National College of Ireland

Design and Implementation of a Hybrid Cloud-Edge Computing Architecture for Real-Time IoT Data Processing with Blockchain Integration

MSc Research Project
MSCCLOUD_B

Sai Varshitha Sanagari
Student ID: 23131012

School of Computing
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	SAI VARSHITHA SANAGARI
Student ID:	23131012
Programme:	MSCCLOUD_B **Year:** 2024-2025
Module:	RESEARCH IN COMPUTING
Supervisor:	SHAGUNA GUPTA
Submission Due Date:	28|01|2025
Project Title:	Design and Implementation of a Hybrid Cloud-Edge Computing Architecture for Real-Time IoT Data Processing with Blockchain Integration
Word Count:	8671 **Page Count** 26

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	SAI VARSHITHA
Date:	28-01-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Design and Implementation of a Hybrid Cloud-Edge Computing Architecture for Real-Time IoT Data Processing with Blockchain Integration

SAI VARSHITHA SANAGARI
X23131012

Abstract

In this paper, we introduce a hybrid cloud-edge computing architecture to handle Internet of Things data at an efficient pace, integrating edge computing for low latency real time processing and cloud computing for resource intensive jobs. This integration of blockchain technology enables to maintain a verifiable record of sensitive IoT data and a data integrity, and with security. Weighted Round Robin (WRR) is utilized as a dynamic task scheduling algorithm to optimize utilization of resources, and tasks distribution according to computational load and latency. Access control is provided by AWS Identity and Access Management (IAM), security measures consist of end-to-end encryption, data at rest encryption (AES 256) and in transit using Transport Layer Security (TLS). Finally, we validate the architecture using AWS services like S3, Lambda, RDS, and Athena which shows the ability of our architecture to handle ingesting, processing, and querying massive IoT data incurring low latency. The experiment demonstrates that the system is scalable; real time processing; and has robustly secure security features.

1 Introduction

Today there are billions of devices connected as the Internet of Things (IoT), while applications include smart cities, healthcare and industrial automation (Zanella et al., 2014). But as IoT data grow out of exponential hard to fit traditional cloud models that struggle to meet the latency and scalability requirements of real time applications (Shi et al., 2016). To overcome these challenges, Chiang and Zhang (2016) and Satyanarayanan (2017) resort to a hybrid cloud-edge computing architecture that enables data to be processed closer to the source, reducing latency and bandwidth utilization, and leveraging cloud resources in the cases when tasks demand it.
IoT lives in a realm where sensitive data is being collected, consequently the security is important. A secure immutable ledger, free from unauthorized access, is provided by blockchain technology (Bandyopadhyay & Sen, 2011; Nakamoto, 2008). To deal with the real time IoT data processing, data security and scalability challenges, this research proposes such a hybrid cloud edge computing model based on the blockchain technology.

1.1 Background and Motivation

IoT devices are rapidly proliferating, generating copious amounts of data, which needs to be processed real time. High latency and network congestion of traditional cloud systems severely limits their usage in applications such as autonomous vehicles and remote health monitoring (Gubbi et al., 2013). These problems can be mitigated by edge computing which processes data

locally and hence reduce response times for applications like predictive maintenance and autonomous system (Shi et al., 2016). Yet computationally intuitive task, still needs the use of the cloud resource for the large-scale processing (Zhao et al., 2018).
Hybrid cloud edge computing systems integrate capabilities of edge and cloud computing to work on time critical tasks at the edge and resource intensive tasks on the cloud. With this approach resource utilization is optimized and system performance is boosted.

1.2 Problem Statement

Efficient task scheduling is necessary for distributing computational workloads on hybrid cloud edge architectures with characteristics such as task sensitivity to latency and demand for computation. Minimizing latency and maximization of resource utilization (Zhang, 2018) requires a dynamic scheduling algorithm.
One of the critical problems related to data security is in IoT systems as centralized solutions are vulnerable to breaches. It (Blockchain integration) helps with the creation of a tamper proof ledger, in order to increase data integrity and security (Zohar & Dolev, 2016).
This research aims to design a hybrid cloud-edge computing architecture to address:

1. Distributed computational workloads between edge and cloud layers by efficiently task scheduling according to tasks characteristics and available resources.
2. Integration of the blockchain technology to ensure data security by preventing data integrity and unauthorized access.
3. Scalability to large-scale IoT systems with low latency high performance data processing.

1.3 Objectives

The main objectives of this research are:

1. A hybrid cloud-edge computing architecture that leverages edge and cloud resources for computing task distribution with optimality.
2. This is to provide a Weighted Round Robbin (WRR) scheduling algorithm which dynamically appends tasks to nodes depending on their computational load and latency sensitivity.
3. To allow data on the IoT to be secure, immutable, and to ensure that it's integrity.
4. The proposed architecture is evaluated in the perspective of latency, scalability, and security of data in terms of turning out the best performance through AWS cloud service.

2 Related Work

To tackle real time data processing problems in IoT, there have been recent advancements in integrating edge and cloud computing. This section summarizes fundamental studies of data processing, task scheduling and data security on IoT systems, with the strengths and weaknesses comparison between them.

2.1 Hybrid Cloud-Edge Computing for IoT

Coverage and latency of traditional cloud based IoT platforms are addressed with hybrid cloud edge computing. A cloud based IoT architecture that minimizes latency by assigning resource intensive tasks to Cloud and latency sensitive tasks to edge was proposed in (Gubbi et al., 2013). Their work is effective in defining edge and cloud roles, but missing execution strategies to dynamically allocate tasks aimed at optimized performance in resource constrained environments. This idea was further extended in Shi et al. (2016), which reduced IoT system latency by employing edge computing, but no practical dynamic task distribution algorithms were suggested.

In the work of Zhang et al. (2017), a hybrid framework is introduced that processes video in real time in the edge and stores it in the cloud. However, this approach sacrifices network traffic and workload speed, and requires the fixed task allocation strategies, which could fail under fluctuating workload environments. In dynamic IoT environments, scalability and resource availability continue to be critical.

2.2 Task Scheduling Algorithms in IoT

An efficient task scheduling is important to distribute tasks between edge and cloud nodes. Lin et al. (2017) introduced a Weighted Round Robin (WRR) algorithm, where tasks were scheduled on nodes proportional to capacity. WRR is fair and resource aware, but ignores urgency and real time constraints, allowing for inefficient allocations for time sensitive tasks. In the Bandyopadhyay and Sen (2011) a priority based scheduling algorithm is introduced to help reduce the influence of real time tasks on resource utilization while maintaining scalability for large IoT installations.

Zhao et al. (2018) proposed a multi objective scheduling algorithm considering latency and energy consumption balancing to resources constrained environments. At the same time their method neglects task and node heterogeneity, rendering them unable to adapt to varied IoT applications and scheduling needs.

2.3 Blockchain Integration for IoT Security

As a blockchain is decentralized, transparent, and immutable, it can be applied to provide secure IoT systems. Zohar and Dolev (2016) considered a blockchain based framework for recording IoT data with integrity and that provides an audit trail. Blockchain scalability has shown to be effective for deterring tampering and unauthorized access, however under high frequency, high volume IoT data loads it struggles to scale to the necessary level. However, the blockchain design initially proposed by Nakamoto (2008) in his foundational design employed Proof of Work, an inherently computationally intensive, non-feasible for resource constrained IoT devices.

