Performance Analysis of Kubernetes-based Data Distribution Service

Eojin Kim*, Yongseong Kim*, Young Choi*, Byungkwon Song*, Hyery Shin**, DeukWoo Kim**

*Seokyeong Univ., **Korea Institute of Civil Engineering and Building Technology

eojin1030@skuniv.ac.kr, wiz@skuniv.ac.kr, cy@skuniv.ac.kr, bksong@skuniv.ac.kr, hyeryshin@kict.re.kr, deukwookim@kict.re.kr

Kubernetes 기반 Data Distribution Service 성능 분석

김어진*, 김용성*, 최영*, 송병권*, 신혜리**, 김덕우**
*서경대학교, **한국건설기술연구원

Abstract

Given the increasing interest in utilizing container virtualization for IoT application deployment and management, this study aims to optimize DDS application architecture within a Kubernetes framework. To achieve this goal, this study evaluated the performance by implementing containerized versions of OpenDDS, an open-source DDS solution, and RTI Connext DDS, another DDS implementation. Furthermore, this study evaluated the performance of the conventional Kubernetes container network interface plug-in, Kube-Router, and Flannel. The results of this study serve as a benchmark to help users build the optimal CNI environment for DDS application execution within the Kubernetes framework and select the optimal DDS solution.

Ⅰ. Introduction

Cloud computing and virtualization technologies are indispensable components of contemporary information technology (IT) infrastructure. Notably, container-based virtualization and orchestration platforms, such as Kubernetes, are instrumental in driving these advancements [1]. In addition, data distribution and real-time data exchange are a fundamental requirement in numerous industries and technological domains. Data distribution services (DDS) are acknowledged as a crucial standard technology for ensuring reliable data communication [2]. In the context of this framework, the current study focuses on implementing DDS applications within a cloud-based infrastructure. Additionally, we conducted a comprehensive performance analysis of various DDS implementations, notably, OpenDDS and RTI DDS. Specific performance metrics, such as throughput and latency, were employed to assess the effectiveness of these implementations under the designated scenarios. Moreover, we investigated and compared the performances of various container network interface (CNI) plug-ins.

Ⅱ. Method

A. Backgrounds on underlying technologies

DDS is a standard communication protocol and application programming interface (API) that has been defined by the Object Management Group (OMG). It employs a publisher–subscriber model for data exchange among disparate systems. Within this architectural framework, the publisher generates and disseminates data via DDS, while the subscriber selects and receives only the necessary data through DDS [3].

Kubernetes automates the deployment, scaling, and governance of containerized applications, thereby simplifying the construction and management of intricate software systems. CNI plug-ins are responsible for the management of network connections among the containers [4].

B. Experimental Setup and Configurations

The experimental setup used in this study employed a Kubernetes environment installed on three virtual machines (VMs) within the VMware framework. Each VM played a distinct role; one functioned as a master node, while the remaining two were designated as worker nodes. One worker node was assigned the role of the DDS publisher, with the other serving as a DDS subscriber. An additional VM, positioned external to the Kubernetes environment, was employed to store the data received by the subscriber in a PostgreSQL database. To facilitate the independent operation of the two DDS environments—RTI DDS and OpenDDS—on the worker nodes, the namespace within the cluster was logically segmented. Each VM was equipped with the requisite software and modules, including Kube-Router and Flannel, to establish a Kubernetes network environment. Table 1 provides the specifications of the master and worker nodes.

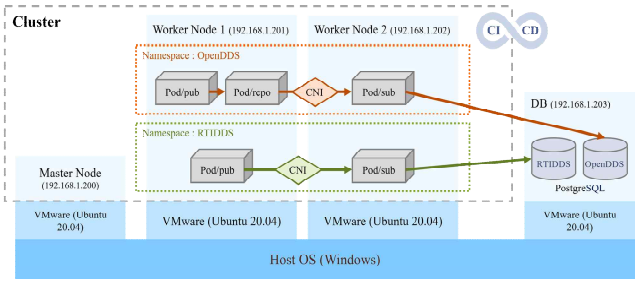

Fig. 1. Analysis Environment implementation

Specification	Master	Worker1	Worker2
Hardware			
CPU	12th Gen Intel® Core™ i7-12700		
Memory (RAM)	16 GB		
Software			
Operating System	Ubuntu 20.04 LTS (64bit)		
Orchestration	KubeAdmin v1.24.14		
Container	Containerd 1.7.3		
CNI Plugin	Flannel 0.22.0 Kube-Router 1.6.0		
DDS	RTI Connext DDS v6.0.1 OpenDDS v3.24.1		

Table 1. Specification of the master and worker node

C. Performance Evaluation

The performance analysis was conducted when the data size was 10K. Figure 2-(a) illustrates the throughput results for both RTIDDS and OpenDDS. In the scenario utilizing the Kube-Router CNI plug-in, RTIDDS reached a throughput of 4024.1 Mbps, while OpenDDS achieved a more substantial throughput of 6032.6 Mbps. In a parallel test utilizing the Flannel CNI plug-in, RTIDDS registered a throughput of 3185.4 Mbps, with OpenDDS recording a throughput of 4823.1 Mbps. In both instances, OpenDDS demonstrated superior throughput performance.

