Workflow-based Recovery for Network Congestion in Kubernetes Environment.

Ta Phuong Bac, YoungHan Kim*,
Soongsil University

bactp@dcn.ssu.ac.kr, younghak@ssu.ac.kr*

Abstract

Kubernetes is one of the most widely recognized and convenient open sources for deploying and orchestrating containerized applications. However, Kubernetes enforces resource limitations and employs scheduling mechanisms that only consider CPU and memory but also overlook network considerations. As a result, network fair sharing among pods and applications cannot be maintained. It may lead to congestion problems when workloads critically depend on network resources. Therefore, in this paper, we propose workflow-based recovery solutions for addressing network congestion problems in Kubernetes environments. Workflow-based recovery works incorporated with machine learning-based network congestion predictions to enable proactive recovery from predicted results. The remediation process is firmed into workflow and automatically performs recovering action following defined logic. In this study, two basic recovery workflows, P2P_congested and N2N_congested, are presented to address network congestion at the service level and node level.

Ⅰ. INTRODUCTION

Utilizing containerization empowers IT development teams to operate fast, deploy highly efficient software, and manage operations at an unprecedented scale. Simultaneously, managing large-scale applications deployed in the cloud poses a significant challenge for many organizations due to the escalating complexity of enterprise computing requirements [1-2]. To tackle these challenges, sophisticated automation and orchestration platforms have been developed.

Currently, Kubernetes [3] stands out as a widely recognized and convenient tool for deploying and orchestrating container systems composed of diverse components. Deploying a service platform involves the sequential configuration of each service, demanding careful coordination of internal and external resources.

In Kubernetes, Pods serve as the smallest deployable entities that can be created and managed on the physical nodes. A pod is essentially a collection of one or more containers (e.g., Docker containers) operating on the same node, sharing underlying resources such as CPU, memory, and network. Whether within a public or private cloud infrastructure, the shared utilization of resources among various workloads enhances the efficiency of the physical infrastructure, ultimately reducing operational costs. However, multiple workloads on the same server can lead to performance interference due to competition for shared resources.

Kubernetes incorporates resource limitation and resource-aware scheduling for CPU and memory, enabling the Kubernetes scheduler to identify nodes with sufficient available resources for pod placement. However, Kubernetes does not manage the network setup for communication between pods. It adopts a plug-in architecture for networking utilizing a variety of projects (such as *Calico* [4] and *Cilium* [5]) supporting a wide variety of networking needs. Consequently, it lacks built-in support for network management solutions. Without network considerations, we cannot maintain a fair network share among pods with the same priority. Ensuring network quality, allocating bandwidth to specific pod networks, and addressing storage I/O-intensive workloads cannot be guaranteed. As a result, the workload may suffer sub-optimal performance when contending for the network, especially leading to congestion in clusters.

To address network congestion issues, we propose a workflow-based recovery for network congestion in Kubernetes environments. In this approach, a workflow-based recovery module is developed in integration with other modules, such as machine learning-based network congestion predictions and root cause analysis to predict congestion situations of the system and then predict the root cause of the problem and send it to the recovery module. This study automatically performs the recovery process by declaring workflow incorporation with other system APIs to perform recovery action. Istio [6] service mesh is leveraged to collect more service metrics, and it also does recovery action at the service level and is flexible in system upgrades to large-scale. The details of system architecture will be presented in the following section.

Ⅱ. PROPOSAL SYSTEM PROCEDURE

In this section, we discuss the main architecture of workflow recovery approaches proposed for handling network congestion in the Kubernetes environment. The overview of the workflow-based recovery system procedure is illustrated in Figure 1.