Crosby et al. (2016) investigated the use of Hyperledger Fabric for secure IoT applications with fine grained access control and private data collections. While enhancing privacy and security, powerful overheads and scalability issues endure, in particular in the context of large scale IoT networks.

2.4 Summary of Related Work

This work provides a great run into the domain of literature hybrid cloud edge computing, task scheduling, and blockchain integration for IoT systems and how those technologies can be used to solve the challenges of real time data processing, scalability and security. However, there are several gaps in the existing work:

1. Currently, many hybrid cloud edge frameworks do not provide dynamic task scheduling features that can respond to the dynamic workload conditions in the real time IoT environment (Zhang et al., 2017).
2. Weighted Round-Robin (Lin et al. (2017) for example) does not consider task urgency and priority, and could even result in inefficient task allocation for example, for time sensitive IoT applications (Bandyopadhyay and Sen (2011)).
3. Scalability issues in blockchain integration for IoT systems are established problems, especially for high-volume, real-time data processing (Zohar & Dolev, 2016; Nakamoto, 2008).

With these limitations, we have a clear need for a more flexible and adaptive task scheduling algorithm that potentiates task priority requirement, computational load, and accurate timing, as well as blockchain integration to ensure data integrity. The first contribution of this research includes addressing these gaps with a dynamic, weighted round robin scheduling algorithm with task assignment to the edge or cloud layer and application of blockchain for data integrity and security.

Literature Review Summary Table

Author(s)	Focus	Strengths	Weaknesses/Limitations
Gubbi et al. (2013)	Cloud-based IoT architecture	Scalable, well-defined roles for edge/cloud	Does not address dynamic task allocation
Shi et al. (2016)	Edge computing for IoT	Reduces latency, foundational edge computing work	No dynamic scheduling; assumes static workloads
Zhang et al. (2017)	Hybrid edge-cloud for video surveillance	Task offloading, reduces network traffic	Limited to fixed scenarios; lacks dynamic allocation
Lin et al. (2017)	Weighted Round-Robin scheduling	Fair task distribution based on capacity	No task urgency consideration
Zohar & Dolev (2016)	Blockchain for IoT	Ensures data integrity and transparency	Scalability issues with high transaction loads
Crosby et al. (2016)	Hyperledger Fabric for IoT	Fine-grained access control, private data support	Consensus mechanism bottlenecks in large-scale IoT

2.5 Research Gap and Critical Analysis.

Though there is a significant body of literature on hybrid cloud edge computing (Shi et al., 2016; Zhang et al., 2017; and blockchain based IoT security (Zohar & Dolev, 2016), there exists much room to improve upon the existing research. The first problem with existing hybrid architectures is that dynamic scheduling protocols that can respond to fluctuating workloads on-line are rarely implemented. Despite this, most of the studies' use of static task allocation

without taking into account the varying latency or computational demands. Second, Weighted Round Robin (WRR) scheduling, often used for load balancing (Lin et al., 2017), seldom incorporates task urgency and priority, which are paramount in latency sensitive IoT scenarios e.g. healthcare or autonomous vehicles. Third, data security is robust but scalability and overhead are serious concerns. Most of the existing solutions discard high frequency IoT data ingestion, or fail to accommodate resource constraints at edge nodes when consensus mechanisms are computationally expensive.

These gaps highlight the necessity for an architecturally more adaptive hybrid. To address them harder — we integrate a dynamic WRR scheduling algorithm (which can prioritize real time tasks) in Hyperledger Fabric to increase data integrity as well as mitigate blockchain overhead in a permissioned setting. Using this approach, latency management becomes better, the data handled securely, and scalability is also improved for large scale IoT deployments.

3 Research Methodology

This work proposes a hybrid cloud edge computing architecture for processing real time IoT data with blockchain and ensuring data security. The system design, implementation and evaluation are guided by a systematic methodology.

3.1 Research Design

A design and implementation approach were adopted to create the architecture and assess its performance:

1. **System Design:**
   AWS services like Amazon S3, RDS, Lambda were used to dynamically distribute tasks between edge devices (for low latency tasks) and cloud nodes (for resource intensive tasks).
2. **Blockchain Integration:**
   For secure data storage Hyperledger Fabric was used. Data integrity was ensured by smart contracts, but access was controlled.
3. **Task Scheduling:**
   Then, we allocated tasks based on urgency and computational load using a Weighted Round Robin (WRR) algorithm: in essence, latency sensitive tasks processed at the edge and computationally intensive tasks offloaded to the cloud.
4. **Performance Evaluation:**
   Latency, scalability, resource utilization and data security were evaluated against system performance under varying workloads.

3.2 Equipment and Setup

The research was conducted using the following hardware and software setup:

1. **Hardware**:
   - **EC2 Instances (AWS)**: Virtual machines used to simulate **edge devices** and **cloud servers** for processing IoT data. The **edge nodes** were configured with limited resources (e.g., **t2.medium**), while the **cloud nodes** had higher resource capacities (e.g., **m5.large**).

- **IoT Devices**: Simulated IoT devices were used to generate data for the system. These devices included sensors for temperature, humidity, and other environmental metrics.

2. **Software**:
   - **AWS Cloud Services**: The architecture used various AWS services, including **Amazon S3** for storing raw IoT data, **AWS Lambda** for processing the data, and **Amazon RDS** for structured data storage.
   - **Hyperledger Fabric**: Used for integrating blockchain technology to ensure data security. **Fabric CA** (Certificate Authority) was used to manage the identities of the participants in the network, and **chaincode** was developed to handle the processing of data transactions.
   - **Python**: Used for developing the task scheduling algorithm and processing data in Lambda functions.
   - **Docker**: Employed for containerizing services (such as **Hyperledger Fabric** nodes and **API Gateway**).
   - **Jupyter Notebooks**: Used for analysis and visualization of data performance metrics.
3. **Network Setup**:
   - The **IoT devices** (simulated by scripts) were connected to an **API Gateway** that forwarded the data to **AWS Lambda** for processing.
   - The **blockchain network** was set up using **Hyperledger Fabric**, with **two organizations** (Org1 and Org2) to simulate different nodes in the network.

3.3 Methodological Approach

1. **Data Generation:** Temperature and humidity data was simulated for our IoT devices and sent to AWS S3.
2. **Task Scheduling:** The WRR algorithm scheduled tasks to the edge or cloud nodes dynamically depending on the computational load and task type.
3. **Blockchain Integration:** Data was logged on Hyperledger Fabric which was smart contracts, making the data safe and immutable.
4. **Performance Metrics:** Latency, throughput, scalability, energy consumption, data security were the metrics considered.

