Final paper

by Tasnim Fuyara Chhoan

Submission date: 03-Jan-2024 05:33PM (UTC+0530)
Submission ID: 2266387520
File name: rrigation_Systems_Using_Kubernetes_Managed_Edge_Computing_4.pdf (3.24M)
Word count: 4814
Character count: 29349

Optimization of Drip Irrigation Systems Using Kubernetes Managed Edge Computing

1st Abrar Al Sayem
School Data and Sciences
BRAC University
Dhaka, Bangladesh
abrar.al.sayem@g.bracu.ac.bd

2nd Tasnim Tayara Chhoan
School Data and Sciences
BRAC University
Dhaka, Bangladesh
tasnim.tayara.chhoan@g.bracu.ac.bd

3rd Ripa Sarkar
School Data and Sciences
BRAC University
Dhaka, Bangladesh
ripa.sarkar@g.bracu.ac.bd

4th Asika Islam
School Data and Sciences
BRAC University
Dhaka, Bangladesh
asika.islam@g.bracu.ac.bd

5thMd Humaion Kabir Mehedi
School Data and Sciences
BRAC University
Dhaka, Bangladesh
email address or ORCID

6th Annajiat Alim Rasel
School Data and Sciences
BRAC University
City, Country
annajiat@gmail.com

***Abstract*—This paper presents an innovative approach to optimizing drip irrigation systems through the deployment of Kubernetes-managed edge computing. The objective is to harness the power of modern distributed computing to enhance the efficiency and responsiveness of irrigation in agricultural practices. The proposed system architecture integrates cloud-based nodes with field-based nodes, utilizing Kubernetes for orchestration, Docker for containerization, and micro-services for real-time monitoring and control of irrigation processes. By deploying sensor and micro-controller containers directly in the field, the system significantly reduces latency, enabling immediate data processing and actuation based on environmental conditions. The study delves into the use of Prometheus for gathering performance metrics and Grafana for visualizing this data, offering a comprehensive monitoring solution that not only tracks but also predicts irrigation needs through advanced computational analysis. The discussion highlights the system's modularity, scalability, and potential to leverage predictive analytics while addressing challenges such as technical complexity, and system resilience. The conclusion underscores the transformative impact of Kubernetes in precision agriculture, suggesting that such systems can lead to more informed water use, improved crop yields, and sustainable farming practices. This research contributes to the field by demonstrating the feasibility of a Kubernetes-managed edge computing framework in agriculture, setting the stage for future advancements in smart farming technologies.**

***Index Terms*—Drip Irrigation, Edge Computing, HPC, Kubernetes, Containerization, Prometheus**

I. Introduction

For thousands of years, the ancient cultures of all nations have focused on agricultural practices as a means of overall development. Human energy needs, specifically about consuming nutritious foods, are impacted by agricultural operations. [1]. The revolution of precision agriculture has transformed the approach to farming, with drip irrigation systems leading the way in this revolution. Drip irrigation, a technique that enables accurate delivery of water directly to the roots of plants, greatly improves water efficiency, crop yield, and quality. Simultaneously, it minimizes water wastage and reduces the environmental impact of farming practices [2]. Moreover, it made better fertilizer management and nutrient distribution possible, which reduced plant stress, allowed for earlier harvesting, improved crop quality, and increased production consistency. [3]. It also calculates the required amounts of water and fertilizer for crops based on soil nutrient and water balance. Utilizing a precision drip irrigation system, it delivers fertilizer in stages around the crop's root zone. Drip irrigation allows for flexible adjustments in nutrient types and quantities. These characteristics capitalize on synergies, decrease nutrient fixation by the soil, provide a stable environment for root growth, and enable adaptable nutrient supply aligned with climate, soil conditions, and the nutritional requirements of crops.

In recent times, the significance of distributed systems in agriculture has grown noticeably. Distributed computing architectures, especially within the framework of the Internet of Things (IoT), have demonstrated considerable promise in improving the efficiency and scalability of agricultural operations [4] [5]. These systems make it possible to collect and analyze large amounts of data, which facilitates informed decision-making and effective resource allocation. Agriculture 4.0's use of distributed and cloud architectures for data management highlights the sector's growing trend toward digital transformation [6].

Edge Computing presents an opportunity for the farming community to enhance access to and utilization of smart agriculture services. For service providers, integrating an Edge model into agricultural designs is a problem. The agricultural industry, especially at the farm level, provides an excellent example to verify the effectiveness of the Edge computing model for delivering smart services [1].

