Received: 19 Oct 2024 | Revised: 18 Dec 2024 | Accepted: 21 Dec 2024 | Published: 30 Dec 2024
Volume 2(2) Pages 107-122
DOI: https://doi.org/10.56279/jicts.v2i2.125

JICTS
Journal of ICT Systems

Fog-Based Computing Architecture for Enhancing Secondary Distribution Grid Services Management

Gilbert M. Gilbert [a,1], Shililiandumi Naiman[b],

[a]*Department of Information Systems and Technology, The University of Dodoma, Dodoma, Tanzania*
[b]*Department of Electronics and Telecommunications Engineering, University of Dar es Salaam, Dar es Salaam, Tanzania*

[1]Corresponding author
Email: gilbert.gilbert@udom.ac.tz

Funding information

This work received partial funds from the University of Dar es Salaam, Tanzania.

Abstract

In monitoring the health of secondary distribution networks (SDNs), power utility providers have faced an increasing need to deploy intelligent solutions with affordable sensing and data-driven technologies. Existing manual-based approaches are not capable of collecting large volumes of real-time operational data to achieve significant monitoring of the SDN network for reliable power distribution. Effective monitoring would require real-time sensing, scalable high-performance computing, and appropriate grid-based applications designed for efficient data processing. This paper presents a computing architecture for grid services monitoring to enhance real-time fault management in SDN. The architecture leverages wireless sensor networks, a hybrid cloud-fog computing architecture, and a heuristic-based application coordination mechanism to efficiently manage grid applications. Experimental results indicate that coordination mechanism improved workload distribution by up to 70% in fog nodes and to 40% in the cloud. A fog-based architecture provided low latency improvements of 70% compared with that of cloud-only architectures. This signifies that most of the data processing was pushed to the local fog nodes, which is crucial for distributed fault management applications.

Keywords

Fog Computing
Edge Node
Computing Architecture
Secondary Distribution
Fault Management

1. Introduction

The current electric power grid systems face growing economic and environmental concerns, prompting the need for more efficient and sustainable solutions [1]. Smart grid architectures have emerged as a potential replacement, leveraging Information and Communication Technologies (ICT) to offer a range of intelligent services. These services include the integration of renewable energy sources, load monitoring and control, and management of power generation and consumption [2]. However, the complexity of smart grid systems arises from the multitude of devices and applications that are interconnected through two-way communication networks, posing significant challenges for effective control and management [3].

As smart grid technologies are gradually integrated into distribution networks, which represent the largest segment of the entire power system, the management of both soft and hard assets becomes increasingly difficult [4, 5]. These challenges are exacerbated by the vast geographical coverage and size of the networks. The architecture of a smart grid demands an extensive deployment of sensors and Internet of Things (IoT) devices that generate massive volumes of data. This large-scale deployment of IoT devices and sensors leads to the typical "big data" issues of volume, velocity, and variety. Additionally, these data streams introduce requirements for accuracy, security, Quality of Service (QoS), and user expectations, while ensuring operational cost efficiency. The generated data are essential for effective grid services and applications management, but they also demand robust computing architectures for data processing, analysis, and storage.

Cloud computing has become the de facto standard for supporting IoT-enabled smart grid solutions, providing scalability, reliable data analysis, and operational efficiency. However, centralizing cloud facilities geographically necessitates multi-hop transmission of data collected from sensors to the cloud systems for processing. This setup negatively impacts latency-sensitive applications, network bandwidth, and data locality, leading to delays, security issues, and inefficiencies[6, 7].

In response to these challenges, fog (edge) computing has emerged as an effective platform that bridges the gap between cloud and IoT [8]. By leveraging lightweight and customizable computing resources located closer to the data source, fog computing reduces the need for multi-hop data communication. Typical edge computing devices, such as switches, routers, and low-profile computers are integrated with the necessary computational infrastructure and management models to support local data processing. This architecture not only reduces latency but also enhances service flexibility, security and distributing resource demands more effectively. Although fog computing nodes have limited computational power, they are customizable to address specific application needs. Additionally, fog computing enhances data privacy by enabling local storage of sensitive information instead of transmitting them to centralized data centers.

Despite its benefits, fog computing faces challenges in ensuring seamless interoperability and coordination between cloud and fog services [9]. There is a need for techniques that facilitate the migration of secondary distribution network (SDN) grid services and applications between cloud and fog nodes without compromising system performance, while also addressing the inherent architectural differences. The focus is on identifying a design capable of supporting seamless application migration while maintaining the required performance levels. To achieve this, monitoring the resource usage of each grid application/service and identifying appropriate triggers for migration decisions are essential. This will ensure computing resources between the fog-cloud continuum are well utilized.

This paper introduces a design for grid applications/services monitoring platform that leverages fog computing to address the limitations of cloud-only architectures in managing faults in SDNs. The proposed architecture integrates wireless sensor networks fog-based computing and an application coordination algorithm to support real-time monitoring and management of grid services, particularly for fault management. A proof-of-concept prototype (Figure 3) is developed to demonstrate the platform’s effectiveness, focusing on workload distribution, latency reduction, and improved fault detection in SDNs.

2. Literature Review

A fault in an electric power system is an abnormal condition that can be produced by a variety of factors, such as weather, human error, fire, and hardware problems. Faults can include open circuits and short circuits, among other things. Short circuits are the most common problems in distribution networks, and they can be diagnosed by analyzing phase currents [10]. Furthermore, depending on whether the faults occur on distribution lines or transformer substations, they can be classed as overcurrent, over/undervoltage, over-temperature, overloading, low oil level, or earth faults. Based on the severity of the fault and the duration of the problem, it may lead to poor power quality, unreliable supply, reduced consumer comfort, and potential equipment damage or safety hazards.

Fault detection automation in electrical power systems is one of the most effective technologies for reducing outage times. There are significant works that address fault detection and clearance in transmission and primary distribution networks using centralized systems, such as Supevisory Control and Data Acquisition (SCADA) and Distributed Management Systems (DMS) [11]. However, because of the complexity and pervasiveness of SDN, more research leveraging distributed architecture is required to overcome network fault clearance issues. Management of faults in SDNs, including the Tanzanian network, relies on customer call information or virtual inspections for fault identification, which leads to longer duration for resolving problems [12].

A work by Gilbert et al. [6] and Gooi et al. [13] have thoroughly reviewed the application potentials of using edge computing and intelligence in power systems. A study by Mei et al. [14] discusses edge-cloud collaboration for fast and accurate fault detection in Low Voltage (LV) distribution networks using deep learning techniques. It does not detail much about the architecture design. Similarly, a study done by Sodin et al. [15] demonstrated the use of a hybrid edge-cloud approach in fault management by successfully utilizing Phasor Measurement Unit (PMU) and Long-Term Evolution (LTE) technologies. However, it did not deal with grid application/service coordination for resource management. Further, Huo et al. [16] looked into proposing wavelet transform applications in fault detection based on edge computing. Issues of application/services management were not discussed. A study by Netsanet et al. [17] proposes a cognitive edge computing-based fault detection strategy using support vector machines (SVM) for detection and long short-term memory (LSTM) models for fault localization. It leverages a distributed architecture to improve real-time detection accuracy and reduce fault-clearing times in active distribution networks. This work addresses challenges related to adaptive settings, high impedance faults, and complex grid conditions without relying heavily on centralized systems. The study heavily relies on machine learning models, which require significant training data and computational resources for real-world applications. It does not address grid service coordination, limiting scalability for large SDNs. Fog nodes may suffer from resource constraints during large-scale deployments in complex environments.