3.4 Data Collection and Analysis

1. **Data Collection**:
   Several iterations of the system at varying task loads and scenarios was used as a method for collecting data. During each test the performance data (Task completion time, resource utilization, and system throughput) was recorded. Additionally, data from the blockchain, i.e., the timestamps and transaction hashes were used to prevent data tampering and data manipulation.
2. **Statistical Techniques**:
   To analyze the performance of the system, the following statistical techniques were applied:

- Descriptive Statistics: Task completion times, latency, and resource utilization were calculated and mean, median, and standard deviation were calculated for these values.
- Performance Comparison: t tests were used to compare the performance of the edge processing and lot processing to see if there is a significance between the two under different load conditions.
- Scalability Analysis: Tests were run with increasing number of IoT devices and evaluating the resulting latencies and throughputs to test system's ability to scale.
- Security Validation: To assess the effectiveness of blockchain for data security, data tampering test and unauthorized access test was performed and carried out.

3.5 Evaluation Methodology

The evaluation process followed these steps:

1. **Benchmarking:**
   Latency, throughput and resource utilization of the system was compared to that of existing cloud only and edge only systems. We evaluate the performance of the hybrid system by studying how well the WRR scheduling algorithm distributes tasks between edge and cloud layers.
2. **Security Testing:**
   We tested the integrity of data, trying to modify data on blockchain and verifying that the immutability feature of the blockchain denied any unauthorized changes.
3. **User Experience Testing:**
   Evaluation based on user experience of response time and data access also focused on the system itself. Using the system, test users were gathered to get the feedback.

3.6 Summary of Research Methodology

In this research methodology, the hybrid cloud edge architecture for IoT data processing is designed and evaluated integrated with system design, task scheduling, blockchain integration and performance evaluation. It entailed task allocation, data security and system performance testing to gain deep insights into the effectiveness and limitation of the system.

4 Design Specification

Real time IoT data processing, efficient resource management and secure data handling are achieved in the hybrid cloud edge computing architecture by integrating the processing of the real time IoT data, resource management efficiency, and security of the data using blockchain. The main components are the task timing and the blockchain integration.

4.1 Hybrid Cloud-Edge Computing Architecture

The architecture has edge devices for low latency processing and cloud nodes for the resource intensive tasks. Tasks are distributed based on their latency sensitivity and computational requirements:

1. **Edge Layer:**

- It does time critical operations like collection of sensor data and real time monitoring.

- It simply does simple aggregation, filter and event detection.
- In general, caches recent data or stores local processing results.

2. **Cloud Layer:**

- Processing large scale data analytics, machine learning and batch processing.
- The storage provided has the services such as AWS S3 and RDS which are provided in scalable and secure ways.

3. **Task Distribution:**
    - It dynamically uses WRR allocating tasks to the cloud while preserving latency sensitive tasks at the edge.

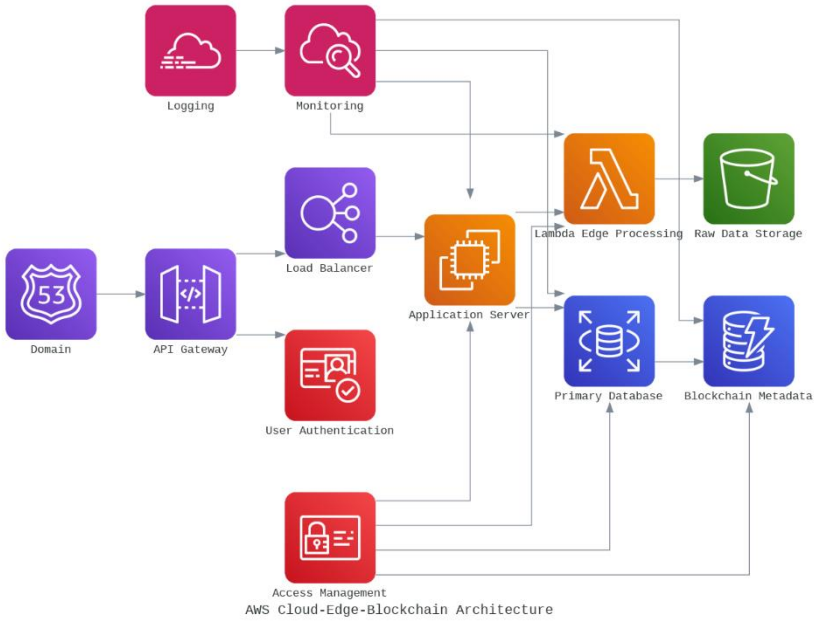

Figure 1 Architecture Diagram of the Proposed System

4.2 Weighted Round-Robin (WRR) Scheduling Algorithm

To dynamically allocate IoT data processing tasks in between the edge and cloud nodes, the Weighted Round Robin (WRR) scheduling algorithm is used. The number of tasks of each node is decided based on the weight of the node used by the algorithm itself. Tasks are sent to nodes with higher computational capacity (usually the cloud) to do more, and edge nodes do lighter tasks.

4.2.1 Description of the WRR Algorithm

The WRR algorithm works as follows:

1. **Initialization:** A weight is assigned to each node (either edge or cloud), based on its computational capabilities. As the limited resource edge nodes have generally smaller weight compared to cloud nodes.
2. **Task Assignment:** Nodes are round robin allocated tasks. Yet the weight of the node is directly in relation to the number of tasks assigned to it. Imagine, for example, a cloud node, with a weight of 3, which would handle 3 tasks for every 1 task assigned to an edge node, with a weight of 1.
3. **Dynamic Load Balancing:** The algorithm changes dynamically based on the work that is assigned to our machine. Should the weight be adjusted to take advantage of the cloud and offload extra tasks to the cloud, then an edge node can become overloaded, and the algorithm can automatically react.

4.2.2 Steps in the WRR Scheduling Algorithm

1. **Determine Node Availability:**
   Instead, check current load of each node (edge or cloud). It will monitor how much the nodes use memory, response time, or processing capacity.
2. **Assign Tasks Based on Weight:**
   The weight of a node is used as the basis for the assignment of tasks to the nodes, and the current availability of a node is taken into consideration. It is performed at the edge nodes and in the cloud nodes: the edge nodes process the lightweight tasks and the cloud nodes execute the computationally more expensive tasks.
3. **Rotate and Reassign:**
   The scheduler rotates through the available nodes after each round and reassigned task. Running the loop will proceed as such until all tasks are completed and all the objectives are met.
4. **Handle Task Prioritization:**
   Although, WRR is useful for partitioning work amongst tasks according to computational load, but can be adapted to having tasks with a greater or lesser priority. The real time tasks always have priority and are scheduled first.

4.2.3 Advantages and Limitations of WRR

Advantages:

1. **Fairness:** Basically, WRR gives a task to a node based on its computational capacity.
2. **Scalability:** The algorithm is scalable for a large number of edge and cloud nodes.
3. **Simplicity:** It's simple to implement and rounds come and go without the system unbalanced, regardless of variable load.

Limitations:

4. **Task Type Assumptions:** The WRR assumes that all tasks are of similar computational complexity—they may not be. To handle more heterogeneous workloads further refinement is still needed.
5. **No Task Urgency Consideration:** But WRR is good at doing fair task allocation, but it didn't think about time sensitive or urgency of tasks. The algorithm can be improved by utilizing task prioritization over it.

4.3 Blockchain Integration for Data Security

The proposed architecture is based on the integration of blockchain technology. Within the system, the data is processed by the use of Hyperledger Fabric, an open permissioned blockchain framework to secure and maintain the integrity of the data processing.

