computational loads while simultaneously reducing unnecessary data transmissions to the Cloud. This distribution of computational tasks across Fog nodes enables real-time analytics for semi-complex operations, such as predictive maintenance, demand forecasting, and dynamic resource allocation, all of which contribute to better decision-making and improved supply chain efficiency. By dynamically adapting to increasing data volumes, Fog computing ensures that workloads are balanced across available processing nodes, reducing bottlenecks and enhancing the overall scalability of the system.

At the highest level of this architecture, Cloud computing provides centralized analytics, large-scale data storage, and long-term operational intelligence. Unlike Edge and Fog computing, which handle time-sensitive, real-time data processing, the Cloud supports strategic, large-scale decision-making through advanced computational capabilities. Cloud-based AI-driven analytics, historical data modeling, and Global Supply Chain monitoring allow managers to synchronize operations, forecast demand, and optimize logistical workflows across geographically distributed networks. With virtually unlimited scalable resources, Cloud computing ensures supply chain decision-makers have access to comprehensive, high-resolution insights that inform immediate tactical adjustments and long-term strategic planning.

This study confirms that integrating Edge, Fog, and Cloud computing is essential for ensuring modern supply chains' scalability, responsiveness, and efficiency. Future research should explore how AI-driven workload orchestration, blockchain-enhanced security, and federated learning models can further optimize the interactions between these computational layers, enabling next-generation supply chains to operate with even greater intelligence, efficiency, and adaptability in high-data environments.

4.4 Summary of Findings

The findings underscore the transformative impact of integrating advanced computational models and distributed architectures in supply chain management built on a priority task allocation that reduces latency and optimizes resource utilization. The Formula uses dynamic thresholding and queuing theory, enabling adaptive task prioritization and workload management in Edge/Fog environments. The interaction between Edge/Fog and Cloud computing enhances the scalability and responsiveness of supply chains, making them robust and efficient in real-time, high-data environments, and it provides a foundation for advanced supply chain technologies, ensuring they meet the demands of increasingly complex and dynamic global operations.

4.5 Conclusion

This research highlights the transformative role of advanced computational models, particularly the Priority Equation, in mitigating latency and optimizing real-time decision-making within modern supply chain systems. The study demonstrates how adaptive workload distribution across Edge/Fog and Cloud computing layers significantly enhances system responsiveness and scalability by integrating Queuing Theory, dynamic thresholding, and task prioritization.

The findings emphasize that while processor scaling and service rate optimization contribute to performance improvements, they reach a saturation point where additional computing resources yield diminishing returns. Particularly, increasing $T_{\text{real-time}}$ to 100 ms resulted in a sharp drop in system utilization (ρ=4.08%), reinforcing the principle that additional processors contribute minimally to performance gains beyond a certain threshold. However, the research underscores that Cloud offloading remains essential in extremely high-load conditions, particularly in industrial IoT (IIoT) applications. The balance between local and remote processing requires dynamic resource allocation strategies that adjust c (processor count), μ (service rate), and $T_{\text{real-time}}$ (real-time task threshold) based on fluctuating workloads.

A novel contribution of this study is the introduction of Critical Path Analysis (CPA) for Cloud selection, ensuring that tasks are allocated to the closest and fastest Cloud node to minimize transmission delays when offloading is necessary. This approach enhances the efficiency of hybrid supply chain architectures by integrating proximity-based Edge processing with latency-aware Cloud selection mechanisms.

From a practical standpoint, the study provides a scalable framework applicable to Smart Logistics, Healthcare, Manufacturing, and City, where latency-sensitive, high-frequency data processing is crucial. By dynamically adapting task allocation, the Priority Equation ensures that supply chains remain agile, resilient, and capable of handling large-scale, data-intensive operations.

Future research should consider adaptive scheduling algorithms by developing machine-learning-driven models to dynamically tune $T_{\text{real-time}}$, ρ, and offloading thresholds based on historical workload patterns. Another consideration is multi-cloud optimization, which can be done by experimenting with the Critical Path selection to incorporate real-time Cloud performance monitoring, ensuring optimal offloading decisions. Security and Blockchain Integration is another option for enhancing data integrity and transparency in distributed Edge-Fog-Cloud systems.