A study by Alhanaf et al.[18] introduces fault detection methods leveraging artificial neural networks and one-dimensional convolutional neural networks for smart grids. It achieves impressive accuracy in detecting, classifying, and locating faults using sensor data, such as voltage and current signals. However, it has limited applicability to complex grid environments, including SDNs.

Smart grid services involve several applications that take advantage of the resources in cloud and fog systems. Since these applications have differing computing needs and priorities, there is a need for mechanisms to coordinate their processing. A comprehensive review by Li et al. [19] provides the status of edge-cloud systems in smart grids, detailing architectures and applications, including fault management. It highlights challenges, such as latency, dynamic resource allocation, and integration complexity. This study focuses heavily on theoretical architecture without practical case studies or real-world validations. A work by Santoro et al. [20] considered a platform for workload orchestration in a fog computing environment by offering the functionality of negotiation, scheduling, and workload placement, taking into account traditional requirements (e.g., RAM, CPU, Storage size). Li et al. used neural networks to orchestrate resources between cloud and fog systems for smart grid fault detection [21]. These solutions did not consider environmental complexities, such as those found in SDNs.

There are various application resource monitoring systems. For instance, Nagios[1], Datadog[2], Prometheus[3] and Dargos [22] are used for monitoring applications, services, operating systems, and network protocols across several platforms. Choosing the right tool depends on individual project needs, hence a lot of customization is needed.

There are still issues related to limited practical SDN fault management solutions, grid services coordination for resource management, and complexities of SDN. This work seeks to contribute to enhancing fault detection accuracy, improving system reliability, and ensuring efficient orchestration of grid services in complex SDNs.

3. Cloud-Fog Application Monitoring Architecture

To efficiently manage, monitor, and optimize applications deployed in distributed computing in the SDN environment that spans both cloud and fog layers, there is a need for an architecture to coordinate computing resource usage.

The process of coordinating applications relies on the control loop that watches fog application events and then reacts accordingly to the observed events. Application monitoring is crucial in determining the characteristics of the applications in different states to control them. Tracking the way the application behaves gives an opportunity for application management tools to improve the current status of applications and even provide adaption mechanisms for resource optimization.

Figure 1 represents a monitoring architecture for scraping various metrics from different components of the cloud-fog system architecture with multiple clusters. A pod includes a collection of running applications or containers that are deployed together on the same host. The pods by design can expose the required metrics, which are then collected by the Application Monitoring Metrics Server. For the case of metrics related to node utilization, another separate component Node Exporter is used to collect the required metrics. The monitoring metrics are exposed using standardized techniques in the URL formats for easier querying by the Application Monitoring Metrics Server. For applications or jobs that are outside of the ‘local network’, their metrics can be scraped through a push gateway to the Application Monitoring Metrics Server. All metrics are stored in a time-series database located within the Application Monitoring Metrics Server for persistence. The recorded metrics are then displayed through

[1] https://www.nagios.org/

[2] https://www.datadoghq.com/

[3] https://prometheus.io/

dashboards or visualization tools, or they can be queried through various API clients.

Alerts manager is used for abnormal readings obtained from the server for administrative responses. Apart from the standard metrics, customized readings can also be configured at the level of the applications, such as App_A, App_B and App_C as shown in Figure 1.

In the fog and cloud layers, all functions related to the management of computing resources and task scheduling are performed by the broker component (Figure 2). In addition, the broker in the fog layer (in each fog device) is responsible for receiving application requests and determining the availability of computing resources. This will help to create an appropriate applications coordination schedule for the distribution of applications according to requirements and availability of resources.

The application registry is the part of the broker that provides some interface for users to submit applications as shown in Figure 1. Each application that is received in the registry is tagged with relevant information that specifies its type and computing resources requirements, which is stored in the Applications Database.

The next component of the broker is the Data Scraper, which is used to get information about the input data in the form of queries that come along or are needed with the registered applications. The aim is to establish the amount of data that is distributed in the computing platform. The returned results from the queries are included with information, such as locations for input data, and then made available for applications to use during scheduling.

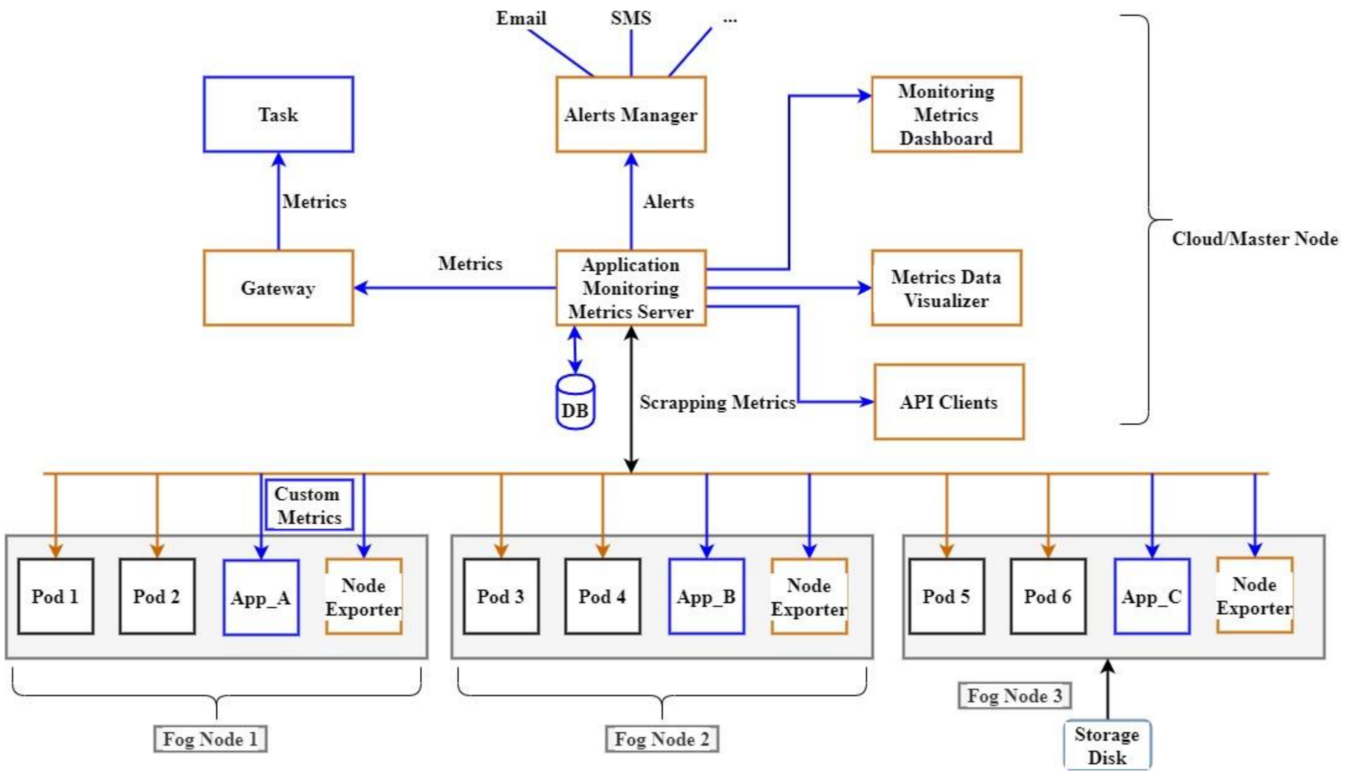


Figure 1. Architecture of grid service applications monitoring.

