Implementation of Continuous Diagnosis for Secondary Systems in Operating Nuclear Power Plants

DongYun Cho [a,b*], You-Rak Choi[b], Tae-Jin Park[b]
[a]*Korea National University of Science and Technology (UST), Daejeon, South Korea*
[b]*Nuclear System Integrity Sensing & Diagnosis Division, Korea Atomic Energy Research Institute (KAERI), Daejeon, South Korea*
[*]*Corresponding author: dycho@kaeri.re.kr*

***Keywords :* continuous diagnosis, cloud-native, nuclear power plant**

1. Introduction

Ensuring the safe operation of nuclear power plants (NPPs) remains one of the most important considerations in South Korea. With ongoing efforts to enhance safety measures of these facilities, many fault diagnosis techniques are emerging [1]. In particular, wireless sensor-based fault diagnosis is becoming increasingly popular in the monitoring of the secondary systems in NPPS due to its wide-area coverage. However, relying solely on wireless sensors is not sufficient for long-term diagnostics. To address this issue, this proposes the implementation of continuous diagnosis for secondary systems in operating NPPs, leveraging a cloud-native solution to enhance consistency and reliability.

2. Continuous Monitoring System

Current diagnosis systems in NPPS require on-site personnel to quickly respond to potential system failures. This dependence on human intervention makes it difficult to respond to emerging problems in a timely manner. However, these concerns can be addressed by integrating a continuous monitoring system that operates autonomously in the background. In this way, the system is always on alert for any failures, enhancing its capability for real-time data processing. Real-time data processing is a crucial component in diagnosis to promptly detect faults in secondary systems and ensure efficient operation of NPPs.

2.1 Cloud Native Solution

Cloud-native solutions enable dynamic provisioning of hardware resources and allow automation of tasks such as initiating and managing applications through programmable infrastructure.

In addition to automation, cloud-native solutions provide auto-scaling capabilities [2], which adjust the number of applications to fluctuating application demands without human intervention. Moreover, in case of system failure, cloud-native solutions can also automatically restart the corresponding application,minimizing downtime for the proposed diagnosis system. For this reason, cloud-native solutions are applied to the diagnosis system.

2.2 Cluster Design/Schematics

The continuous diagnosis system is designed to work within a cloud-native solution as shown in Fig. 1. The system begins by collecting wireless sensor data. The data is analyzed using Fast Fourier Transform (FFT), and is sent to Transmission Control Protocol (TCP) servers. Each server has different responsibilities and these are labeled as “RECV_SV”, “Process”, and “GUI_SV.”

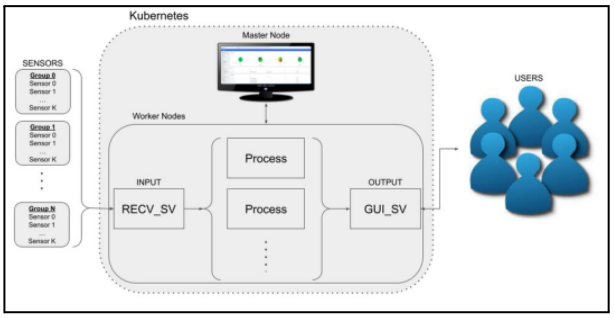

Fig. 1. Schematics of the proposed continuous diagnosis for secondary systems in operating NPPs. Dotted line is a boundary for Kubernetes.

The “RECV_SV” server collects wireless sensor data. The “Process” server then takes the processed data and inputs it into a machine learning model to calculate the leakage probability. Finally, both the sensor data and the calculated probabilities are delivered to the “GUI_SV” server. The “GUI_SV” server is a web-server that operates based on communication between the server and users. When a user accesses the user interface (UI), real-time sensor data in the frequency domain is displayed in a graph. The fault diagnosis system is mounted on Kubernetes, which is a widely used cloud-native solution.

Kubernetes is based on clusters consisting of master nodes and worker nodes. The master node continuously monitors the services of the worker nodes, while the worker nodes are responsible for running diagnosis

applications. When one of the applications fails, the worker node notifies the master node, and the master node deploys a new copy of the application to the available worker nodes. This disaster recovery feature allows for continuous diagnosis.