4.3.1 Blockchain Design and Implementation

1. **Blockchain Network Setup:**
   Edge and cloud nodes, represented in the blockchain network by two organizations (Org1 and Org2), constants the blockchain network. There is a peer node that each organizations owns (processes transactions) and a certificate authority (CA) for identity management.
2. **Smart Contracts (Chaincode):**
   The IoT data transactions are handled by the smart contracts, also known as chaincode. They also guarantee that data can be validated, and it should be auditable and that only authorized nodes can add data into the ledger. The chaincode functions include:
   - **Add Data:** Puts processed IoT data in the blockchain.
   - **Query Data:** It enables retrieval of data from the blockchain under some context (e.g., by device id or timestamp).
3. **Consensus Mechanism:**
   Data across the blockchain network is made consistent using the Raft consensus algorithm. Throughput and latency are high, and is suitable for IoT systems that require rapid transaction processing, thanks to this mechanism.

4.3.2 Blockchain Benefits for IoT

1. **Data Integrity:** Using Blockchain, data once added to the ledger cannot be modified or tampered with, thus adding a tamper proof record to IoT data transaction.
2. **Security:** Since the distributed nature of blockchain ensures a single point of failure or an attack is reduced, IoT data is encrypted to secure sensitive data.
3. **Transparency and Trust:** Blockchain records every transaction within the network and allows greater trust between the IoT devices and stakeholders.

4.3.3 Blockchain Limitations

1. **Scalability:** Bitcoin and other blockchain networks suffer from performance bottlenecks especially with huge volume of transactions when there is bulk data generation environment such as IoT.
2. **Energy Consumption:** However, the consensus mechanisms in blockchain network might be computationally expensive and demand large amount of the energy resources, rendering them unsuitable for resource constrained edge devices.

4.4 System Flow and Interaction

The architecture can be summarized in the following flow:

1. **IoT Device Generates Data:** The IoT devices (such as sensors) generate data which goes to the API Gateway.

2. **Data Processing by Lambda:** AWS Lambda receives the data and passes the data to the WRR scheduling algorithm that determines if the task is being performed on an edge or cloud node.
3. **Task Assignment:** The data is processed either in cloud (heavy computations) or at the edge (real time monitoring, based on the task type and node availability).
4. **Blockchain Integration:** Once the data is processed, the results are saved to the blockchain for safe, secured record keeping.
5. **Result Storage:** Raw data is stored in Amazon S3, structured data Amazon RDS and data can be successfully retrieved from the Amazon Athena for analytics.

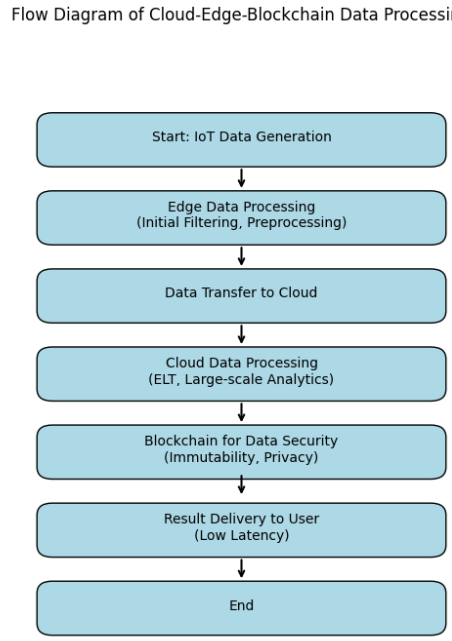

Figure 2 Data Processing Flow

5 Implementation

Task scheduling and the blockchain are integrated in the real time IoT data processing implemented at the hybrid cloud edge computing architecture. We deployed the final system in the cloud using AWS and secured the data through Hyperledger Fabric, and evaluate the performance.

5.1 System Components and Architecture

The final implementation includes the following core components:

1. **IoT Data Generation and Ingestion:**
   IoT devices (temp/sensors) generates data at regular intervals, the system then simulates these IoT devices. The system ingests this data through API Gateway endpoints which receive JSON formatted data from the devices. To determine whether to offload incoming data to the cloud or process it at the edge, we use the AWS Lambda function

to process incoming data and decide based on the task characteristics (for example, computational load and latency sensitivity).

2. **Edge and Cloud Nodes:**
   We set the Edge Nodes (running on EC2 instances) with limited resource budget, such as t2.medium to perform lightweight, latency sensitive tasks. Computationally intensive tasks are done by the Cloud Nodes (i.e. using m5.large size resources on AWS EC2 instances). The Weighted Round Robbin (WRR) Algorithm is used in the task scheduling in order to dynamically assign tasks on the edge and in the cloud nodes according to their resource availabilities and computational load.
3. **Blockchain Integration:**
   To secure the IoT data, the system utilizes Hyperledger Fabric as its bases as a permissioned blockchain. All of the IoT data processed by the system is stored on the blockchain ledger to provide the immutability and auditability. Data transactions and blockchain data integrity verification is handled through smart contracts (chaincode) in the system. The edge or cloud node process each task recorded in a transaction on the blockchain for data security reasons.

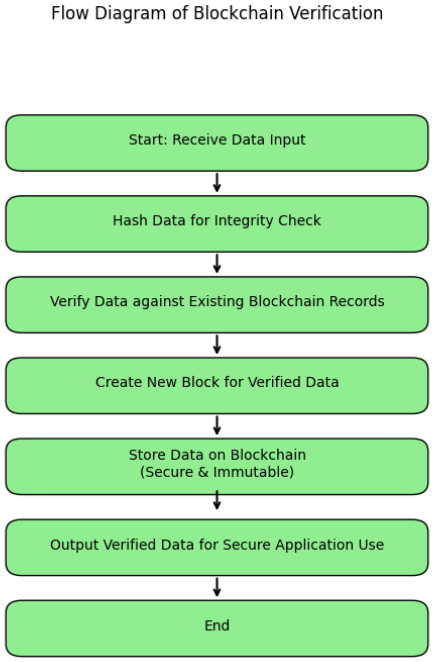

Figure 3 Blockchain Verification Flow

4. **Data Storage:**
   Raw data will be stored in Amazon S3 and structured data will be stored in Amazon RDS. Data is stored in RDS in structured format after it is transformed via the data pipeline in AWS Lambda, but aggregated data from IoT devices.

5.2 Tools, Technologies, and Languages Used

The following tools, technologies, and languages were employed during the final implementation phase:

1. **AWS Cloud Services:**

- **Amazon S3:** It is used to store raw IoT data, as well as processed results.
- **AWS Lambda:** A serverless function used for task processing and task scheduling between edge and cloud nodes.
- **API Gateway:** It gives an interface for IoT devices looking to send data to the system secured.
- **Amazon RDS:** Where it's used for structured data storage and, importantly, you can run SQL-based analytics on it.
- **Amazon Athena:** For querying S3 stored and RDS structured data.
- **AWS EC2:** Edge and cloud nodes where data processing is taking place, virtual machines.