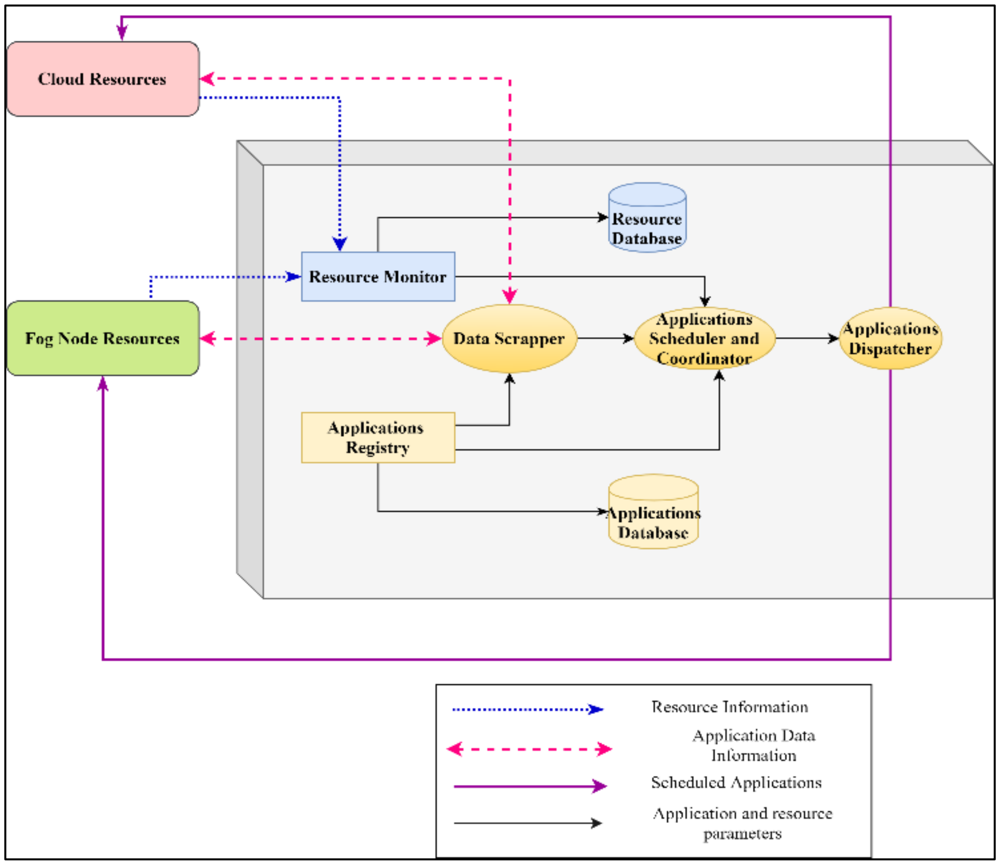


Figure 2. Architecure of service application broker.

Resource Monitor is another component of the broker that is used to collect information and monitor the use of computing resources available in the fog nodes and the cloud. This information is useful in determining the execution and data transfer rates and is stored in the Resource Database. This information is frequently changing as resources are consumed, and released or there is a change in configurations of policies, and therefore the database is also updated accordingly. Having the latest information about the status of the resources in the fog and cloud nodes is crucial in the scheduling plan of the application coordination.

The Application Scheduler and Coordinator componentis responsible for devising the best plan for scheduling and coordination of applications based on information about resource consumption, bandwidth usage, and other policies configured in the computing platform. The output of the scheduling plan is sent to the Applications Dispatcher, which then sends the tasks of each application to appropriate fog nodes and/or cloud as determined in the scheduling plan.

4. Experiments

To illustrate the grid applications/services monitoring for fault detection as a result of abnormal current and voltage readings in SDN, a simulation layout involving three transformers was set up to mimic SDN owned by TANESCO. The simulation layout is represented in Figure 3. A Raspberry Pi computer (acting as a fog node) was deployed in each of the three transformers in locations 1 to 3 to track the current and voltage readings of distribution lines. Most transformers may serve up to 250 customers. When excessive readings are detected on a fog node, the corresponding utility alarm system is activated to notify close by engineers. In addition, the observed event reading is also sent to the cloud for data aggregation. The aggregated information is accessible through the system's dashboard to the technicians.

To manage and coordinate the computing resources, the system monitoring architecture shown in Figure 4 was set up to collect, aggregate, process, and export information about running applications. cAdvisor[4] components provided metrics of the fog nodes and the master or cloud node. The metrics collected by cAdvisor are short-lived (only 60 seconds), and hence, for the long-term persistence of the metrics data, a backend time-series database, Prometheus, was used. Prometheus was chosen because of its powerful query language and ability to aggregate and filter data from distributed devices. On the aspect of visualization, the Grafana[5] tool was adopted because of its rich open-source-based features for visualizing graphs, charts and alarms.

This setup was put forward in order to facilitate the collection of the following metrics:

CPU Usage: this involved the measure of the utilization of the CPU by the applications on fog and master nodes. The cAdvisor component provides a Counter metric that calculates the *"per-second average rate of increase of the time series in the range vector."* It is named as *container_cpu_usage_seconds_total*. Prometheus captures this metric and sends it to Grafana to obtain CPU usage over a period of time for each application and the total CPU usage by all applications. It is measured in terms of percentage.

Memory Usage: It is the measure of memory that is being utilized by applications or processes in a given computing node. Again, the cAdvisor component exposes a metric called "*container_memory_usage_bytes*," which is then used by the Grafana tool to obtain the memory usage by each application and also the aggregate (overall) memory usage.

Network Activity: It provides the metric about the use of the network I/O in terms of data transfer (Tx) and data received (Rx) to give an indication of bandwidth usage and throughput.

Table 1 shows the configurations of experiments that were used to assess the performances. A minimum of 28 experiments were conducted to evaluate the resource usage (CPU and memory) among the computing nodes as more data arrived from the sensor nodes. The Master node was configured to host up to 60 containers, and the fog node capacities were limited to just 10 containers. The Master node was considered to have more computing resources than fog nodes. The first experiment involved the configuration of the Master with five (5) sensor nodes and no fog node. Then, the number of sensor nodes was increased to 40. The next experiment used the Master node with the addition of one (1) fog node, starting with 5 sensor nodes and then increasing to 40 nodes. The last experiment used the Master node and three (3) fog nodes with 40 sensor nodes.

The system was implemented using the FogFlow[6] platform. During operation of the platform, each sensor entity is associated with a single application instance (container), which means each fog node could only afford to accommodate a maximum of 10 sensors.

[4] https://github.com/google/cadvisor
[5] https://grafana.com/
[6] https://github.com/smartfog/fogflow