Kubernetes is an open-source platform designed to automate the deployment, scaling, and management of application containers across host clusters and has become a prominent

figure in the domain of distributed systems [7]. Its proficiency in effectively overseeing containerized applications positions it as a suitable choice for deploying and orchestrating intricate agricultural systems. Kubernetes' capacity in fog computing settings, particularly in overcoming limitations and enhancing the capabilities of distributed systems, holds particular significance for agricultural applications [8]. Prometheus, on the other hand equipped with PromQL providing easy monitoring of Kubernetes nodes and services enabling real-time monitoring and visualizations by utilizing Grafana tools [9].

II. Research Objective

In the context of smart agriculture, our study aims to explore a hypothetical to optimize distributed drip irrigation systems. The framework utilizes Kubernetes' container orchestration and robustness to improve the efficiency of drip irrigation and Prometheus for metric analysis, contributing to sustainable agricultural practices. The paper will conduct a literature review, propose a theoretical framework, and explore the potential impact and future directions of this innovative approach. Numerous studies have demonstrated the promising integration of Kubernetes in agricultural systems, offering a pathway to advance precision agriculture and address the challenges of contemporary agriculture.

III. Literature Review

Edge Computing in agriculture utilizes local processing near data sources [10], overcoming challenges posed by limited rural internet access. This approach, demonstrated in prototypical systems across various agricultural domains, enables real-time analytics and decision-making on Edge nodes. To fully realize its potential, integration of robust commercial platforms is crucial, recognizing the constraint of inadequate internet access in farming communities. Drip irrigation technology has undergone notable advancements, becoming increasingly advanced and effective. Designed to deliver water directly to the root zone of plants, these systems reduce both evaporation and runoff, ultimately maximizing the efficient use of water. As per [11], these systems have a pivotal function in modern agriculture, contributing to increased crop productivity and fostering water conservation. However, the efficacy of these systems relies heavily on accurate control and monitoring, prompting the incorporation of advanced technologies.

The agricultural industry has experienced a notable increase in the utilization of technologies like IoT, cloud computing, and artificial intelligence. According to [3], the application of AI in precision agriculture shows promise in improving crop management and decision-making. Additionally, [4] investigates the deployment of distributed computing architectures in IoT scenarios, emphasizing their contribution to real-time data processing and decision-making within the agricultural domain.

Kubernetes has become a pivotal technology for overseeing distributed systems, especially in container orchestration. [7] explores the utilization of Kubernetes in high-performance computing (HPC) systems, showcasing its effectiveness in orchestrating containerized applications. In a related context, [8] evaluates Kubernetes within the realm of fog computing, recognizing its viability and pinpointing areas for enhancement in distributed settings. These investigations underscore the potential of Kubernetes in improving the scalability and dependability of distributed systems.

Despite the progress in utilizing technology for agricultural purposes, there are notable research deficiencies, particularly in the incorporation of Kubernetes into drip irrigation systems. While investigations such as those conducted by [12] and [13] shed light on deploying microservices and network solutions through Kubernetes, a specific research gap exists in understanding its role in enhancing the efficiency of drip irrigation systems. This gap offers the potential to delve into customizing Kubernetes to address the distinctive challenges and needs of precision agriculture and drip irrigation systems.

Kubernetes also focuses on advanced monitoring and management techniques for the clusters, particularly in the context of scientific computing and high-performance computing environments. This paper [14], introduces SLATE (Services Layer at the Edge), a project aimed at accelerating collaborative scientific computing. It offers a reliable framework for container orchestration, with a specific emphasis on the Science DMZ (Demilitarized Zone). This framework aims to streamline the development of sophisticated multi-institution platforms and innovative science gateways. This study [15] proposes a new solution for automatic anomaly detection and alerting to aid system administrators in the early detection and prevention of defects. The solution revolves around Prometheus, an open-source monitoring system, aiming to reduce the need for human intervention and enhance proactive system management. In a different study [9], a novel architecture is outlined for the Notification Infrastructure (OMNI) and Operations Monitoring at NERSC. This design integrates diverse technologies such as Kubernetes, Grafana, Prometheus, and predictive platforms.

IV. Kubernetes Architecture Overview

This research focuses on developing a theoretical framework to incorporate Kubernetes into drip irrigation systems for advanced agriculture setup. It delves into the principles of distributed systems, the structure and elements of Kubernetes, their application in drip irrigation systems, and the potential advantages and obstacles of this integration. Distributed systems, are characterized by a network of independent computers that function as a single system to users [6]. They enable efficient data processing, storage, and retrieval, which are crucial for precision agriculture practices [4]. The distributed nature of these systems allows for scalability, fault tolerance, and resource sharing, making them ideal for agricultural applications where data is collected from various sources and locations. This framework also explores the potential benefits and challenges of such integration.