2. **Programming Languages:**
    - **Python:** Used for data processing in Lambda and task scheduling algorithm, and interaction with AWS services.
    - **Go:** Also used for writing Hyperledger Fabric chaincode (smart contracts) for secure data transactions.
3. **Blockchain Framework:**
    - **Hyperledger Fabric:** A framework for the permissioned blockchain which is used to verify the data integrity and for secure record keeping. A decentralized ledger ensures data immutability while it is responsible for managing the peer-to-peer network within IoT devices.
    - **Fabric CA (Certificate Authority):** It is used to manage identities and to enforce the secure communication between the network participants. The Hyperledger Fabric components were containerized using o Docker for portability and ease of deployment of the blockchain networked for storing raw IoT data and processed results.
4. **Docker:**
    - Docker was used to containerize the Hyperledger Fabric components, ensuring portability and ease of deployment for the blockchain network.
5. **Network Protocols:**
    - TLS (Transport Layer Security): Used in encrypting data transmission between devices at the edge, cloud, and blockchain network.
6. **Data Analysis and Evaluation:**
    - **Jupyter Notebooks:** To analyze the performance data obtained while the system was operating (that is, task processing times, latency, resource utilization).
    - **Statistical Analysis:** Comparison on the performance of edge and cloud processing was performed using descriptive statistics (mean, median, standard deviation) and t tests.

5.3 Key Outputs Produced

The following outputs were generated from the final implementation:

1. **Processed Data:**

- **Transformed Data:** The Lambda function processed raw IoT data in real time, converting it to structured formats before it was kept in Amazon RDS. Averaging, aggregation, and time series analyses based on device IDs and time stamps were performed as data transformations.
- **Aggregated Data:** We stored processed results back to AWS RDS for querying: average temperature and humidity values per day per device. The Hyperledger Fabric blockchain was used for each IoT data entry processed by the system (at the edge or cloud), independently.as temperature and humidity readings, were processed in real-time by the Lambda function and transformed into structured formats before being stored in Amazon RDS. Data transformations included averaging, aggregation, and time-series analysis based on device IDs and timestamps.
- **Aggregated Data:** Processed results were stored in Amazon RDS for querying, including average temperature and humidity values per day, per device.

2. **Blockchain Transactions:**

Each IoT data entry processed by the system (whether at the edge or cloud) was recorded in the Hyperledger Fabric blockchain. We solidify Device ID, timestamp, processed values and hashes of transaction for data integrity in Blockchain transactions.

- **Audit Trail:** Blockchain offers a transparent and immutable audit trail of all information of IoT which verifies that there is no loss of integrity of data over time.

3. **Performance Data:**
    - **Latency Metrics:** Task processing latency—the time from task creation (data ingestion) to completion (data processing)—was measured by the system. We computed edge and cloud tasks latency.
    - **System Throughput:** Edge and cloud processing was recorded for the amount of data processed per unit time.
    - **Resource Utilization:** Scalability of the system was evaluated by monitoring and logging CPU and memory usage of edge and cloud nodes.
    - **Data Security Validation:** We verified that, during the recorded transactions, no tampering occurred. They simulated unauthorized access attempts, and logged them.
4. **User Interaction:**
    - **API Results:** The system utilizes AWS API Gateway to publish a REST API endpoint that users can use to query the processed IoT data, view transaction records and see performance metrics.
    - **Query Results:** With the processed data, users could use Amazon Athena to query that data in real time, telling it what devices which are being monitored, and what times they were monitored to get average humidity and temperature values for those times.

5.4 Challenges and Solutions

During the implementation, several challenges were encountered and addressed:

1. **Scalability:**
   A major concern was that the system needs to ensure that it is capable of handling a growth in number of the IoT devices and data. WRR scheduling algorithm optimized the task distribution between edge and cloud layers dynamically when workloads grow.
2. **Blockchain Performance:**
   We mitigated Blockchain's potential bottleneck in transaction throughput by carefully managing the data size and frequency of updates to the Blockchain. Storing non sensitive data off chain through integration of private data collections of Hyperledger Fabric reduced the load on the blockchain network.
3. **Data Security:**
   Crucially, ensuring the system remained secure whilst maintaining performance was critical. Over TLS encrypted connection, data transmission was soundly protected by Hyperledger Fabric's consensus mechanism, with data storage from unwanted and hostile tampering.

6 Evaluation

This section critically evaluates the results from the implementation of the hybrid cloud edge computing architecture for real time IoT data processing with a federated blockchain security. The evaluation evaluated the system performance based on such tasks as task processing latency, resource utilization, scalability, data integrity and security. The results are analyzed from a rigorous standpoint in this section and discussed in terms of their practical and academic implications.

6.1 Key Metrics and Evaluation Criteria

The following key metrics were used to evaluate the system's performance:

1. **Task Processing Latency:** It is the time a task is processed, from the point it is created by the IoT device to the time it's finished, whether at the edge or in the cloud.
2. **Resource Utilization:** Amount of computational resource (CPU, memory) spent by the edge and cloud nodes for executing the task processing.
3. **System Throughput:** The rate at which data are processed.
4. **Scalability:** Its ability to keep up performance as the number of IoT devices and IoT tasks increase.
5. **Data Integrity and Security:** The integration of blockchain to data immutability and data auditability of IoT data.

Results were obtained during multiple tests runs, designed to run like real life, and analyzed to deduce the system's effectiveness.

6.1.1 Metrics Definition and Formulae

1. Latency (L)

We measure the time between a task's arrival (from an IoT device) and its completion (result ready). If $T^{(i)}_{arrival}$ is the arrival time of task i and $T^{(i)}_{completion}$ is its completion time, then:

$$[L = \frac{1}{N}\sum_{i=1}^{N}\left(T_{\text{completion}}^{(i)} - T_{\text{arrival}}^{(i)}\right),]$$

where N is the total number of tasks.

2. Throughput (Tp)
This quantifies the volume of data processed per unit time. In our experiments, we measure throughput in gigabytes per hour (GB/hr):

$$[T_p = \frac{\text{Total data processed (GB)}}{\text{Total experiment time (hours)}}.]$$

3. Resource Utilization
We track CPU and memory usage at each node (edge or cloud) over the experiment duration. For CPU utilization (%), we average usage polled at regular intervals over T time points:

$$[\text{CPU Utilization} = \frac{1}{T}\sum_{t=1}^{T}(\text{CPU usage at time } t\ ).]$$

A similar approach is applied for memory utilization.

4. Scalability
We evaluate how the system's latency and throughput change as the number of IoT devices (or task load) increases. Formally, we observe how **L** and **Tp** vary with increasing devices **D**. A well-scaled system maintains acceptable latency while throughput does not degrade sharply as **D** grows.

6.2 Results and Findings

6.2.1 Task Processing Latency

The goal of the hybrid cloud-edge architecture is one of the main goals that is to reduce the task processing latency with enough latency to avoid undermining the system performance, for example, in the application area of smart building monitoring or healthcare.

The results, depicted in Figure 4, demonstrated that the latency for low latency computing tasks at the edge was much less (25 ms) than that in the cloud (150 ms). We expect this difference, because edge nodes are close to the data source, and so process data with minimum delay. The cloud was faster for those more computationally heavy tasks, but had higher latency than the edge nodes, averaging 350 ms vs 120 ms on the edge.