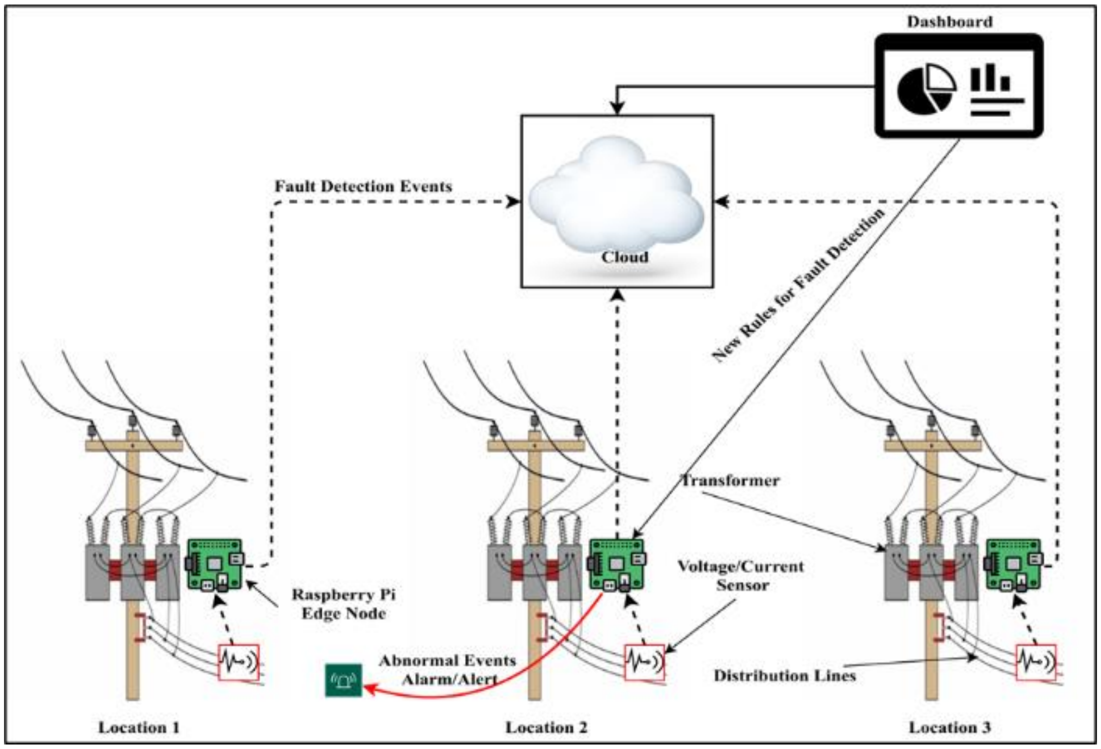


Figure 3. Fog Based Architecture for fault detection in secondary distribution network.

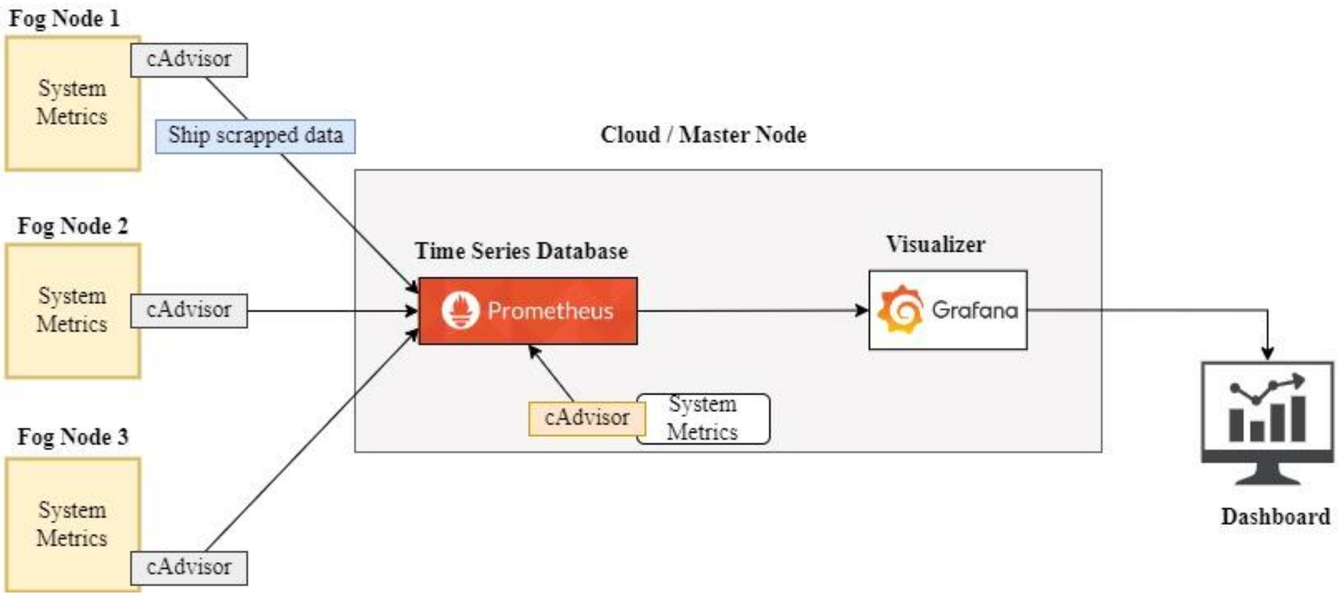


Figure 4. Setup for monitoring computing resources.

Table 1. Experiment configurations.

Computing Nodes		Number of Sensor Nodes Experimented							
Master Node	Fog Nodes	5	10	15	20	25	30	35	40
1	0	🟩	🟩	🟩	🟩	🟩	🟩	🟩	
1	1	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
1	2				🟩	🟩	🟩	🟩	🟩
1	3						🟩	🟩	🟩

5. Results

5.1 Resource Usage

Since real-world devices were used in this part of the study, the suitability of the proposed architecture was tested based on memory, CPU and network usage. From the implemented dashboard, utilization of these metrics was obtained periodically from the monitoring agents installed on all the computing nodes. Figure 5 to Figure 8 show the current usage metrics (in a particular period) captured from the computing nodes. Figure 5 illustrates the aggregated resource usage by fault detection application components that have been configured using containers. The overall number of active containers, total memory and CPU usage are shown.

Figure 6 and Figure 7 show the network usage in terms of data transmitted and received, respectively, recorded from the Master and fog node containers for a particular period. Generally, it can be observed that there is an increased network activity as sensor streams are increased to the computing platform. For instance, at time 12:00 (Figure 6) network usage for transmission was less than 1 kB/s, but at 12:48, the network usage was more than 12 kB/s. Further, the spike in network activity at some points could indicate a fault detection event. Fog nodes detected abnormal conditions (e.g., voltage/current faults) on the distribution lines. The detected fault data was transmitted to the cloud, causing increased network traffic. It is easier for system operators to make informed and quick decisions based on the dashboard's presented readings. Dynamic application orchestration methods in Fog computing can be enhanced through the use of up-to-date data based on network usage.

From the presented results, it can be seen that the more the amount of data to be processed the more the computing resources needed, particularly at the beginning of faults. Utility companies, such as TANESCO in Tanzania, should allocate corresponding computing resources to prepare to deal with those incidents.

5.2 CPU Usage

It can be observed that, overall, CPU usage increased with the addition of more sensors, as depicted in Figure 8 and Figure 10. In Figure 8, as more sensor streams were added for processing, more CPU was needed for each individual application. This is because there was an increasing amount of data coming from the sensors for processing.

The effectiveness of the proposed computing architecture in resource usage was evaluated based on the fault detection use case in the SDN. The intention was to estimate the bottlenecks and impact on the computing resources. Each containerized application was designed to handle a particular sensor node stream. In this way, it was easy to examine how additions of sensor nodes affect the usage of resources on the master node.

Figure 10 illustrates the resource (CPU) usage as the number of sensor nodes increased. It can be observed that, overall, as more sensors were added, the CPU usage was also increasing. This is because there was an increasing amount of data coming from the sensors for processing.