A. Kubernetes Architecture and Components

Kubernetes, an open-source platform designed to automate the deployment, scaling, and operation of application containers across clusters of hosts controlled by **kubernetes Master** [16]. It is the control plane of a Kubernetes cluster, responsible for managing its state and configuration. It includes several components: the API server (orchestrates the Kubernetes API), the etcd storage (stores all cluster data), the scheduler (assigns work, like Pods, to nodes), and the controller manager (oversees various controllers that regulate the state of the cluster). Here we propose a Kubernetes architecture for running a DI distributed systems resiliently with scaling, failover, deployment patterns, and more, as shown in Figure 1. Some Key components of Kubernetes include:

- **Pods:** The smallest deployable units created and managed by Kubernetes.
- **Nodes:** Physical or virtual machines where pods are scheduled and run.
- **Services:** An abstraction layer that defines a logical set of Pods and a policy to access them.
- **Docker:** Docker is a container runtime, used in nodes (individual machines or VMs in a cluster) to run containers. It packages an application and its dependencies into a container. Containers are isolated environments where applications can run. This isolation ensures that the application behaves consistently across different environments, as it includes all necessary dependencies.
- **Kubelet:** This is an agent that runs on each node within a Kubernetes cluster. Its primary role is to ensure that containers are running in a Pod as expected. A Pod is the smallest deployable unit in Kubernetes, which can contain one or more containers.
- **Kube-Proxy:** This component maintains network rules on the nodes of a Kubernetes cluster. These rules allow for network communication to your Pods from network sessions both inside and outside of your cluster. It essentially helps in directing traffic to the correct containers based on the IP address and port number of the incoming request.

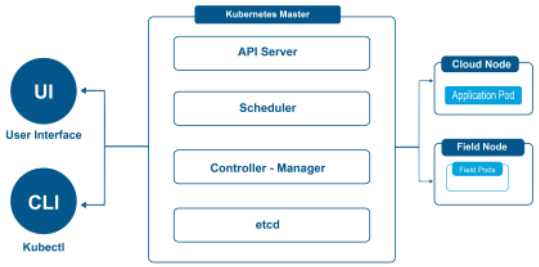


Fig. 1: Kubernetes Architecture

- **API Server:** The API Server in Kubernetes validates and configures data for various API objects like pods, services, replication controllers, etc. It handles REST operations and acts as the interface to the cluster's shared state, which all other components interact with. [16].
- **Controller Manager:** The Controller Manager operates background processes known as controllers, which perform routine tasks in the cluster. For instance, the Replication Controller ensures that the actual number of replicas for a service aligns with the defined number in the cluster.

B. Cloud Node

Cloud Node in Figure 2, is the central hub of the Kubernetes-based system. They host application pods that include both frontend and backend components, crucial for the system's operation and user interaction. Kubernetes architecture supports the separation of concerns between the frontend and backend while maintaining a cohesive operational model. It provides mechanisms for service discovery, scaling, load balancing, and self-healing, which are critical for maintaining the reliability and efficiency of the application serving the system's requirements.

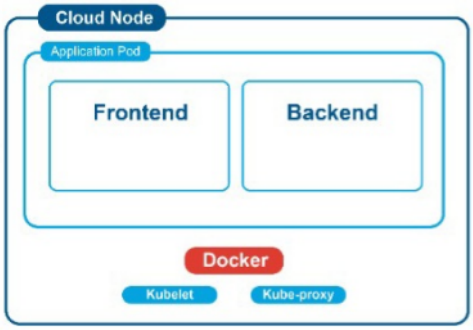


Fig. 2: Cloud Node

- **Frontend Components:** The frontend serves as the user interface, offering real-time and historical data visualizations and control mechanisms. This interface allows users to interact with the system, monitor conditions, and make informed decisions about their irrigation practices
- **Backend Components:** The backend is responsible for data processing, including applying complex analytics to sensor data. It manages the operational requirements of the DI system, such as calculating water needs, scheduling irrigation, and optimizing resource use. Processing the enormous volumes of data produced by the Field Nodes and turning it into useful insights depends on this component.

C. Field Node

1) Proposed Fields Setup: In the figure 3, there is a schematic overview of a networked irrigation system. It shows multiple plots of land, each with its own set of plants.