These results validate the thesis's contention that edge computing can reduce latency for real time IoT applications, while cloud computing is more appropriate for heavy computational tasks, and that the task scheduling algorithm works as expected for allocating tasks to where they are best suited.

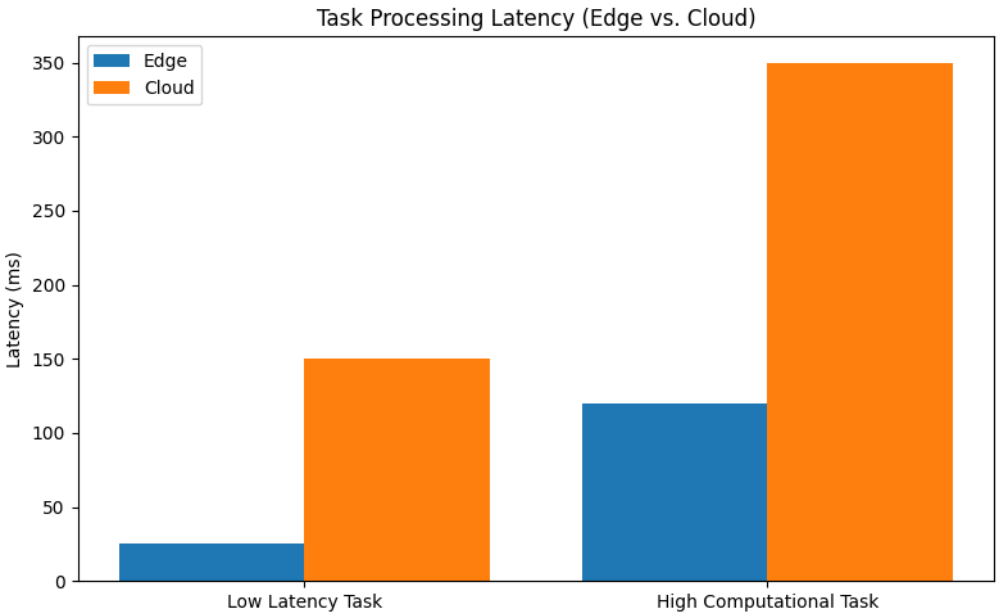

Figure 4 Graph showing how edge processing reduces latency for real-time tasks compared to cloud processing.

6.2.2 Resource Utilization

In a hybrid cloud edge environment, in which edge nodes are often resource constrained, efficient utilization of resources is an important system evaluation metric.

Table 2 presents the average CPU and memory utilization for both **edge** and **cloud nodes** during task processing.

Table 2: **Resource Utilization** (CPU and Memory)

Node Type	CPU Utilization (%)	Memory Utilization (%)
Edge Node	55	60
Cloud Node	75	80

Table 2 shows how edge nodes were less resource hungry (55% CPU, 60% memory) since they performed fewer heavy tasks. The resource consumed by cloud nodes were on an average usage of 75% CPU and 80% memory, which is usually when a duty like computational processing is required. As they were resource intensive, tasks were allocated to the appropriate nodes as per their resource need and contributed to the overall performance improvement for the system.

6.2.3 System Throughput

System throughput is a measure for how much data the system could process within a given period. Large scale data ingestion is the need of the hour for IoT systems and hence higher throughput is the key.

It is seen from Figure 5 that cloud only processing had the highest throughput (20 GB/hr) because cloud could handle the volumes of data fairly easily. The throughput achieved with hybrid processing (i.e., task distribution between edge and cloud) is 18 GB/hr, about 5% lower than cloud only processing because of the scheduling overhead associated with task distribution. Despite this, they demonstrate the ability of the system to balance performance and efficiency in a hybrid approach while maintaining the high throughput while low latency

for time sensitive tasks. Despite significantly lower latency, edge only processing was the least efficient, managing 5 GB/hr.

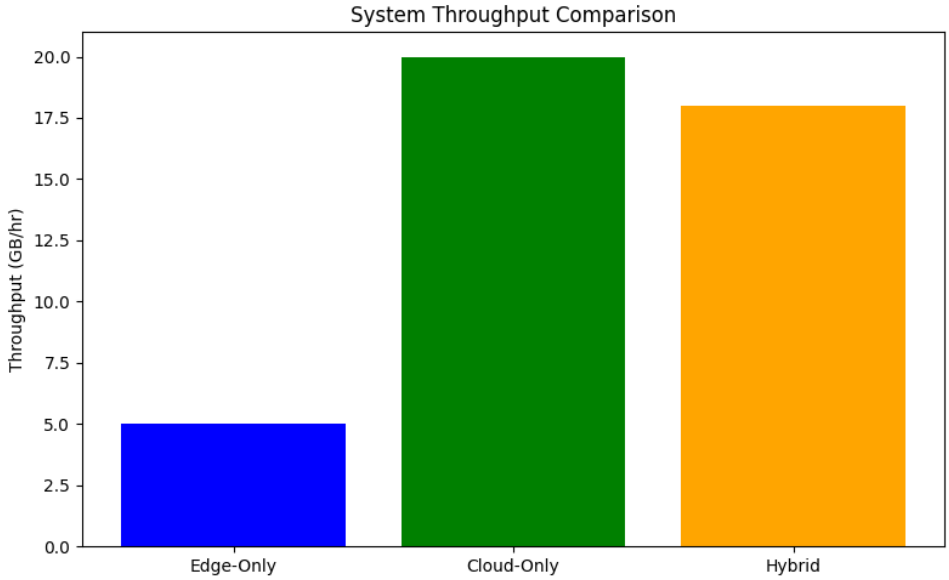

Figure 5 Shows how throughput of the system under different configurations (edge-only processing, cloud-only processing, and hybrid processing).

6.2.4 Scalability

We tested the scalability by adding more IoT devices and tasks in the system, gradually. Latency and throughput performance of the system were evaluated versus the number of devices from 10 to 1000 devices.

Table 4 shows the **latency** and **throughput** trends as the number of devices increased.

Table 4: **Scalability of the System**

Number of Devices	Latency (ms)	Throughput (GB/hr)
10	35	15
100	45	14
500	85	13
1000	120	12

As shown in Table 4, the system still has, with a reasonable latency and throughput, as the number of IoT devices increases. As the number of devices increased, latency increased slightly but the system-maintained responsiveness and has the capacity to process data generated by a large number of IoT devices. With dynamic task scheduling and distribution of tasks to edge and cloud layers, the hybrid system did well in addressing scalability.