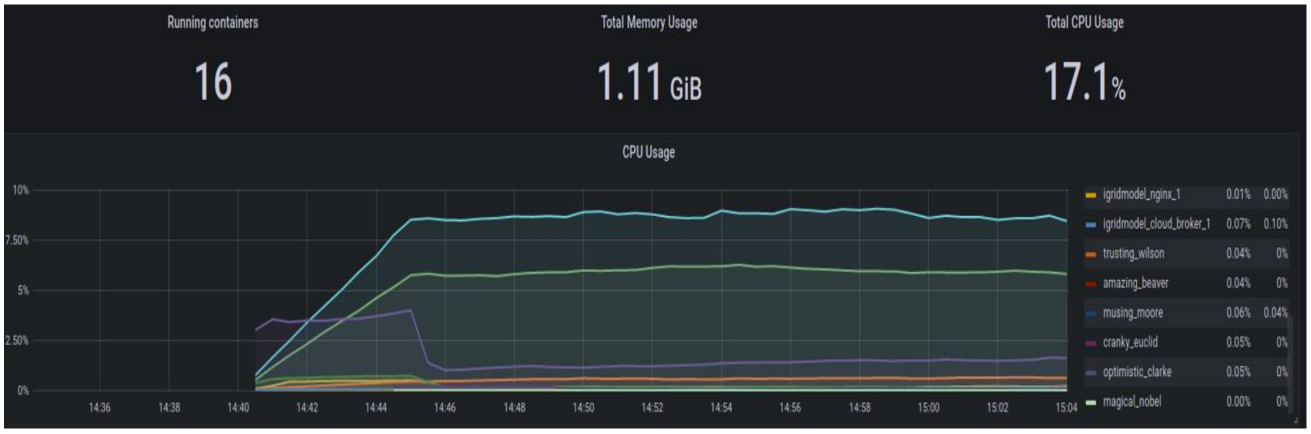


Figure 5. Aggregated system resource usage.

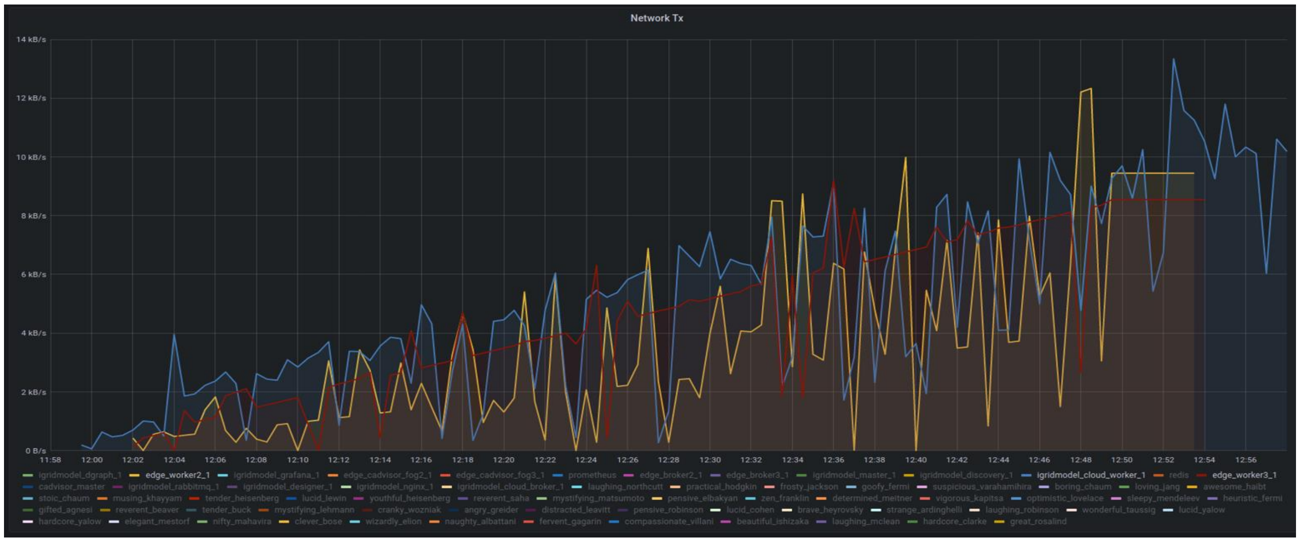


Figure 6. Network resource usage (Transmit).

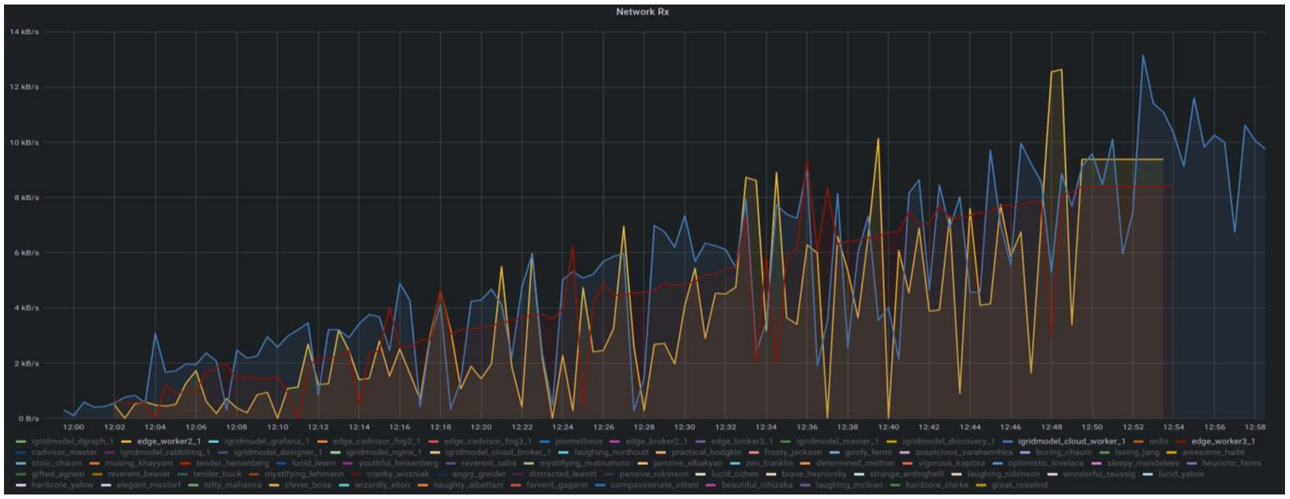
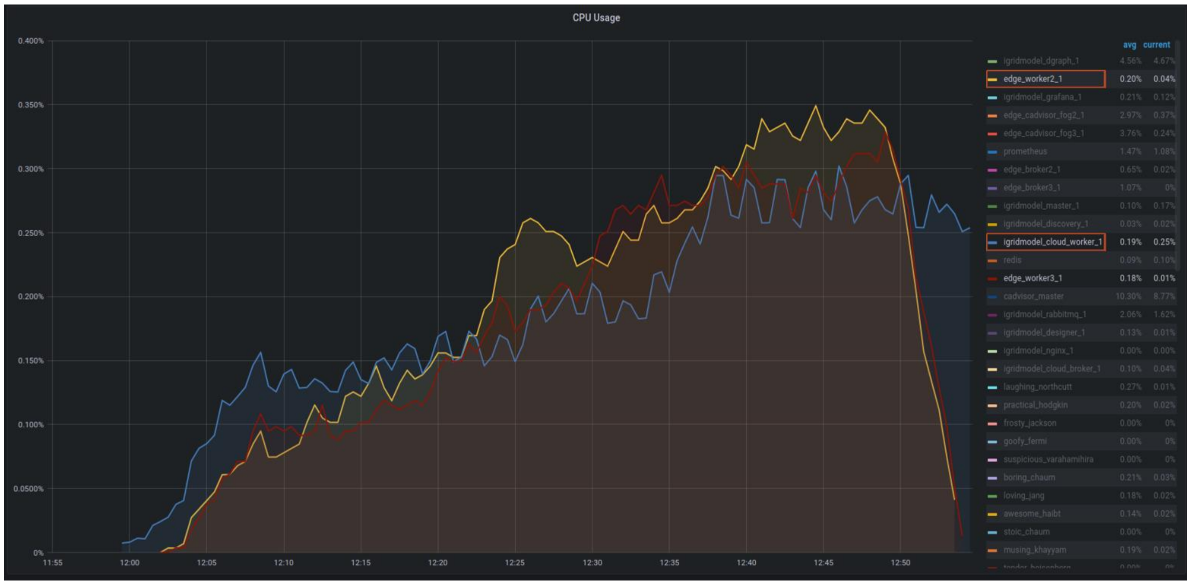
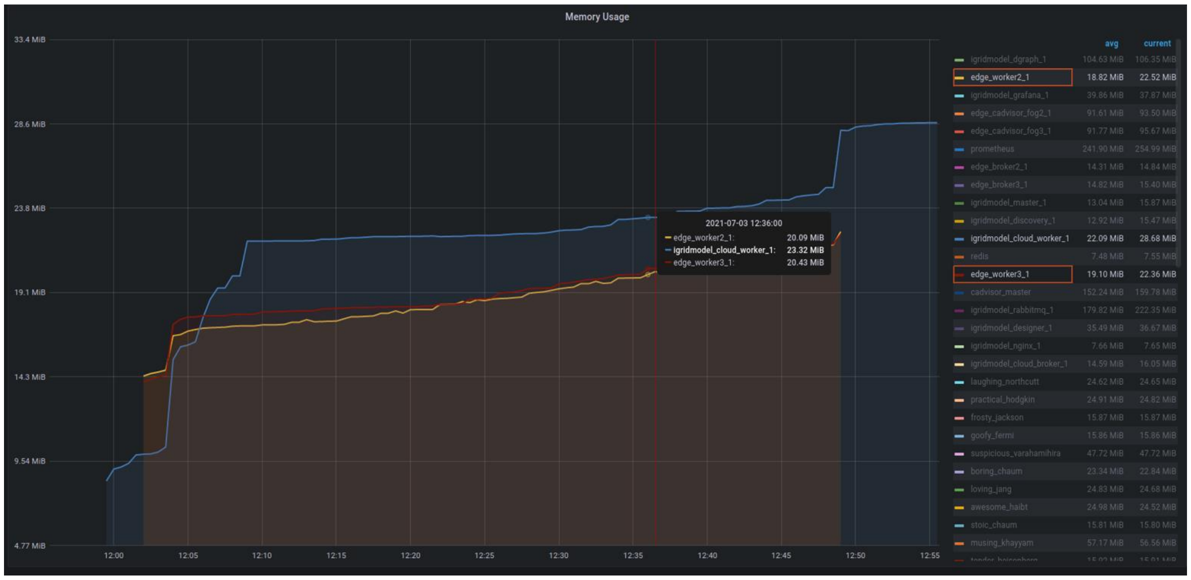