Each plot in this setup is outfitted with sensors and emitters dedicated to water distribution to plants. These components are linked to a microcontroller, which serves as the local 'brain' of the system. The microcontroller is not only responsible for managing the sensors and emitters but also maintains a connection to a cloud server and a water supply. This integration allows for efficient and smart management of water distribution, leveraging real-time data and cloud computing for optimal irrigation and plant care. The cloud server collects data from the sensors and sends commands to the emitters, managing the irrigation based on various factors such as soil moisture, weather forecasts, and water availability. Kubernetes' ability to handle containerized applications can be leveraged to deploy and manage IoT applications that are integral to drip irrigation systems, such as moisture sensors, rainfall, and others.

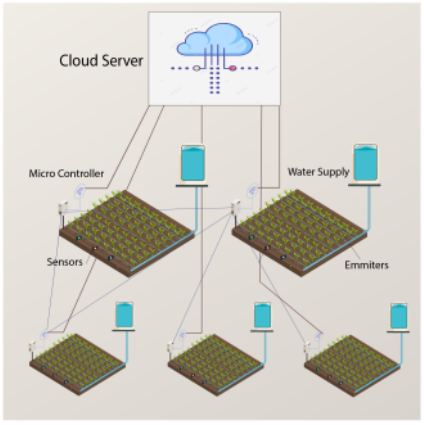


Fig. 3: A Drip Irrigation System

Field Node is situated in the agricultural fields and consists of field pods equipped with various sensors and microcontrollers. In the node, they are virtual representations of the group of IoT components, as shown in Figure 4. Each Pod contains sensors and a microcontroller that interacts directly with the physical world, such as collecting data from the environment or controlling other devices. The Field Node is designed to communicate seamlessly with Cloud Nodes. Then the sensors transmit their collected data to the cloud backend for comprehensive analysis. This integration ensures that the vast amounts of data generated in the field are effectively utilized to inform irrigation strategies.

V. Edge Computing and Data Storage in Kubernetes

A. Computing Containers

Kubernetes, traditionally used in cloud computing, has been adapted for edge environments to bring its powerful orchestration capabilities closer to the source of data generation—the field nodes. The field nodes are configured as worker nodes within the Kubernetes architecture, each running a lightweight

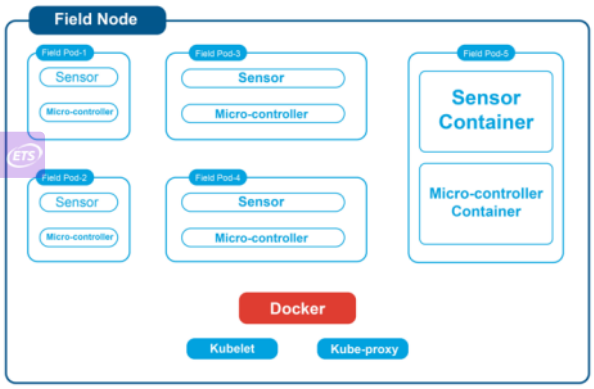


Fig. 4: Field Node

version of Kubernetes capable of managing edge workloads [7]. This allows for the deployment and management of workloads directly on the field nodes, enabling real-time data processing and immediate actuation for irrigation control. The proximity of computation to the data source significantly reduces latency, a critical factor in time-sensitive agricultural decision-making [8]. Kubernetes' scheduler ensures that these pods are deployed on the most appropriate field node based on the current load and resource availability, optimizing the system's responsiveness and reliability [17]. The containerized approach also simplifies the deployment and scaling of applications, allowing for rapid updates and rollbacks as necessary [14]. For instance, if a particular sensor type requires an update to its data processing algorithm, only the relevant container needs to be updated without affecting the rest of the system. This level of automation in container orchestration is essential to maintain the high availability and efficiency of the irrigation system, ensuring that water is delivered precisely and judiciously to the crops.

B. Data Flow and Persistent Volume

Kubernetes employs an overlay network or Container Network Interface (CNI) to enable communication among multiple containers. In overseeing the communication between containers and pods [8], pods in the field gather data from sensors via the microcontroller and may temporarily store it in local storage or memory. This data is then sent to the cloud node's backend services, which might involve using Kubernetes services for reliable communication. Therefore, persistent storage options in Kubernetes, such as PersistentVolumes [18], could be used to store sensor data if needed to prevent data loss or caching or local processing and ensure the efficient and reliable operation of storage resources in Kubernetes. A persistent volume in the cluster refers to storage that has either been set up by an administrator or automatically provisioned through Storage Classes in response to a Persistent Volume Claim (PVC) request [18]. This setup shows how

storage resources are dynamically provisioned and used by applications within the cluster.