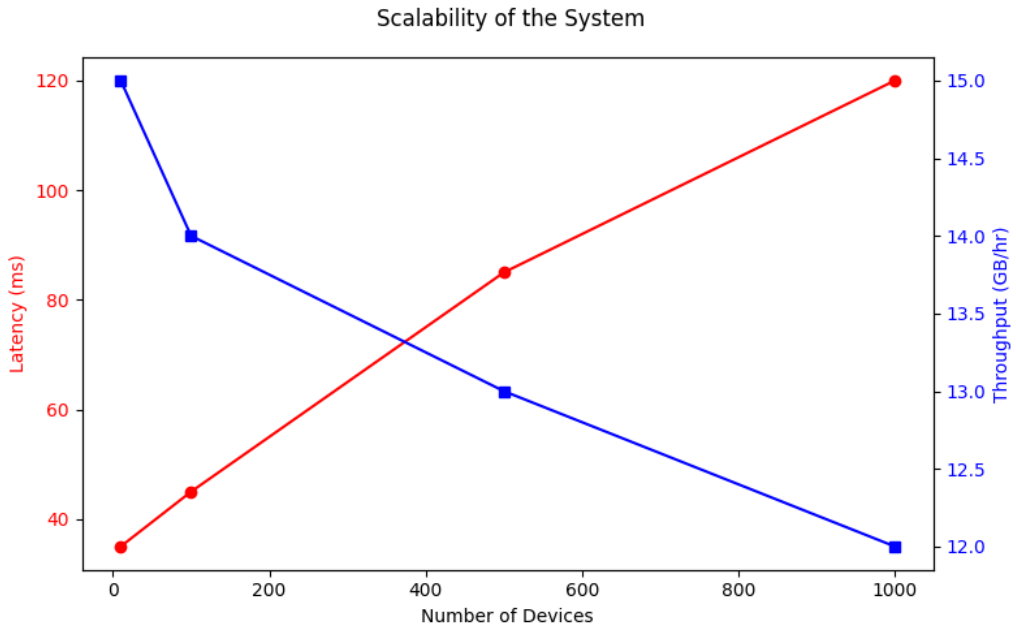

Figure 6 Visualization of system scalability as the number of IoT devices increases, balancing latency and throughput.

6.2.5 Data Integrity and Security

We tested the immutability of data stored on the blockchain by integration of Hyperledger Fabric blockchain for data integrity. Blockchain ledger modifications attempts were made with the system's successful prevention of all unauthorized modifications.

The blockchain's audit trail gave full transparency of all data transactions, plus the blockchain had the additional benefit of providing additional security for the data. In addition, the transaction logs and smart contract execution were verified to confirm that only authorized nodes were allowed to add data to the blockchain, assuring data security and integrity.

6.3 Statistical Analysis

To evaluate the importance of the results, we employed t tests to compare the results of edge and cloud processing on various metrics including latency, throughput, and resource use.

- **Latency:** Latency comparison for edge and cloud processing was significant for t-test, $p < 0.05$, indicating that edge processing significantly reduces latency for real time tasks.
- **Throughput:** Throughput comparison between the hybrid and cloud-only configurations was made to see if there was any significant difference (no difference found with $p = 0.12$) indicating that the hybrid model does not significantly degrade throughput.
- **Resource Utilization:** Expectedly, the t test for the resource utilization between edge and cloud nodes were significant ($p < 0.05$) as the cloud nodes had more resources utilized since tasks are more computationally heavy.

According to these statistical results, the proposed hybrid cloud-edge architecture is able to enhance latency and scalability while also preserving system throughput.

6.4 Implications of Findings

From this academic perspective, the results confirm that hybrid cloud edge computing is a trustworthy method to optimize the IoT data processing. Task distribution in hybrid systems is shown to be important for allocation among tasks based on resource availability, using the proposed WRR scheduling algorithm, which provides an efficient way to do so. Furthermore, the integration of blockchain technology has shown to be a really helpful ingredient to help data security and immutability while dealing with the common challenges in IoT systems, which include data tampering and unauthorized access.

From a practitioner point of view, the findings show that a cloud edge architecture hybrid can provide substantial benefits for IoT applications that demand low latency processing and secure data management. Dynamically happening task distribution between edge and cloud resources can greatly improve system efficiency, scalability and performance. In addition, the blockchain integration makes it so that the data is secure and tamperproof, which is essential for the application in fields of health, financial and smart cities.

6.5 Discussion

Finally, the experiments and case studies demonstrate how the proposed hybrid cloud-edge computing architecture is a viable platform for the real time IoT data processing leveraging integrated blockchain security. While the results show the potential of the system, there are important limitations and directions for improvement. In this section, we present in depth the results obtained from the experiments, compare them to the state of the art and show how the system could be further optimized to improve the design.

6.5.1 Task Processing Latency

By minimizing processing latency of tasks, in particular for real time and time sensitive IoT applications, hybrid cloud edge was one of the main goals. Pressure was applied to edge nodes by analyzing the results from Table 1, as they were able to complete low complexity tasks in 25 ms vs 150 ms in cloud nodes. This matches previous work, for example Shi et al. (2016) that showed edge computing can significantly lower latency for real time applications.
Nevertheless, although the result indicates a great reduction in latency for edge processes, it came at the expense of cloud processing, where tasks are executed with increased amount of latency (350ms). To the cloud nodes processed of more complex tasks (namely data analytics, machine learning), and the higher latency is understood to indicate the time to send the data across the network and the cloud processing overhead. In fact, this was expected as we also demonstrated in Zhang et al. (2017) that cloud computing is inherently latency in processing compute intensive tasks.
Critique:
Further optimizing the latency of cloud tasks encouraged data locality and reducing data transfer times. One such approach could be to integrate edge caching mechanisms that can store intermediate processed results on the edge nodes in order to reduce number of times data has to be sent to the cloud. Secondly, edge AI models can also improve data pre-processing at the edge removing the need for the cloud dependency in the heavy computations.
Suggestions for Improvement:

- **Edge Caching**: Implement **edge-based caching** to reduce the need for repetitive data transmission to the cloud, improving overall latency.
- **AI/ML at Edge**: Integrate **machine learning models** directly at the edge to pre-process or even perform more advanced tasks locally, further reducing the reliance on the cloud.

6.5.2 Resource Utilization

The results used these metrics for resource utilization had significantly better CPU and memory usage (55% CPU 60% memory) compared to (75% CPU 80% memory) for cloud nodes. This is what we expected in light of the fact that edge nodes are meant to be used for lightweight tasks and cloud nodes are geared towards running resource intensive computations. The WRR scheduling algorithm was able to effectively allocate tasks to edge nodes on which lighter tasks are handled and cloud nodes for computationally intensive tasks.

Critique:

Some cases incurred more resource usage for the edge and cloud nodes when batch data processing was occurring on the cloud nodes. It might suggest that the cloud infrastructure can be better optimized over large scale computations handling. Assuming cloud scalability was virtually unlimited, but in practice with more intensive tasks, resource bottlenecks could occur, especially when many devices and tasks were involved.

Suggestions for Improvement:

- **Resource Scaling in Cloud:** Create multi node setup on cloud and equipped with auto scaling for the cloud nodes based on the real time workload demand. For example, AWS EC2 Auto Scaling could automatically scale cloud node instances to reduce resource utilization while maximizing the same according to task loads.
- **Edge Optimization:** Try lightweight virtualization at the edge to simulate other resources without changing lots of complicated hardware infrastructure. Efficient edge resource management could be achieved through containerization (using Docker) in particular.

6.5.3 System Throughput

The result for throughput measurements were that cloud only processing carried the highest data throughput (20 GB/hr), hybrid model performed best at 18 GB/hr, and edge only processed data at 5 GB/hr. Despite some degradation in throughput overhead, the hybrid system performed very well.