Figure 7. Network resource usage (Receive).



	avg	current
igridmodel_dgraph_1	4.56%	4.67%
edge_worker2_1	0.20%	0.04%
igridmodel_grafana_1	0.21%	0.12%
edge_cadvisor_fog2_1	2.97%	0.37%
edge_cadvisor_fog3_1	3.76%	0.24%
prometheus	1.47%	1.08%
edge_broker2_1	0.65%	0.02%
edge_broker3_1	1.07%	0%
igridmodel_master_1	0.10%	0.17%
igridmodel_discovery_1	0.03%	0.02%
igridmodel_cloud_worker_1	0.19%	0.25%
redis	0.09%	0.10%
edge_worker3_1	0.18%	0.01%
cadvisor_master	10.30%	8.77%
igridmodel_rabbitmq_1	2.06%	1.62%
igridmodel_designer_1	0.13%	0.01%
igridmodel_nginx_1	0.00%	0.00%
igridmodel_cloud_broker_1	0.10%	0.04%
laughing_northcutt	0.27%	0.01%
practical_hodgkin	0.20%	0.02%
frosty_jackson	0.00%	0%
goofy_fermi	0.00%	0%
suspicious_varahamihira	0.00%	0%
boring_chaum	0.21%	0.03%
loving_jang	0.18%	0.02%
awesome_haibt	0.14%	0.02%
stoic_chaum	0.00%	0%
musing_khayyam	0.19%	0.02%

Figure 8. CPU Usage by individual applications (Containers).



	avg	current
igridmodel_dgraph_1	104.63 MiB	106.35 MiB
edge_worker2_1	18.82 MiB	22.52 MiB
igridmodel_grafana_1	39.86 MiB	37.87 MiB
edge_cadvisor_fog2_1	91.61 MiB	93.50 MiB
edge_cadvisor_fog3_1	91.77 MiB	95.67 MiB
prometheus	241.90 MiB	254.99 MiB
edge_broker2_1	14.31 MiB	14.84 MiB
edge_broker3_1	14.82 MiB	15.40 MiB
igridmodel_master_1	13.04 MiB	15.87 MiB
igridmodel_discovery_1	12.92 MiB	15.47 MiB
igridmodel_cloud_worker_1	22.09 MiB	28.68 MiB
redis	7.48 MiB	7.55 MiB
edge_worker3_1	19.10 MiB	22.36 MiB
cadvisor_master	152.24 MiB	159.78 MiB
igridmodel_rabbitmq_1	179.82 MiB	222.35 MiB
igridmodel_designer_1	35.49 MiB	36.67 MiB
igridmodel_nginx_1	7.66 MiB	7.65 MiB
igridmodel_cloud_broker_1	14.59 MiB	16.05 MiB
laughing_northcutt	24.62 MiB	24.65 MiB
practical_hodgkin	24.91 MiB	24.82 MiB
frosty_jackson	15.87 MiB	15.87 MiB
goofy_fermi	15.86 MiB	15.86 MiB
suspicious_varahamihira	47.72 MiB	47.72 MiB
boring_chaum	23.34 MiB	22.84 MiB
loving_jang	24.83 MiB	24.68 MiB
awesome_haibt	24.98 MiB	24.52 MiB
stoic_chaum	15.81 MiB	15.80 MiB
musing_khayyam	57.17 MiB	56.56 MiB

Figure 9. Memory usage by individual applications (containers).

As expected, the highest value of 39% was recorded when 35 sensor nodes were deployed, and the lowest value of 19.5% was measured with 5 sensor nodes. In each case, the CPU usage increased up to the highest peak and then levelled across the execution-time window. This is due to the fact that higher CPU power was required during the initialization of application containers on the computing node. Thirty-five Sensor nodes required 35 containers to be initialized to handle data streams as compared to 5 containers needed for 5 sensor nodes. Therefore, more CPU power would be needed in the former case than in the latter. The same trend of increased CPU usage in relation to the increasing number of sensor nodes was observed in all the fog nodes.

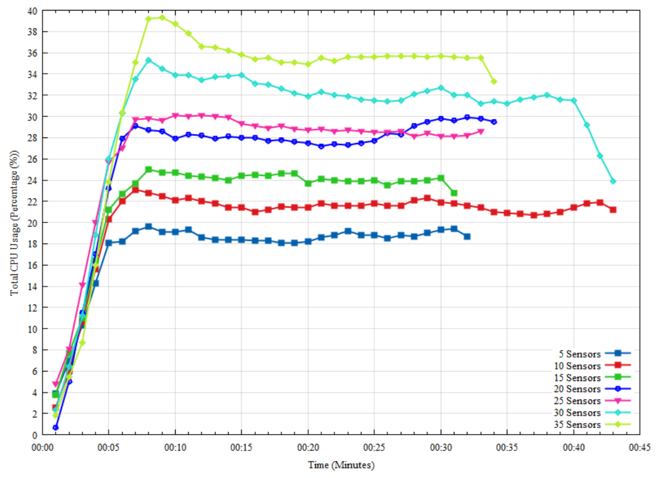


Figure 10. CPU Usage based on the number of sensor nodes.