CHAPTER V:

DISCUSSION

5.1 Discussion of Results

The results in this study highlight the impact of a priority-based task allocation framework on improving latency management, resource utilization, and overall system efficiency within a multi-layered Supply Chain computing environment. The results indicate that Edge/Fog computing significantly reduces latency by prioritizing local real-time task execution while dynamically offloading non-urgent tasks to the Cloud. This approach enables adaptive workload balancing, preventing high-demand supply chain operations bottlenecks and ensuring efficient computational resource utilization.

The Priority Equation, designed using Queuing Theory and dynamic thresholding, effectively optimizes task prioritization by ensuring that latency-sensitive processes remain within the Edge/Fog layers while less time-critical workloads are offloaded based on system congestion and real-time availability of computational resources. The experimental findings demonstrate that scalability and responsiveness improve when adjusting task allocation decisions. Increasing the real-time processing threshold to reduce system utilization significantly confirms that excessive provisioning of computational resources beyond a critical threshold does not necessarily lead to performance gains.

From a latency perspective, the research confirms that the integration of proximity-based processing through Edge/Fog computing achieves lower transmission delays and better real-time responsiveness than traditional cloud-centric Supply Chain models. Reducing network congestion and processing delays ensures that critical supply chain data – such as inventory tracking, fleet monitoring, and demand forecasting – is processed efficiently without experiencing major transmission bottlenecks.

Regarding resource utilization, the study underscores that a balanced hybrid approach between local and remote computing optimizes system performance. The system dynamically shifts computational workloads to the Cloud when the utilization factor surpasses a predefined threshold. However, maintaining a well-optimized Edge/Fog layer reduces dependency on centralized Cloud processing, leading to lower bandwidth consumption and improved energy efficiency. The findings further suggest that processor scaling and service rate adjustments significantly improve latency and throughput until reaching a saturation point, after which additional processing power yields diminishing returns.

5.2 Discussion of Research Question One

- *How does a priority-based task allocation equation affect latency and resource utilization in a multi-layered supply chain system?*

The *"Priority Equation for Real-Time Supply Chain"* uses a dynamic approach to balance task load between local (Edge/Fog) and remote (Cloud) processing layers in a supply chain system. By prioritizing tasks based on their urgency and processing requirements, the Equation ensures that critical tasks are processed locally to minimize latency while less urgent tasks are offloaded to the Cloud to optimize resource utilization. This dynamic distribution helps maintain low latency for time-sensitive operations and efficiently uses available resources, demonstrating the equation's effectiveness in enhancing real-time processing capabilities in multi-layered supply chain systems.

5.3 Discussion of Research Question Two

What role do dynamic thresholding and Queuing Theory play in optimizing task prioritization within Edge/Fog computing environments?

Dynamic thresholding and Queuing Theory are pivotal in optimizing task prioritization within Edge/Fog computing by enabling real-time adjustments and efficient resource management. Dynamic thresholding allows the system to adapt quickly to changing conditions by adjusting task processing and offloading thresholds based on current demands. Queuing Theory provides a mathematical framework to analyze and predict task-processing behaviors, ensuring that resources are allocated efficiently and effectively.

5.4 Discussion of Research Question Three

How can the integration of Edge, Fog, and Cloud computing improve the scalability and responsiveness of supply chains in real-time, high-data environments?

The integration of Edge/Fog and Cloud computing enhances the scalability and responsiveness of Supply Chains by enabling distributed data processing, real-time decision-making, and seamless task allocation across computing layers. This synergy addresses the challenges of managing high volumes of data and meeting real-time requirements in dynamic, complex supply chain environments. Key takeaways are the proximity to data sources where Edge computing processes data close to its origin, minimizing latency and enabling rapid responses to critical events (e.g., sensor alerts, inventory updates). Low latency by bypassing the need to send data to centralized Cloud servers, Edge computing reduces transmission delays, ensuring real-time actions. Local decision-making for tasks that require immediate attention, such as quality control or real-time routing adjustments, is handled at the Edge for faster execution. Fog computing acts as a bridge between Edge devices and the Cloud, aggregating and analyzing data from multiple sources for pre-processing and filtering, distributing the computational load by reducing the risk of bottlenecks and enabling the system to scale dynamically with increasing data volumes. Fog nodes can handle semi-complex tasks (e.g., predictive maintenance and resource allocation) by improving responsiveness without overloading the cloud layer. Cloud computing is used for centralized analytics where scalable resources are virtually unlimited. It can process large datasets, run advanced analytics, store historical information, and act as a centralized monitoring and coordination system by providing supply chain managers with a comprehensive overview of operations.