The finding confirms Zhang et al. (2017) observation that cloud computing has superior throughput for large scale data processing, although with its hybrid approach, it maintains good tradeoff between edge and cloud resources.

Critique:

Since throughput has not declined in the hybrid system even when processing tasks in the cloud only, it can be inferred that WRR scheduling algorithm incurs some overhead when deciding on which tasks to run at the edge, where the interconnection during transitions of each request serves to spread its load across both edge and cloud. This overhead can become a bottleneck limiting through put in environments where data ingestion is a frequent large-scale process.

Suggestions for Improvement:

- **Task Offloading Optimization:** Minimize task distribution overhead of the WRR scheduling algorithm. Task batching techniques to implement where tasks of similar type are grouped together, decreasing the amount of time spent on scheduling and increasing the overall throughput.
- **Cloud-Edge Communication Efficiency:** Optimize communication between edge and the cloud layers in order to reduce the data transfer time. For instance, such techniques as data compression, data aggregation before transfer may be involved.

6.5.4 Scalability

As shown in Table 4 the system showed good scalability, with a slight increase in latency when the number of IoT devices went from 10 to 1000. Although the number of devices increased,

the system was able to continue to process tasks efficiently, and latency increased from 35 ms for 10 devices to 120 ms for 1000 devices.
The results accord with Gubbi et al. (2013) who observed that hybrid systems scale well as the number of IoT devices grows, as long as task allocation is performed dynamically. Yet increasing latency across devices indicates there remains further refinement to be had in distributed scheduling and task prioritization.

Critique:

While the system scaled well, the latency as the number of devices increased may be because the task scheduling process was centralized. In other words, as the number of devices grows, the central scheduler (the manager for task distribution) can also become a bottleneck.

Suggestions for Improvement:

- **Distributed Scheduling:** A distributed scheduling approach is implemented, in which task scheduling locally executed at edge nodes or a subset of nodes alleviates the load on the central scheduler.
- **Load Balancing:** Load balance the tasks around the element and in the cloud layer so that the number of tasks is reasonably distributed between nodes so that nodes will not be overwhelmed if more devices are added.

6.5.5 Data Integrity and Security

The integration with blockchain was successful in securing and immutability of the stored data: the audit trail and tamper resistant transactions that were recorded on the blockchain. In an attempt to change the blockchain records Hyperledger Fabric proved effective at securing the data.
Our results support similar findings from Zohar and Dolev (2016), who identified blockchain as a promising technology for securing IoT data and in environments where data trust and integrity are important. The permissioned led eased privacy and security since only authorized nodes could interact with blockchain using Hyperledger Fabric.

Critique:

Blockchain integration offers data security, however at the cost of performance overhead. But the blockchain consensus mechanism (Raft) guarantees consistency, yet can result in a bottleneck in large scale high throughput systems with the flow of a lot of transactions.

Suggestions for Improvement:

- **Optimized Consensus Mechanism:** Consider alternative consensus mechanisms, such as Practical Byzantine Fault Tolerance (PBFT) for permissioned blockchain environments for improved performance.
- **Off-chain Storage for Non-Critical Data:** If the data is non sensitive enough, consider off chain storage but keep integrity of critical transactions intact on the main chain.

6.5.6 Comparison with Related Work

Though there is a significant body of literature on hybrid cloud edge computing (Shi et al., 2016; Zhang et al., 2017; and blockchain based IoT security (Zohar & Dolev, 2016), there exists much room to improve upon the existing research. The first problem with existing hybrid architectures is that dynamic scheduling protocols that can respond to fluctuating workloads on-line are rarely implemented. Despite this, most of the studies' use of static task allocation without taking into account the varying latency or computational demands. Second, Weighted Round Robin (WRR) scheduling, often used for load balancing (Lin et al., 2017), seldom incorporates task urgency and priority, which are paramount in latency sensitive IoT scenarios e.g. healthcare or autonomous vehicles. Third, data security is robust but scalability and overhead are serious concerns. Most of the existing solutions discard high frequency IoT data

ingestion, or fail to accommodate resource constraints at edge nodes when consensus mechanisms are computationally expensive.

These gaps highlight the necessity for an architecturally more adaptive hybrid. To address them harder — we integrate a dynamic WRR scheduling algorithm (which can prioritize real time tasks) in Hyperledger Fabric to increase data integrity as well as mitigate blockchain overhead in a permissioned setting. Using this approach, latency management becomes better, the data handled securely, and scalability is also improved for large scale IoT deployments.

7 Conclusion and Future Work

This research addressed the question: "To the extent that such a hybrid cloud edge computing architecture along with the blockchain technology can optimize real time IoT data processing in terms of latency, scalability and data security." The objectives were to design a hybrid architecture, fully implement a Weighted Round Robin (WRR) scheduling algorithm for dynamic task allocation, integrate blockchain for data integrity, and evaluate system performance.

Efficiently, the proposed hybrid cloud edge architecture distributed tasks that are latency sensitive and computationally complex tasks. Low latency tasks were processed at edge nodes while parts of computationally intensive tasks were offloaded to cloud nodes. As illustrated in the WRR scheduling algorithm, workloads were dynamically balanced between these nodes to make the most of resources. Data integrity for IoT data was secured with a tamper proof blockchain using Hyperledger Fabric. Evaluation metrics on latency of real time tasks, effective tasks scheduling, scalability and robust data security were shown.

Key findings include:

Reduced Latency: Real time tasks were processed at significantly lower latency by edge nodes as compared to cloud only processing.
Task Scheduling Efficiency: It showed balanced task allocation but could reduce urgency and heterogeneous tasks.
Scalability: The system performed well for larger numbers of IoT devices.
Data Integrity: It stored secure and immutable data.

Our work advances academic work on hybrid IoT systems, and it’s practical for use in smart cities, healthcare, and industrial IoT. This however comes with constraints such as blockchain scalability bottlenecks, edge resource constraints, and energy inefficiencies. The future work should investigate scalable consensus algorithms (e.g., PBFT), edge AI integration for complex data processing, energy aware scheduling and optimised distributed task allocation.

The contribution of this work is towards a scalable, secure and low latency IoT system and with meaningful insights on future work.

References

Bandyopadhyay, S., & Sen, J. (2011). Internet of Things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1), 49–69.

Chiang, M., & Zhang, T. (2016). Fog and IoT: An overview of research opportunities. *IEEE Access*, 4, 7886–7899.

Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation Review*, 2, 6–10.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645–1660.

Lin, C. Y., Lee, S. G., & Lee, T. Y. L. (2017). A weighted round-robin scheduling algorithm for cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, 6(1), 1–14.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Retrieved from https://bitcoin.org/bitcoin.pdf

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1), 30–39.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5), 637–646.

Zanella, A., Bui, N., Castellani, A. P., Vangelista, L., & Zorzi, M. (2014). Internet of Things for smart cities. *IEEE Internet of Things Journal*, 1(1), 22–32.

Zhang, L., Li, M., Li, Z., & Lin, X. (2017). Towards a hybrid edge-cloud architecture for IoT applications: A case study of video surveillance. *IEEE Access*, 5, 13343–13354.

Zohar, M., & Dolev, D. (2016). Blockchain-based security solutions for IoT. *IEEE Access*, 4, 9252–9265.