VI. Computational Analysis

A. Parallel/Distributed Computation in Kubernetes

The ability of Kubernetes to handle parallel processing is critical for the analysis and decision-making processes in precision agriculture [7]. By distributing workloads across multiple computing nodes, Kubernetes ensures that data from sensors embedded in a drip irrigation system can be processed concurrently, thus reducing the time to action [8]. This is especially pertinent when considering the need for real-time data processing to make immediate adjustments to irrigation levels, nutrient delivery, and more.

The distributed nature of Kubernetes also allows for redundancy and fail-over mechanisms, enhancing the reliability and uptime of the monitoring system [12]. In an agricultural context, this means that even if one node fails, others can take over the workload without disrupting the monitoring and control of the drip irrigation system. Furthermore, Kubernetes' distributed computation capabilities are complemented by its robust scheduling algorithms, which efficiently allocate computational resources based on workload demands [18]. This is crucial in scenarios where sensor data might show significant temporal and spatial variability, requiring dynamic adjustments to computational resources.

This approach, both parallel and distributed, streamlines operations and simultaneously opens avenues for integrating advanced predictive analytics and machine learning models. These integrations have the potential to further enhance irrigation practices [1].

B. High-performance Computation in Edge Computing

High-performance computation in edge computing refers to the capability of processing large amounts of data rapidly and efficiently at the edge of the network, closest to where data is generated or where actions are to be taken [5] [9]. In the context of agriculture, and specifically drip irrigation systems, this means analyzing sensor data on-site to make immediate decisions regarding water distribution, nutrient levels, and crop health.

The edge computing model provides a significant advantage in terms of response times compared to traditional centralized cloud computing. By minimizing the distance data must travel for processing, edge computing reduces latency, a crucial factor for time-sensitive agricultural operations where delays can affect crop yields.

Another aspect of high-performance computation at the edge is the ability to deploy machine learning models locally [1]. This means that predictive analytics can be performed in real-time, enabling the system to anticipate irrigation needs based on patterns and trends identified directly from the data streams. This also entails robust data handling and storage capabilities, ensuring that data integrity is maintained even in remote or rural locations with intermittent connectivity. This resilience is critical in agricultural settings, where environmental factors can be unpredictable and harsh [5].

For a drip irrigation system, high-performance computation at the edge facilitated by technologies such as Kubernetes can translate into a highly adaptable and responsive infrastructure. It allows for the continuous adjustment of irrigation protocols based on real-time data, leading to water conservation, improved crop health, and ultimately, higher agricultural productivity.

VII. System Monitoring and Visualization

The monitoring and Visualization system of Kubernetes-managed irrigation systems is fundamental to ensuring optimal performance and resource efficiency, as well as anomaly detection [19]. Utilizing tools like Prometheus and Grafana for monitoring, and analyzing parallel/distributed computation and high-performance computation in edge computing, can significantly enhance the management of such systems. Kubernetes provides the tools necessary for dynamic resource allocation and scheduling, while Prometheus offers the monitoring capabilities needed to ensure the system operates at peak efficiency [20]. Together, they create a responsive, efficient, and reliable infrastructure that can adapt to the demands of modern precision agriculture.

A. Prometheus

Prometheus is a monitoring and alerting toolkit that operates as open-source software, widely used in the context of cloud computing and microservices [9]. SoundCloud originally developed it in 2012, and subsequently, in 2016, the Cloud Native Computing Foundation adopted it as part of its suite of tools. Prometheus is configured within the Kubernetes ecosystem to continuously collect data from the various nodes and pods, and also uses Alertmanager that handles all the alerting mechanisms for Prometheus metrics [9]. It scrapes metrics exposed by Kubernetes and application code at specified intervals, storing them in a time-series database. This setup allows for live metrics stored in a time series database, providing insights into the performance and health of each component [14].