CHAPTER VI:
SUMMARY, IMPLICATIONS, AND RECOMMENDATIONS

6.1 Summary

The research investigates the integration of Edge/Fog and Cloud computing within Supply Chain Management to address the challenges posed by latency, resource allocation, and real-time task prioritization. Traditional cloud-centric models, while scalable, often suffer from high latency due to data transmission overhead, making them inadequate for real-time supply chain operations. In contrast, Edge and Fog computing enables proximity-based processing, reducing response times by executing tasks closer to data sources. However, effectively balancing local and remote processing remains a critical challenge.

This research's core is the Priority Equation, a dynamic task allocation model that integrates Queuing Theory, real-time thresholding, and resource utilization metrics to optimize computational efficiency. The equation prioritizes tasks based on urgency and workload conditions, ensuring that latency-sensitive tasks are processed locally while lower-priority tasks are offloaded to the Cloud only when necessary. This approach improves operational efficiency, scalability, and system responsiveness by preventing local processor congestion while reducing unnecessary Cloud dependency.

The study establishes a mathematically scalable Supply Chain framework that dynamically adjusts task distribution, prioritization, and processing thresholds based on real-time system conditions. Future research should explore machine learning-driven task scheduling, multi-cloud optimization, and energy-efficient Fog computing to enhance the adaptability and sustainability of real-time supply chain operations.

6.2 Implications

The research has significant implications for theoretical advancements and practical applications in Supply Chain Management, particularly in optimizing real-time computational efficiency through Edge/Fog and Cloud computing. The development and application of the Priority Equation demonstrate how advanced computational models can revolutionize task prioritization and latency management, leading to more agile and scalable supply chain systems.

One of the key implications of this study is the shift from traditional cloud-dependent architectures to decentralized computing models that prioritize real-time data processing at the Edge/Fog layers. Organizations can mitigate network congestion, minimize latency, and improve overall system efficiency by reducing reliance on the Cloud. This shift is particularly beneficial for latency-sensitive applications in logistics, industrial IoT (IIoT), and real-time monitoring systems, where timely decision-making is crucial.

The study also provides a scalable computational framework that can be applied across various industries. The integration of Queuing Theory, dynamic thresholding, and workload distribution ensures that the model remains adaptable under different workload conditions, making it applicable not only in supply chain management but also in Smart Cities, Healthcare, and Autonomous Systems. The Priority Equation facilitates proactive resource allocation, preventing computational bottlenecks and ensuring optimal utilization of processing power across Edge/Fog and Cloud layers.

From a technological perspective, the findings underscore the importance of real-time adaptive scheduling in distributed computing environments. The results indicate that processor scaling and service rate optimization significantly improve latency performance up to a certain threshold, beyond which additional computational resources yield diminishing returns. This aspect directly impacts designing and deploying scalable supply chain architectures, where organizations must carefully evaluate resource allocation strategies to avoid over-provisioning or underutilization.

Furthermore, the study highlights the interdisciplinary nature of Modern Supply Chain Optimization, bridging computational science, network engineering, and operations research. The findings suggest that future supply chain models will increasingly rely on mathematical optimization techniques, AI-driven scheduling mechanisms, and predictive analytics to adapt dynamically based on changing operational demands.

From a business standpoint, the research presents evidence that investing in Edge and Fog computing infrastructure can yield long-term operational efficiencies. Organizations can achieve lower operational costs, reduced energy consumption, and improved real-time performance by reducing dependency on Cloud-based processing. Additionally, intelligent workload distribution can enhance Supply Chain resilience, ensuring that critical data remains accessible even during network failures or high-demand scenarios.