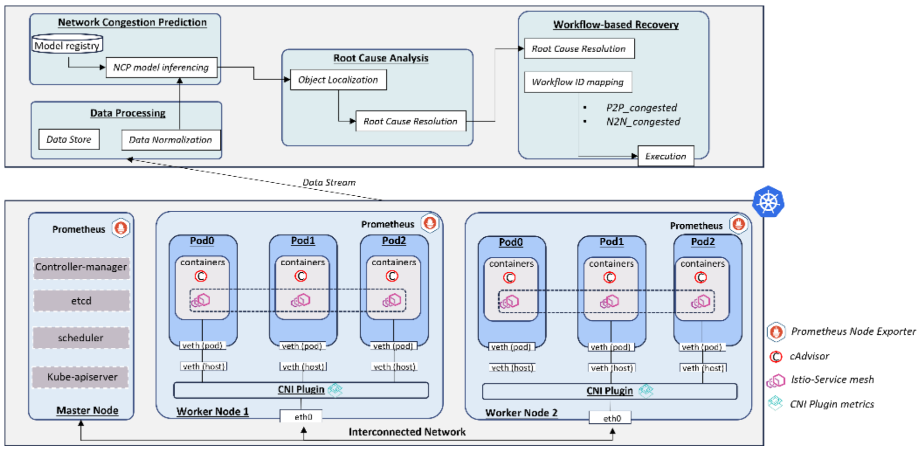

Figure 1 Workflow-based recovery: system procedure

As shown in Figure 1, Kubernetes-based infrastructure is monitored by several monitoring agents to get network metrics. In this proposal, we used Istio to collect network metrics data of service level from containerized applications. Istio also makes it possible to handle the system on a large scale, and further requirements, such as traffic limiting and access control, will be integrated into a workflow-based recovery module to define and execute the remediation action to avoid network congestion. Network metrics from cloud infrastructure, then streaming to Network Congestion Prediction for predicting congestion status.

The main part of our proposal is a workflow-based recovery module, which consumes results from the network congestion predictions module to determine which action needs to be performed to recover the system from congestion status to normal status and guarantee network quality for service. The logic of workflow processing is declarative, making it flexible and efficient to adapt to system changes and upgrades. In this initial proposal, we consider two main recovery workflows, *P2P_congested* and *N2N_congested*, for the workflow mapping part. In particular, *P2P_congested* responsibilities for handling network congested from pod to pod level, in which one pod can consume higher traffic and affect network quality in the cluster and service quality. Istio service mesh is combined in this workflow's type to handle congestion status by several recovery actions such as traffic limiting and breaking by the circuit breaker. In *N2N_congested*, node network congestion is considered, in which network communication from node to node is the root cause, such as delay in the network interface of the Linux host. *N2N_congested* will trigger kubeAPI to inform network congestion problems in nodes (worker nodes), and then the congested node can be replaced or restarted to mitigate the congested situation. Also, consider migrating/scaling services from a congested node to another node to ensure service quality.

IV. CONCLUSION

In this paper, we present a new approach for recovering network congestion in Kubernetes environments. We provide system procedures with two types of recovery workflow, *P2P_congested* and *N2N_congested*, for handling network congestion at the pod and node levels. Istio service mesh is leveraged to get network metrics and perform remediation action in congested situations. With this approach, the recovery process can automatically be executed and incorporated with prediction modules. Recovery's logic can be defined in a declarative way, making it flexible with system changes and easy to configure. In the future, we will research efficiency recovery's logic for two proposed workflows and create other workflows to cover system availability in network congestion problems fully, then implement the proposed solutions.

ACKNOWLEDGMENT

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. 2020-0-00946, Development of Fast and Automatic Service recovery and Transition software in Hybrid Cloud Environment) and (No.2022-0- 01015, Development of Candidate Element Technology for Intelligent 6G Mobile Core Network).

REFERENCES

[1] Ogbole, M., Ogbole, E., & Olagesin, A. (2021). Cloud Systems and Applications: A Review. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 142-149

[2] Zhang, Teng, and Lihong Du. "Research on IT Operation and Maintenance and Management and Maintenance Methods in Cloud Computing Environment." 2022 International Conference on Creative Industry and Knowledge Economy (CIKE 2022). Atlantis Press, 2022.

[3] Kubernetes, https://kubernetes.io/. Last Accessed 05 Jan, 2024.

[4] What is project Calico, https://www.tigera.io/project-calico/. Last Accessed 05 Jan, 2024

[5] What is Cilium, https://cilium.io/get-started/. Last Accessed 05 Jan, 2024

[6] The Istio service mesh, https://istio.io/latest/about/service-mesh/. Last Accessed 05 Jan, 2024.