3. Diagnosis System Implementation

The overall system within Kubernetes is developed using the Python programming language, and each TCP server is containerized using Docker to streamline deployment and management. Three mini-PCs with different specifications were prepared to ensure optimal performance. The most robust mini PC was selected as the master node. A private registry for Docker containers is established on the master node to efficiently distribute container applications to worker nodes.

The Kubernetes UI in Fig. 2 provides administrators with a comprehensive overview of the system's performance. For example, a pie chart displays the status of each component within the Kubernetes cluster. Deployment plays the most important role for the proposed system, as it is responsible for the generation and management of a predefined number of pods by the master node. Each pod contains the diagnosis system application and is distributed to the available worker nodes.

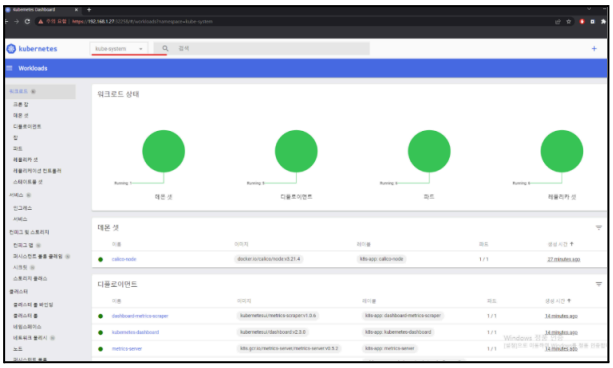

Fig. 2. Kubernetes UI. It is easy to manage and create necessary application deployment.

After the pod is created, the Kubernetes service components are initialized. This configuration allows pods containing different TCP servers to communicate internally or externally as needed. For example, the "Process" server performs internal data transfers, while the "RECV_SV" and "GUI_SV" servers facilitate both types of transfers.

To verify the successful integration of the diagnosis system with Kubernetes, the functionality of the "GUI_SV" server is tested using the collected wireless sensor data. The proposed system was successfully implemented as the graph was drawn on the "GUI_SV" server. Fig. 3 shows the image of the graph.

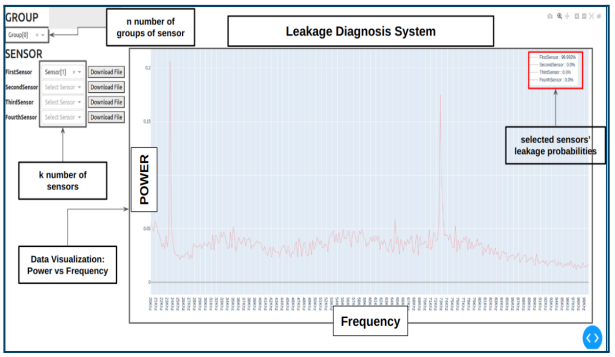

Fig. 3. "GUI_SV" UI displays a real-time graph of sensor data with leakage probabilities on the top corner.

Currently, the proposed system is being tested under multi-user traffic scenarios. This is to prevent data loss when multiple users try to observe graphs from the same sensor.

4. Conclusions

In summary, integrating Kuberentes with wireless sensor-based methods improves system consistency and reliability when monitoring secondary systems in operating NPPs. Through testing with collected wireless sensor data, the proposed diagnosis system demonstrates its ability to streamline the fault diagnosis process and guarantee real-time data processing. This system provides a promising solution to improve the safety of NPPs. Future work will focus on further testing and optimization of the proposed system for different operating conditions of NPPs.

REFERENCES

[1] Korea Institute of S&T Evaluation and Planning (KISTEP), KISTEP Report, 11-1721000-000597-01, 2011.
[2] G. Kulkarni, P. Khatawkar, and J. Gambhir, Cloud Computing-Platform as Service, International Journal of Engineering and Advanced Technology (IJEAT), Vol-1, Issue-2, 2011.

ACKNOWLEDGEMENT

We acknowledge the Korean government, Ministry of Science and ICT, for support (No. RS-2022-00144000).