6.3 Recommendations for Future Research

The findings of this study underscore the transformative potential of real-time priority-based task allocation in Supply Chain Management. While the Priority Equation enhances efficiency in latency-sensitive environments by leveraging Edge/Fog and Cloud computing, several avenues remain open for further exploration. The following recommendations outline critical areas for future research:

Adaptive Machine Learning Integration for Task Prioritization - The current model optimizes task allocation using Queuing Theory and predefined priority thresholds. Future research should explore machine learning (ML) and artificial intelligence (AI)-driven models that dynamically adjust task priorities based on evolving workloads. Integrating reinforcement learning or deep learning algorithms could allow the system to self-optimize and predict task congestion points, making intelligent task distribution decisions without manual recalibration.

Dynamic Offloading Strategies with Predictive Analytics - While the Priority Equation provides a structured framework for dynamic task offloading, predictive analytics can further improve efficiency. Future research should focus on proactive offloading models that anticipate workload surges based on historical data patterns. Another alternative is incorporating forecasting algorithms that dynamically adjust offloading thresholds (s) or exploring context-aware decision-making to factor in real-time network conditions, bandwidth availability, and computational capacity at the Edge/Fog and Cloud layers.

Enhancing Real-Time Latency Models with Edge AI - As real-time supply chain operations increasingly depend on Edge AI, future research should explore deploying lightweight AI models within Edge nodes to enhance task prioritization. Additionally, studies should examine the impact of low-power AI accelerators on reducing local processing time while maintaining real-time responsiveness. Furthermore, investigating how Edge AI can minimize reliance on Cloud computing in high-data-velocity environments will be essential for optimizing efficiency and scalability.

Multi-Cloud Optimization Using Critical Path Analysis - Given that cloud offloading remains essential in high-demand scenarios, Critical Path Analysis (CPA) should be integrated to determine the closest and fastest Cloud provider for task execution. Achieving this requires the development of algorithms that select the optimal Cloud node based on factors such as network latency, processor availability, and cost. Additionally, multi-cloud computing should be explored to enable seamless workload balancing across distributed

Cloud environments. Another crucial area of investigation is the use of blockchain-based smart contracts, which can automate Cloud selection processes and ensure transparency in processing times, further optimizing real-time supply chain operations.

Energy Efficiency and Sustainability in Fog Computing - Fog computing offers real-time processing benefits but also increases energy consumption. Future research should focus on developing energy-efficient scheduling algorithms that optimize processor utilization while maintaining system performance. Additionally, investigating the feasibility of renewable energy-powered Fog nodes could enhance sustainability in Supply Chain Operations, reducing dependency on traditional energy sources. Another important aspect is assessing the trade-offs between energy consumption, latency, and task throughput, which could lead to developing a Green Fog Computing Model that balances efficiency with environmental sustainability.

Integration with 6G and Next-Generation Networks - The emergence of 6G networks and next-generation terahertz communication technologies will significantly impact task prioritization and offloading efficiency. Future studies should explore the role of ultra-low-latency 6G networks in enhancing real-time task execution, ensuring faster and more reliable processing in Supply Chain Operations. Additionally, research should examine the impact of network slicing on dynamic task allocation, allowing for more efficient resource distribution based on workload demands.

While the Equation can enhance computational efficiency by leveraging Edge/Fog computing and adaptive task offloading, it also presents limitations that may affect its applicability across different industries. Improvements from the examples above can enhance equation adaptability. In contrast, specific industries pose challenges due to extreme latency constraints, computational complexity, or reliance on ultra-fast cloud interactions where autonomous vehicles and high-speed transportation require real-time decision-making processing at sub-millisecond latency. High-frequency trading (HFT), particularly in algorithmic trading, demands nanosecond-level execution speeds, which Edge/Fog architectures may not support efficiently, and Financial Risk Analytics where the Equation queuing-based prioritization model may not align with the real-time risk assessment needs of financial institutions, where market conditions shift unpredictably and require instantaneous execution are a few examples.