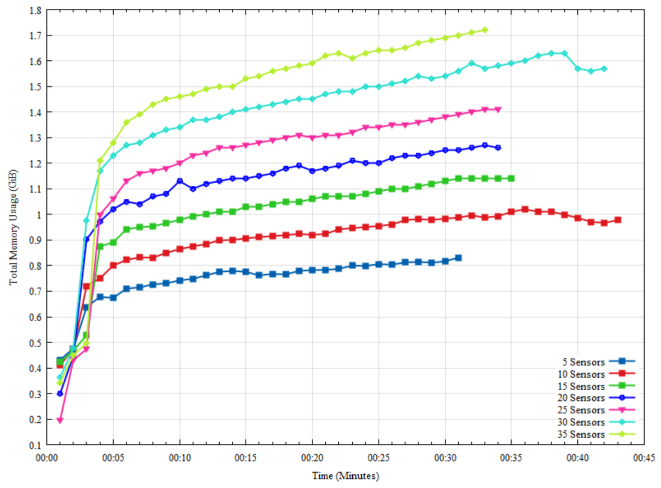


Figure 11. Memory usage based on the number of sensor nodes

5.3 Memory Usage

Figure 9 and Figure 11 present memory usage in relation to the increasing number of sensor nodes. Noting from Figure 9, each individual application required more memory to deal with increasing sensor stream data. Overall, it could be noted that there was an increase in memory usage as more sensors were added, with the highest usage posted at 1.72 GiB with 35 sensor nodes and the lowest being at 0.85 GiB when 5 sensor nodes were used (Figure 11). This was attributed to the number of application instances (containers) that needed to be started for each sensor entity. It can also be observed that there was a sharp jump in memory usage for each group of sensors that were added. For instance, with 35 sensor nodes, memory usage went from 0.5 GiB to 1.2 GiB in less than a minute. This information is crucial in determining the threshold amount of memory needed in the worst-case scenario, which might require all sensor nodes to send data simultaneously.

5.4 Application Coordination

Each sensor node is associated with one application instance (a container) that is invoked by the assigned worker on the master or fog node. Coordination is needed so that some computing nodes are not overloaded while others stay idle. Moreover, the coordination process would allocate applications to the semi-autonomous fog nodes to demonstrate the distributed computing concept. In this case, the author adopted the coordination mechanism used in Fogflow. When extra fog nodes are added, the coordination process needs to be more intelligent by ensuring that local data processing is given higher priority and the load balancing is handled among the computing devices.

Based on experiment configurations in Table 1, fifty sensors (50) were added to demonstrate how applications to process sensor streams can be coordinated between Master and fog nodes.

Table 2. Tasks distribution between master node and one fog node.

Tasks	Tasks Distribution			
	Master Node (MN)	Percent. MN (%)	Fog Node 1 (FN1)	Percent. FN1 (%)
10	0	0.00	10	100
20	10	50.00	10	50
30	20	66.67	10	33.33
40	30	75.00	10	25
50	40	80.00	10	20

Table 3 Tasks distribution between master node and two fog nodes.

	Task Distribution					
Tasks	Master Node (MN)	Per cent. MN (%)	Fog Node1 (FN1)	Per cent. FN1 (%)	Fog Node2 (FN2)	Per cent. FN2(%)
10	0	0	0	0	10	100
20	0	0	10	50	10	50
30	10	33.33	10	33.33	10	33.33
40	20	50	10	25	10	25
50	30	60	10	20	10	20

Table 2, Table 3, and

Table 4 present task allocations observed across computing nodes when the experiments were carried out. The findings show that the workload assigned to the Master node dropped significantly from 80% with one fog node to 40% with all three fog nodes deployed when 50 tasks were allocated.

This is also well illustrated by Figure 12, in which the Master node was allocated 0 tasks for the first 30 tasks that were available for allocation. This signifies that a small amount of data would be sent to the centralized Master node, hence reducing network traffic latency and processing time. Moreover, utilization of the fog nodes is increased as more tasks are allocated across these devices, and as such is the case, there is a profound effect on latency-sensitive applications, which would require real-time decision-making.

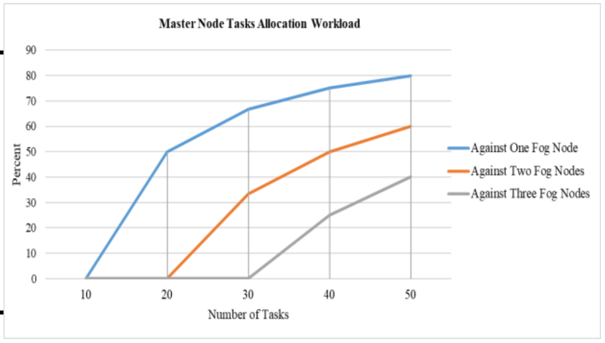


Figure 12. Master node tasks allocation workload.

Figure 13 shows the workload allocation for the Fog Node 1 in conjunction with other computing nodes. When 10 tasks were made available for allocation, the coordination manager allocated 100% of the tasks to the fog node. When the extra fog node was added (Fog Node 2), the allocation dropped considerably to 0%, a difference of 100%. This is explained by the fact that Fog Node 1 was geographically much farther than Fog Node 2 in relation to the 10 sensor nodes. The same trend was observed when Fog Node 3 was added.

Table 4 Tasks Distribution Between Master Node and Three Fog Nodes

	Task Allocation							
Tasks	Master Node (MN)	Per cent. MN(%)	Fog Node1 (FN1)	Per cent. FN1(%)	Fog Node2 (FN2)	Per cent. FN2(%)	Fog Node3 (FN3)	Per cent. FN3(%)
10	0	0	2	20	5	50	3	30
20	0	0	5	25	10	50	5	25
30	0	0	10	33.33	10	33.33	10	33.33
40	10	25	10	25	10	25	10	25
50	20	40	10	20	10	20	10	20

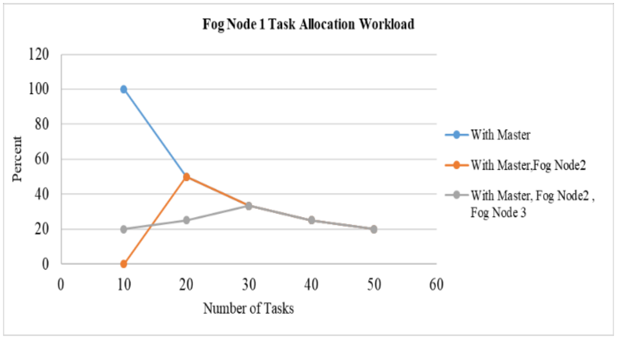


Figure 13. Fog node 1 task allocation workload.

Preliminary results indicate that the introduction of fog nodes significantly reduces latency, cutting processing times from seconds to milliseconds. The improved response times enhance fault detection, enabling quicker interventions in the event of power irregularities.

Furthermore, fog nodes efficiently handle real-time data, reducing the volume of raw data sent to the cloud by processing it locally. Only critical data is transmitted to the cloud for long-term storage and analysis, leading to more efficient data throughput.

In the aspect of scalability performance, simulations and pilot deployments show that the architecture scales well with increasing numbers of sensors. Fog nodes handle additional loads without introducing performance bottlenecks, and the cloud infrastructure adapts to store and analyze increasing amounts of data.