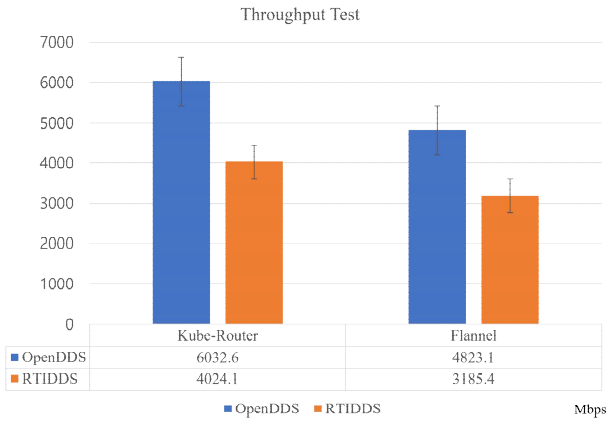

Fig. 2-(a). Throughput test results

Figure 2-(b) represents the latency assessment outcomes for the RTIDDS and OpenDDS. Employing the Kube-Router CNI plug-in, RTIDDS exhibited a latency of 166 μs, in contrast to OpenDDS, which showed a reduced latency of 119 μs. Subsequently, under the conditions facilitated by the Flannel plug-in, RTIDDS demonstrated a latency of 151 μs, while OpenDDS posted a marginally shorter latency of 123 μs. Once again, OpenDDS displayed superior performance in terms of latency.

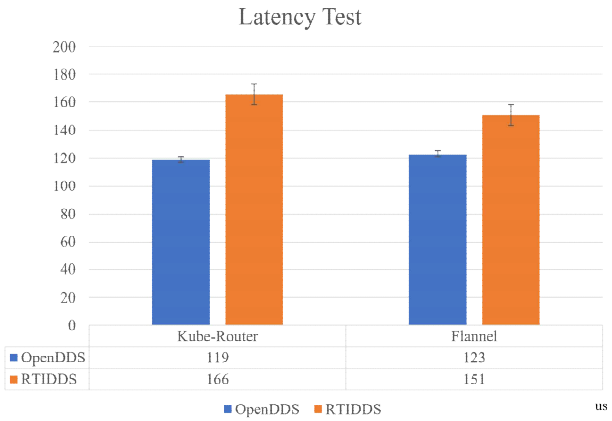

Fig. 2-(b). Latency test results

Ⅲ. Conclusion

This study offers a comprehensive performance evaluation and optimization framework for systems that integrate container-based applications with DDS technology and highlights why OpenDDS has emerged as a more economical and efficient alternative to RTIDDS, particularly in terms of throughput and latency metrics. In conclusion, this study is anticipated to significantly advance the efficient deployment and operation of applications. Moreover, it aims to contribute to the success of future initiatives involving distributed systems and cloud computing environments.

ACKNOWLEDGMENT

This work is supported by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant RS-2023-00244769, Development of a data framework for integrating building energy datasets and applications to accelerate carbon neutrality in the building sector)

REFERENCES

[1] P. Sujatha and R. Vijaya, “Cloud Computing using Open-Source Solution - Open-Stack,” Int. J. Appl. Eng. Res., vol. 9, no. 27 Special Issue, pp. 9636- 9639, 2014.

[2] Je-man Park, “A Fast DDS Data Transport Protocol for Large-Scale CPS Middleware”, Ph.D. thesis, Dept. Electronic & Computer Eng, Hanyang University Graduate School, pp. 10-13, 2014

[3] Ki-jung Kwon, "DDS-based middleware framework for large network-centric systems," Ph.D. thesis, Dept. Computer Eng., Chungnam National University Graduate School, pp. 36- 40, 2012.

[4] S. Novianti and A. Basuki, "The Performance Analysis of Container Networking Interface Plugins in Kubernetes," 6th Int. Conf. on Sustainable Information Engineering and Technology, 2021.