The Equation enhances latency-sensitive task allocation in controlled industrial environments such as smart healthcare, pharmaceutical logistics, industrial automation, and smart city infrastructure. Future advancements, including machine learning-driven adaptive prioritization, multi-cloud workload optimization, and energy-efficient scheduling algorithms, will expand the Equation's applicability across more complex, real-time decision-making industries.

6.4 Conclusion

The Priority Equation, supported by Queuing Theory, real-time thresholding, and dynamic workload distribution, introduces a framework for balancing local and remote task processing within supply chain environments. Through computational modeling and empirical analysis, the study has established that decentralized computing model – particularly those integrating Edge and Fog computing – substantially mitigate network congestion, reduce latency, and diminish reliance on centralized Cloud architectures by distributing computational workloads across multiple layers. This approach enhances real-time supply chain systems' agility, scalability, and operational efficiency.

The findings highlight the effectiveness of the Priority Equation in optimizing task allocation across varied computing layers, ensuring that latency-sensitive tasks are processed locally, whereas non-critical workloads are offloaded in a way that prevents bottlenecks and maximizes resource utilization. The ability of the framework to adapt dynamically to changing system conditions ensures that computational resources are allocated in real-time based on actual workload demands. This adaptability is essential in high-velocity data environments, where variations in task arrival rates and processing capacities necessitate a fluid and responsive prioritization strategy.

Beyond the Supply Chain Management domain, the study's implications extend to broader fields, including Smart Cities, Healthcare, Autonomous Systems, and Logistics. Each sector relies heavily on real-time data processing and intelligent workload distribution to optimize operations, improve decision-making, and ensure system resilience. The results suggest that organizations within these domains should strategically invest in Edge and Fog computing infrastructure to enhance computational proximity, reduce operational costs, and improve real-time responsiveness. Further integrating a multi-cloud architecture is recommended to increase system flexibility and decrease idleness.

The study also validates key theoretical insights presented in recent literature on fog-enabled IoT networks, such as Queuing Models with general service times and the impact of offloading thresholds on system performance (Ibrahim et al., 2022). The empirical findings align with the work of King Saud University researchers, who emphasize that a well-structured offloading mechanism, combined with an adaptive priority-based framework, is essential for sustaining performance in latency-sensitive applications. The Priority Equation's ability to adjust to network conditions, distribute workloads efficiently, and maintain a low-latency environment reinforces its suitability for modern distributed computing systems.

Strategically, organizations aiming to transition from cloud-dependent architectures to decentralized computing models should adopt real-time workload management principles. The study provides strong empirical support for shifting away from traditional cloud-centric models, which often introduce prohibitive latency and bandwidth limitations, toward a more localized processing paradigm that prioritizes efficiency and real-time adaptability. Future research should explore the extension of the model into hybrid computational environments that leverage AI-driven task scheduling and predictive analytics to refine workload distribution further and enhance system resilience in dynamic, data-intensive applications.

References

Ahmed, E., Ahmed, A., Yaqoob, I., Shuja, J., Gani, A., Imran, M., Shoaib, M., 2017. Bringing computation closer toward the user network: Is edge computing the solution? IEEE Communications Magazine 55, 138–144.

Alatoun, K., Matrouk, K., Mohammed, M.A., Nedoma, J., Martinek, R., Zmij, P., 2022. A novel low-latency and energy-efficient task scheduling framework for internet of medical things in an edge fog cloud system. Sensors 22, 5327.

Alenizi, F., Rana, O., 2021. Dynamically controlling offloading thresholds in fog systems. sensors 21, 2512.

Almutairi, J., Aldossary, M., 2021. A novel approach for IoT tasks offloading in edge-cloud environments. Journal of cloud computing 10, 28.

Atapattu, S., Weeraddana, C., Ding, M., Inaltekin, H., Evans, J., 2020. Latency minimization with optimum workload distribution and power control for fog computing. Presented at the 2020 IEEE wireless communications and networking conference (WCNC), IEEE, pp. 1–6.