Prometheus operates on a multi-dimensional data model, where it gathers time series data at regular intervals from various sources and stores it on local disk drives, labeled by metric names and key/value pairings. This method bears a strong resemblance to the way Kubernetes handles infrastructure metadata. Prometheus uses an HTTP-based pull model to accumulate real-time metrics in a time series database and employs PromQL for versatile querying and instant alerting, shown in 5. Unlike traditional blackbox monitoring systems like Nagios or Icinga, which are limited to basic administrative tasks [21],

Prometheus adopts a whitebox monitoring strategy that provides detailed insights into the condition of microservices. The Prometheus ecosystem consists of several key components: 1) the Prometheus server, employs a pull model from Kubernetes nodes for metric collection via HTTP jobs and compiles and logs strictly numerical time series;

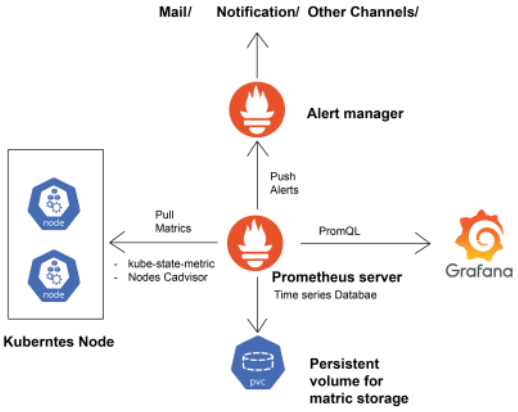


Fig. 5: Prometheus Alert System

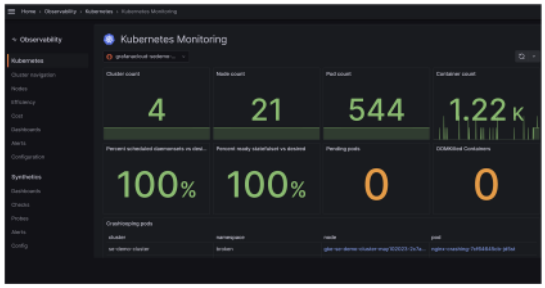
2) Client Libraries tailored to the programming language of the application; 3) Alertmanager, which oversees alert notifications, grouping, and suppression; 4) Exporters, which broadcast existing metrics from third-party systems; and 5) PromQL and Grafana, a highly adaptable query language designed for querying metrics within the Prometheus dashboard. Prometheus utilizes its query language not just within its user interface but also assumes a vital function in Grafana. This involvement aids in visualizing and analyzing metrics, as exemplified in 6, where we can witness Kubernetes monitoring resources by observing the node points, as illustrated in figures 6a and 6b, Additionally, its effectiveness is demonstrated in figure 6c .

The configuration can be optimized using service discovery mechanisms in Kubernetes, ensuring that Prometheus is aware of the dynamic nature of containerized environments.

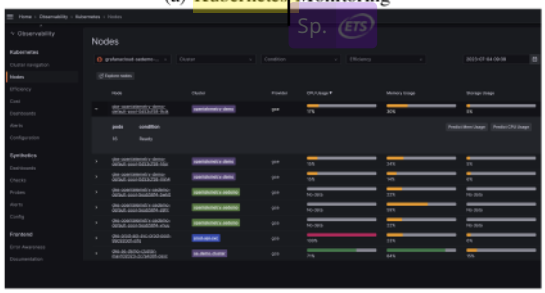
VIII. Discussion and Conclusion

The exploration of Kubernetes-managed edge computing for drip irrigation systems presents a compelling case for the integration of advanced computing techniques in precision agriculture. This paper has demonstrated the architecture of Kubernetes in managing distributed nodes, which facilitates the real-time processing of data crucial for the precise delivery of water to crops. Through the management of sensor data via Kubernetes pods, coupled with the containerization of irrigation monitoring components, the system achieves a high degree of modularity and scalability. Monitoring and computational analysis have been identified as the linchpins of this system. The use of Prometheus and Grafana for real-time data visualization and alerting has demonstrated that such a system can not only react to immediate conditions but also anticipate future needs through predictive analytics.

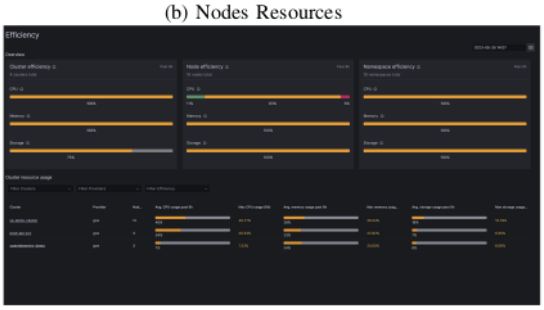
Challenges, however, remain. The implementation complexity, the initial costs of setup, and the need for technical



(a) Kubernetes Monitoring



(b) Nodes Resources



(c) Kuberntes resource usage efficiency

Fig. 6: Prometheus and Grafana

expertise cannot be overlooked. Moreover, the adoption of such systems on a wide scale brings to light concerns about cyber security, data privacy, and the necessity for robust, fault-tolerant designs that can withstand the unpredictable nature of agricultural environments.