Although the architecture demonstrates substantial benefits in latency reduction and real-time processing, challenges remain regarding the cost of deploying fog nodes across SDN's vast network, such as that of TANESCO. Additionally, data security measures require continuous improvement to mitigate the risks of cyberattacks. This can form the basis for future work.

6. Conclusion

The proposed fog-based computing architecture and application coordination mechanism effectively improved the computing workload distribution of up to 70% for fog-based architectures and to 40% for cloud-based architectures, indicating that most data processing is localized to fog nodes. Moreover, fog-based architecture provided improvements of 70%, compared to cloud-only architectures, indicating further that most data processing is localized, hence reducing costs related to transfer and processing to cloud systems. For that matter, it can be confidently stated that pushing data processing and intelligence to the edges can considerably enhance the fault management processes in the electrical secondary distributed network.

Future work could extend the evaluation to include latency, makespan, miss ratio, average delay, and cost per execution to complement the equitable assessment of the proposed architecture. Further, areas related to limited computation power, energy consumption, data privacy and security of fog nodes and networks can be explored.

ACKNOWLEDGEMENT

The authors express their heartfelt gratitude to SIDA, the University of Dar es Salaam, and the University of Dodoma for their invaluable support, which was instrumental in the successful completion of this research.

CONTRIBUTION OF CO-AUTHORS

Gilbert M. Gilbert	[ORCID: 0000-0002-9615-9715]	Conceived the idea, and analyzed results and wrote the paper
Shililiandumi Naiman	[ORCID: 0000-0002-8499-7543	Provided supervision to experiments

REFERENCES

[1] M. I. Henderson, D. Novosel, and M. L. Crow, “Electric Power Grid Modernization Trends, Challenges, and Opportunities,” 2017.

[2] A. Abdalla and K. Ibwe, “Smart Grid in Tanzania: Research Opportunities,” *Tanzania Journal of Engineering and Technology*, vol. 42, no. 2, pp. 170–183, Jun. 2023, doi: 10.52339/tjet.v42i2.838.

[3] J. Powell, A. McCafferty-Leroux, W. Hilal, and S. A. Gadsden, “Smart Grids: A Comprehensive Survey of Challenges, Industry Applications, and Future Trends,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.13105.

[4] S. Ma, H. Zhang, and X. Xing, “Scalability for Smart Infrastructure System in Smart Grid: A Survey,” *Wirel Pers Commun*, vol. 99, no. 1, pp. 161–184, Mar. 2018, doi: 10.1007/s11277-017-5045-y.

[5] P. Boccadoro, “Smart Grids empowerment with Edge Computing: An Overview,” Sep. 2018, [Online]. Available: http://arxiv.org/abs/1809.10060.

[6] G. M. Gilbert, S. Naiman, H. Kimaro, and B. Bagile, “A critical review of edge and fog computing for smart grid applications,” in *IFIP Advances in Information and Communication Technology*, P. Nielsen and H. C. Kimaro, Eds., Cham: Springer, Cham, May 2019, pp. 763–775. doi: 10.1007/978-3-030-18400-1_62.

[7] M. Forcan and M. Maksimović, “Cloud-Fog-based approach for Smart Grid monitoring,” *Simul Model Pract Theory*, vol. 101, no. June 2019, p. 101988, 2020, doi: 10.1016/j.simpat.2019.101988.

[8] R. Mahmud, R. Kotagiri, and R. Buyya, “Fog Computing: A Taxonomy, Survey and Future Directions,” 2018, pp. 103–130. doi: 10.1007/978-981-10-5861-5_5.

[9] S. N. Srirama, “A decade of research in fog computing: Relevance, challenges, and future directions,” *Softw Pract Exp*, vol. 54, no. 1, pp. 3–23, Jan. 2024, doi: 10.1002/spe.3243.

[10] A. Prasad, J. Belwin Edward, and K. Ravi, “A review on fault classification methodologies in power transmission systems: Part-II,” *Journal of Electrical Systems and Information Technology*, vol. 5, no. 1, pp. 61–67, May 2018, doi: 10.1016/j.jesit.2016.10.003.

[11] J. De La Cruz, E. Gómez-Luna, M. Ali, J. C. Vasquez, and J. M. Guerrero, “Fault Location for Distribution Smart Grids: Literature Overview, Challenges, Solutions, and Future Trends,” *Energies (Basel)*, vol. 16, no. 5, p. 2280, Feb. 2023, doi: 10.3390/en16052280.

[12] M. Nunes *et al.*, "Fault detection and location in low voltage grids based on RF-MESH sensor networks,"*CIRED Workshop 2016*, Helsinki, 2016, pp. 1-4, doi: 10.1049/cp.2016.0743.

[13] H. B. Gooi, T. Wang, and Y. Tang, “Edge Intelligence for Smart Grid: A Survey on Application Potentials,” *CSEE Journal of Power And Energy Systems*, vol. 9, no. 5, p. 1623, 2023, doi: 10.17775/CSEEJPES.2022.02210.

[14] L. Mei, M. Qi, and Z. Li, “Edge-Cloud Collaborative Fault Detection for Distribution Networks Using Attention Mechanism,” in *2022 IEEE Power & Energy Society General Meeting (PESGM)*, IEEE, Jul. 2022, pp. 1–5. doi: 10.1109/PESGM48719.2022.9917097.

[15] D. Sodin, U. Rudež, M. Mihelin, M. Smolnikar, and A. Čampa, “Advanced Edge-Cloud Computing Framework for Automated PMU-Based Fault Localization in Distribution Networks,” *Applied Sciences*, vol. 11, no. 7, p. 3100, Mar. 2021, doi: 10.3390/app11073100.

[16] W. Huo, F. Liu, L. Wang, Y. Jin, and L. Wang, “Research on Distributed Power Distribution Fault Detection Based on Edge Computing,” *IEEE Access*, vol. 8, pp. 24643–24652, 2020, doi: 10.1109/ACCESS.2019.2962176.

[17] S. Netsanet, D. Zheng, Z. Wei, and G. Teshager, “Cognitive Edge Computing–Based Fault Detection and Location Strategy for Active Distribution Networks,” *Front Energy Res*, vol. 10, Aug. 2022, doi: 10.3389/fenrg.2022.826915.

[18] A. S. Alhanaf, H. H. Balik, and M. Farsadi, “Intelligent Fault Detection and Classification Schemes for Smart Grids Based on Deep Neural Networks,” *Energies (Basel)*, vol. 16, no. 22, p. 7680, Nov. 2023, doi: 10.3390/en16227680.

[19] J. Li, C. Gu, Y. Xiang, and F. Li, “Edge-cloud Computing Systems for Smart Grid: State-of-the-art, Architecture, and Applications,” *Journal of Modern Power Systems and Clean Energy*, vol. 10, no. 4, pp. 805–817, 2022, doi: 10.35833/MPCE.2021.000161.

[20] B. Cheng, E. Kovacs, A. Kitazawa, K. Terasawa, T. Hada, and M. Takeuchi, “FogFlow: Orchestrating IoT services over cloud and edges,” *NEC Technical Journal*, vol. 13, no. 1, pp. 48–53, 2018.

[21] J. Li *et al.*, “Resource Orchestration of Cloud-Edge–based Smart Grid Fault Detection,” *ACM Trans Sens Netw*, vol. 18, no. 3, pp. 1–26, Aug. 2022, doi: 10.1145/3529509.

[22] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini, “DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds,” *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2041–2056, Oct. 2013, doi: 10.1016/j.future.2013.04.022.