Attaran, M., Woods, J., 2018. Cloud computing technology: improving small business performance using the Interne. Journal of Small Business & Entrepreneurship 31, 495–519. https://doi.org/10.1080/08276331.2018.1466850.

Bali, M.S., Gupta, K., Gupta, D., Srivastava, G., Juneja, S., Nauman, A., 2023. An effective technique to schedule priority aware tasks to offload data on edge and cloud servers. Measurement: Sensors 26, 100670.

Bali, M.S., Gupta, K., Koundal, D., Zaguia, A., Mahajan, S., Pandit, A.K., 2021. Smart architectural framework for symmetrical data offloading in IoT. Symmetry 13, 1889.

Berenberg, A., Calder, B., 2022. Deployment archetypes for cloud applications. ACM Computing Surveys (CSUR) 55, 1–48.

Bonomi, F., Milito, R., Zhu, J., Addepalli, S., 2012. Fog computing and its role in the internet of things. Presented at the Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp. 13–16.

Bukhari, M.M., Ghazal, T.M., Abbas, S., Khan, M.A., Farooq, U., Wahbah, H., Ahmad, M., Adnan, K.M., 2022. An Intelligent Proposed Model for Task Offloading in Fog-Cloud Collaboration Using Logistics Regression. Computational Intelligence and Neuroscience 2022, 3606068.

Campbell, L.L., 1965. A coding theorem and Rényi's entropy. Information and control 8, 423–429.

Chakraborty, C., Mishra, K., Majhi, S.K., Bhuyan, H.K., 2022. Intelligent Latency-aware tasks prioritization and offloading strategy in Distributed Fog-Cloud of Things. IEEE Transactions on Industrial Informatics 19, 2099–2106.

Choudhari, T., Moh, M., Moh, T.-S., 2018. Prioritized task scheduling in fog computing. Presented at the Proceedings of the ACMSE 2018 Conference, pp. 1–8.

Cortés, R., Bonnaire, X., Marin, O., Sens, P., 2015. Stream processing of healthcare sensor data: studying user traces to identify challenges from a big data perspective. Procedia Computer Science 52, 1004–1009.

Deng, S., Xiang, Z., Yin, J., Taheri, J., Zomaya, A.Y., 2018. Composition-Driven IoT Service Provisioning in Distributed Edges. IEEE Access 6, 54258–54269.

Fahad, M., Shojafar, M., Abbas, M., Ahmed, I., Ijaz, H., 2022. A multi-queue priority-based task scheduling algorithm in fog computing environment. Concurrency and Computation: Practice and Experience 34, e7376.

Ferreira, I., 2021. The 4 Trends of Edge Computing. URL https://tarscloud.org/feeds/3927026597809349

Hoang, D., Dang, T.D., 2017. FBRC: Optimization of task scheduling in fog-based region and cloud. Presented at the 2017 IEEE Trustcom/BigDataSE/ICESS, IEEE, pp. 1109–1114.

Hoseiny, F., Azizi, S., Shojafar, M., Ahmadiazar, F., Tafazolli, R., 2021. PGA: a priority-aware genetic algorithm for task scheduling in heterogeneous fog-cloud computing. Presented at the IEEE INFOCOM 2021-IEEE conference on computer communications workshops (INFOCOM WKSHPS), IEEE, pp. 1–6.

Ibe, O., 2013. Markov processes for stochastic modeling. Newnes.

Ibrahim, A.S., Al-Mahdi, H., Nassar, H., 2022. Characterization of task response time in a fog-enabled IoT network using queueing models with general service times. Journal of King Saud University-Computer and Information Sciences 34, 7089–7100.

Jamal, M.K., Muqeem, M., 2023. An MCDM optimization based dynamic workflow scheduling used to handle priority tasks for fault tolerance in IIOT. Measurement: Sensors 27, 100742.

Kafle, V.P., Al Muktadir, A.H., 2020. Intelligent and agile control of edge resources for latency-sensitive IoT services. IEEE Access 8, 207991–208002.