In conclusion, the marriage of Kubernetes-managed edge computing with precision irrigation holds great promise for the future of farming. It offers a blueprint for a future where agriculture is smarter, more responsive, and more attuned to the needs of the environment. With continued innovation and refinement, such systems are poised to revolutionize the way we approach one of humanity's oldest practices.

References

[1] M. Wakchaure, B. Patle, and A. Mahindrakar, "Application of ai techniques and robotics in agriculture: A review," *Artificial Intelligence in the Life Sciences*, vol. 3, p. 100057, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667318523000016

[2] P. Yang, L. Wu, M. Cheng, J. Fan, S. Li, H. Wang, and L. Qian, "Review on drip irrigation: Impact on crop yield, quality, and water productivity in china," *Water*, vol. 15, no. 9, 2023. [Online]. Available: https://www.mdpi.com/2073-4441/15/9/1733

[3] E. Kristiani, C.-T. Yang, Y. T. Wang, and C.-Y. Huang, "Implementation of an edge computing architecture using openstack and kubernetes," in *Information Science and Applications 2018: ICISA 2018*. Springer, 2019, pp. 675–685.

[4] F. J. Ferrández-Pastor, J. M. García-Chamizo, M. Nieto-Hidalgo, and J. Mora-Martínez, "Precision agriculture design method using a distributed computing architecture on internet of things context," *Sensors*, vol. 18, no. 6, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/6/1731

[5] E. A. Abioye, M. S. Z. Abidin, M. S. A. Mahmud, S. Buyamin, M. K. I. AbdRahman, A. O. Otuoze, M. S. A. Ramli, and O. D. Ijike, "Iot-based monitoring and data-driven modelling of drip irrigation system for mustard leaf cultivation experiment," *Information Processing in Agriculture*, vol. 8, no. 2, pp. 270–283, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214317320301864

[6] O. Debauche, S. Mahmoudi, P. Manneback, and F. Lebeau, "Cloud and distributed architectures for data management in agriculture 4.0 : Review and future trends," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 7494–7514, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157821002664

[7] N. L. Zhou, Y. Georgiou, M. Pospieszny, L. Zhong, H. Zhou, C. Niethammer, B. Pejak, O. Marko, and D. Hoppe, "Container orchestration on hpc systems through kubernetes," *Journal of Cloud Computing*, vol. 10, no. 1, p. 16, 2021. [Online]. Available: https://doi.org/10.1186/s13677-021-00231-z

[8] P. Kayal, "Kubernetes in fog computing: Feasibility demonstration, limitations and improvement scope : Invited paper," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020, pp. 1–6.

[9] N. Sukhija and E. Bautista, "Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus," in *2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2019, pp. 257–262.

[10] M. O'Grady, D. Langton, and G. O'Hare, "Edge computing: A tractable model for smart agriculture?" *Artificial Intelligence in Agriculture*, vol. 3, pp. 42–51, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2589721719300339

[11] N. M. Z. Hashim, S. Mazlan, M. Z. Abd Aziz, A. SALLEH, A. Ja'afar, and N. Mohamad, "Agriculture monitoring system: A study," *Jurnal Teknologi*, vol. 77, 10 2015.

[12] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying microservice based applications with kubernetes: Experiments and lessons learned," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018, pp. 970–973.

[13] N. Kapočius, "Performance studies of kubernetes network solutions," in *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, 2020, pp. 1–6.

[14] G. Carcassi, J. Breen, L. Bryant, R. W. Gardner, S. Mckee, and C. Weaver, "Slate: Monitoring distributed kubernetes clusters," in *Practice and Experience in Advanced Research Computing*, ser. PEARC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 19–25. [Online]. Available: https://doi.org/10.1145/3311790.3401777

[15] O. Mart, C. Negru, F. Pop, and A. Castiglione, "Observability in kubernetes cluster: Automatic anomalies detection using prometheus," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2020, pp. 565–570.

[16] T. Kubernetes, "Kubernetes," *Kubernetes. Retrieved May*, vol. 24, p. 2019, 2019.

[17] T. Menouer, "Kcss: Kubernetes container scheduling strategy," *The Journal of Supercomputing*, vol. 77, no. 5, pp. 4267–4293, 2021. [Online]. Available: https://doi.org/10.1007/s11227-020-03427-3

[18] I. Konev, I. Nikiforov, and S. Ustinov, "Algorithm for containers' persistent volumes auto-scaling in kubernetes," in *2022 31st Conference of Open Innovations Association (FRUCT)*, 2022, pp. 89–95.