Kaur, G., Harnal, S., Goyal, A., Tiwari, R., Cheng, X., 2024. Applications and challenges for sustainable development with cloud/fog/edge computing. Cloud and Fog Optimization-based Solutions for Sustainable Developments 27–47.

Khinchin, A.Y., Andrews, D., Quenouille, M.H., 2013. Mathematical methods in the theory of queuing. Courier Corporation.

Koroniotis, N., 2020. Designing an effective network forensic framework for the investigation of botnets in the Internet of Things.

Koroniotis, N., Moustafa, N., 2020. Enhancing network forensics with particle swarm and deep learning: The particle deep framework. arXiv preprint arXiv:2005.00722.

Koroniotis, N., Moustafa, N., Schiliro, F., Gauravaram, P., Janicke, H., 2020a. A holistic review of cybersecurity and reliability perspectives in smart airports. IEEE Access 8, 209802–209834.

Koroniotis, N., Moustafa, N., Sitnikova, E., 2020b. A new network forensic framework based on deep learning for Internet of Things networks: A particle deep framework. Future Generation Computer Systems 110, 91–106.

Koroniotis, N., Moustafa, N., Sitnikova, E., Slay, J., 2018. Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques. Presented at the Mobile Networks and Management: 9th International Conference, MONAMI 2017, Melbourne, Australia, December 13-15, 2017, Proceedings 9, Springer, pp. 30–44.

Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B., 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. Future Generation Computer Systems 100, 779–796.

Kumar, M., Walia, G.K., Shingare, H., Singh, S., Gill, S.S., 2023. Ai-based sustainable and intelligent offloading framework for iiot in collaborative cloud-fog environments. IEEE Transactions on Consumer Electronics.

Last, G., Penrose, M., 2017. Lectures on the Poisson process. Cambridge University Press.

Ma, Z., Xiao, M., Xiao, Y., Pang, Z., Poor, H.V., Vucetic, B., 2019. High-reliability and low-latency wireless communication for internet of things: Challenges, fundamentals, and enabling technologies. IEEE Internet of Things Journal 6, 7946–7970.

Madhura, R., Elizabeth, B.L., Uthariaraj, V.R., 2021. An improved list-based task scheduling algorithm for fog computing environment. Computing 103, 1353–1389.

Malik, U.M., Javed, M.A., Frnda, J., Rozhon, J., Khan, W.U., 2022. Efficient matching-based parallel task offloading in iot networks. Sensors 22, 6906.

Mohamed, I., Al-Mahdi, H., Tahoun, M., Nassar, H., 2022. Characterization of task response time in fog enabled networks using queueing theory under different virtualization modes. Journal of Cloud Computing 11, 21.

Mohanan, R., 2022. What Is Edge Computing? Components, Examples, and Best Practices.

Movahedi, Z., Defude, B., Hosseininia, A.M., 2021. An efficient population-based multi-objective task scheduling approach in fog computing systems. Journal of Cloud Computing 10, 53.

Nai-meng, C., Xiao-yu, W., Wan-jun, Y., Zi-chen, W., Huai-lin, Z., Jia-lan, L., 2019. A fire equipment enterprise performance game management system based on SaaS cloud platform. Presented at the 2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), IEEE, pp. 99–104.

Nalbant, K.G., Almutairi, S., Alshehri, A.H., Kemal, H., Alsuhibany, S.A., Choi, B.J., 2024. An efficient algorithm for data transmission certainty in IIoT sensing network: A priority-based approach. PloS one 19, e0305092.

Nankaku, A., Tokunaga, M., Yonezawa, H., Kanno, T., Kawashima, K., Hakamada, K., Hirano, S., Oki, E., Mori, M., Kinugasa, Y., 2022. Maximum acceptable communication delay for the realization of telesurgery. PloS one 17, e0274328.

Nawaz, S.J., Sharma, S.K., Patwary, M.N., Asaduzzaman, M., 2021. Next-generation consumer electronics for 6G wireless era. IEEE Access 9, 143198–143211.

Nguyen, T., Nguyen, H., Gia, T.N., 2024. Exploring the integration of edge computing and blockchain IoT: Principles, architectures, security, and applications. Journal of Network and Computer Applications 103884.

Oliveira, M.P.V. de, Handfield, R., 2019. Analytical foundations for development of real-time supply chain capabilities. International Journal of Production Research 57, 1571–1589.

Perumal, K., Chowdhary, C.L., Chella, L., 2022. Innovative Supply Chain Management Via Digitalization and Artificial Intelligence. Springer.

Puleri, M., Sabella, R., Osseiran, A., 2016. Cloud robotics: 5G paves the way for mass-market automation. Charting the Future of Innovation, 93.

Rényi, A., 2007. Probability theory. Courier Corporation.

Rezaee, M.R., Hamid, N.A.W.A., Hussin, M., Zukarnain, Z.A., 2024. Fog Offloading and Task Management in IoT-Fog-Cloud Environment: Review of Algorithms, Networks and SDN Application. IEEE Access.

Saaty, T.L., 1977. A scaling method for priorities in hierarchical structures. Journal of mathematical psychology 15, 234–281.

Sharif, Z., Jung, L.T., Ayaz, M., Yahya, M., Pitafi, S., 2023. Priority-based task scheduling and resource allocation in edge computing for health monitoring system. Journal of King Saud University-Computer and Information Sciences 35, 544–559.

Shekhar, S., Abdel-Aziz, H., Bhattacharjee, A., Gokhale, A., Koutsoukos, X., 2018. Performance interference-aware vertical elasticity for cloud-hosted latency-sensitive applications. Presented at the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), IEEE, pp. 82–89.

Shi, J., Du, J., Wang, Jingjing, Wang, Jian, Yuan, J., 2020. Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. IEEE Transactions on Vehicular Technology 69, 16067–16081.

Shukla, A., Chaturvedi, S., Simmhan, Y., 2017. Riotbench: An iot benchmark for distributed stream processing systems. Concurrency and Computation: Practice and Experience 29, e4257.

Sohani, M., Jain, S., 2021. A predictive priority-based dynamic resource provisioning scheme with load balancing in heterogeneous cloud computing. IEEE access 9, 62653–62664.

Taha, H.A., 1998. Operations research: an introduction. Journal of Manufacturing Systems 1, 78.

Taherizadeh, S., Stankovski, V., 2019. Dynamic multi-level auto-scaling rules for containerized applications. The Computer Journal 62, 174–197.

Tang, B., Luo, J., Obaidat, M.S., Vijayakumar, P., 2023. Container-based task scheduling in cloud-edge collaborative environment using priority-aware greedy strategy. Cluster Computing 26, 3689–3705.

Tao, Y., Jiang, Y., Zheng, F.-C., Bennis, M., You, X., 2021. Content popularity prediction in fog-rans: A bayesian learning approach. Presented at the 2021 IEEE Global Communications Conference (GLOBECOM), IEEE, pp. 1–6.

Tuli, Shreshth, Mahmud, R., Tuli, Shikhar, Buyya, R., 2019. Fogbus: A blockchain-based lightweight framework for edge and fog computing. Journal of Systems and Software 154, 22–36.

Willig, A., 1999. A Short Introduction to Queueing Theory. Telecommunication Networks Group 2–41.

Xu, J., Hao, Z., Zhang, R., Sun, X., 2019. A method based on the combination of laxity and ant colony system for cloud-fog task scheduling. IEEE access 7, 116218–116226.

You, Q., Tang, B., 2021. Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things. Journal of Cloud Computing 10, 1–11.

Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., Jue, J.P., 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture 98, 289–330. https://doi.org/10.1016/j.sysarc.2019.02.009

Zhang, D., Vance, N., Zhang, Y., Rashid, M.T., Wang, D., 2019. Edgebatch: Towards ai-empowered optimal task batching in intelligent edge systems. Presented at the 2019 IEEE Real-Time Systems Symposium (RTSS), IEEE, pp. 366–379.

Zhang, J., Dao, S.D., Zhang, W., Goh, M., Yu, G., Jin, Y., Liu, W., 2023. A new job priority rule for the NEH-based heuristic to minimize makespan in permutation flowshops. Engineering Optimization 55, 1296–1315.