[19] O. Mart, C. Negru, F. Pop, and A. Castiglione, "Observability in kubernetes cluster: Automatic anomalies detection using prometheus," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2020, pp. 565–570.

[20] V. Sharma, "Managing multi-cloud deployments on kubernetes with istio, prometheus and grafana," in *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, 2022, pp. 525–529.

[21] S. Ballestrero, F. Brasolin, D. Fazio, C. Gament, C. J. Lee, D. A. Scannicchio, and M. S. Twomey, "Integrated monitoring of the atlas online computing farm," *Journal of Physics: Conference Series*, vol. 898, no. 9, p. 092008, oct 2017. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/898/9/092008

Final paper

ORIGINALITY REPORT

9%	5%	6%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Nitin Sukhija, Elizabeth Bautista. "Towards a Framework for Monitoring and Analyzing High Performance Computing Environments Using Kubernetes and Prometheus", 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 2019 Publication	1%
2	Yu-Wei Chan, Halim Fathoni, Hao-Yi Yen, Chao-Tung Yang. "Implementation of a Cluster-based Heterogeneous Edge Computing System for Resource Monitoring and Performance Evaluation", IEEE Access, 2022 Publication	1%
3	Submitted to Independent University Bangladesh Student Paper	1%

4	Md. Farhadul Islam, Fardin Bin Rahman, Sarah Zabeen, Md. Azharul Islam et al. "RNN Variants vs Transformer Variants: Uncertainty in Text Classification with Monte Carlo Dropout", 2022 25th International Conference on Computer and Information Technology (ICCIT), 2022 Publication	1 %
5	Submitted to Toronto Business College Student Paper	1 %
6	www.mdpi.com Internet Source	1 %
7	Dennis Giovani Balreira, Thiago da Silva Araújo, Fabio Petrillo. "Visualizing Kubernetes Distributed Systems: An Exploratory Study", 2023 IEEE Working Conference on Software Visualization (VISSOFT), 2023 Publication	<1 %
8	github.com Internet Source	<1 %
9	stackoverflow.com Internet Source	<1 %
10	kubernetes.io Internet Source	<1 %
11	D. Scano, A. Giorgetti, F. Paolucci, A. Sgambelluri, J. Chammanara, J. Rothman, M.	<1 %

	Albado, E. Marx, S. Ahearne, F. Cugini. "Enabling P4 Network Telemetry in Edge Micro Data Centers with Kubernetes Orchestration", IEEE Access, 2023 Publication	
12	Submitted to Letterkenny Institute of Technology Student Paper	<1 %
13	Submitted to Fiji National University Student Paper	<1 %
14	Submitted to Lovely Professional University Student Paper	<1 %
15	blogs.sap.com Internet Source	<1 %
16	en.wikipedia.org Internet Source	<1 %
17	Paridhika Kayal. "Kubernetes in Fog Computing: Feasibility Demonstration, Limitations and Improvement Scope : Invited Paper", 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), 2020 Publication	<1 %
18	file.techscience.com Internet Source	<1 %
19	Ritesh Kumar Singh, Rafael Berkvens, Maarten Weyn. "AgriFusion: An Architecture	<1 %

For IoT And Emerging Technologies Based On A Precision Agriculture Survey", IEEE Access, 2021
Publication

20	www.cisco.com Internet Source	<1%
21	www.internationaljournalssrg.org Internet Source	<1%
22	Pei Yang, Lifeng Wu, Minghui Cheng, Junliang Fan, Sien Li, Haidong Wang, Long Qian. "Review on Drip Irrigation: Impact on Crop Yield, Quality, and Water Productivity in China", Water, 2023 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Final paper

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sentence Cap. Review the rules for capitalization.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sentence Cap. Review the rules for capitalization.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sentence Cap. Review the rules for capitalization.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Confused You have used either an imprecise word or an incorrect word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to remove this article.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to remove this article.

Confused You have used either an imprecise word or an incorrect word.

Prep. You may be using the wrong preposition.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Missing "," Review the rules for using punctuation marks.

Proofread This part of the sentence contains an error or misspelling that makes your meaning unclear.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Proofread This part of the sentence contains an error or misspelling that makes your meaning unclear.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to remove this article.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Missing "," Review the rules for using punctuation marks.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

P/V You have used the passive voice in this sentence. You may want to revise it using the active voice.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Run-on This sentence may be a run-on sentence.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Confused You have used either an imprecise word or an incorrect word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Missing "," Review the rules for using punctuation marks.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.