Adapting Kubernetes to IIoT and Industry 4.0 protocols - An initial performance analysis*

Ahmad Alamoush, Holger Eichelberger
{alamoush,eichelberger}@sse.uni-hildesheim.de
University of Hildesheim, Hildesheim, Germany

Abstract

Kubernetes (K8s) is one of the most frequently used container orchestration tools offering, as it offers a rich set of functions to manage containerized applications, it is customizable and extensible. Container virtualization of applications and their orchestration on heterogeneous resources including edge devices is a recent trend in Industrial Internet of Things (IIoT)/Industry 4.0, where K8s is also applied. However, IIoT/Industry 4.0 is a domain with high standardization requirements. Besides equipment standards, e.g., for electrical control cabinets, there are also demands to standardize network protocols, data formats or information models. Such standards can foster interoperability and reduce complexity or deployment/integration costs. Here, the proprietary communication protocol of K8s and similar orchestrators can be an obstacle for adoption.

To explore this situation from an interoperability and integration perspective, we present in this paper an approach to replace the communication protocol of K8s without modifying its code base. We show by an experiment that applying our approach with three current forms of IIoT communication, namely Message Queuing Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), and Asset Administration Shell (AAS), does not significantly affect the validity and the performance of K8s.

1 Introduction

Virtualization and containerization technologies show high potential in the field of IIoT, but a widespread use of containerized applications requires tools to manage containers, scale them up or down, to perform rollout of updates, and more. Container orchestration tools were developed to support those tasks. One of the most popular container orchestrators is Kubernetes (K8s)[1], which is available as open-source software. According to a survey report by CNCF's Cloud Native Landscape[2], there are more than 109 tools to manage containers, but 89% are using variants of K8s. Further, a CNCF microsurvey[3] shows that the most popular edge use case is manufacturing/industrial IoT, with 54% out of 117 respondents, 75% using K8S for edge applications.

Interoperability is one of the main pillars of IIoT/Industry 4.0 besides connectivity, cybersecurity, and artificial intelligence [5]. Interoperability is the capability of heterogeneous entities, e.g., IIoT devices, to connect and exchange meaningful information [5]. A lack of interoperability as it currently exists, e.g., among IIoT field devices of different vendors, but also on higher levels of IIoT systems, significantly increases complexity and costs for IIoT deployment/integration [1]. To overcome this, ongoing efforts aim at standardizing data formats and information models, e.g., Open Platform Communications United Architecture (OPC UA) or Asset Administration Shells. Some argue that also protocols such as MQTT or AMQP, or – more extreme – even networking protocols like TCP/IP shall be standardized.

Orchestrators such as K8s form a cluster of at least one master and multiple worker nodes executing the containerized applications. By default, K8s uses a proprietary REST-based protocol for the communication between master and worker nodes. Such protocols conflict with the aforementioned standardization ideas and can lead to adoption issues. Here, openness can help, e.g., K8s can be extended through the Container Network Interface (CNI), which establishes network connections among containers. However, there is no support to change the master-worker communication, e.g., to a standardized protocol.

In this context, we ask a what-if question: *If IIoT/Industry 4.0 communication protocols would be standardized, would it still be possible to utilize existing container orchestrators like K8s? Subsequently, can we adapt container orchestrators to such a setup and can this lead to a performance impact?* In this paper, we focus on K8s as the most frequently used container orchestrator and present an approach to replace the communication protocol without modifying the K8S code base. As IIoT communication is yet not standardized, we demonstrate our approach with

*IIP-Ecosphere is partially supported by the German Federal Ministry of Economic Affairs and Climate Action (BMWK) under grant 01MK20006D

[1] https://kubernetes.io

[2] https://www.cncf.io/wp-content/uploads/2020/08/CNCF_Survey_Report.pdf

[3] https://www.cncf.io/wp-content/uploads/2021/05/KubernetesEdge_Survey_Report_2021_v2.pdf

currently popular protocols in the domain. For binary payload transport, we rely on the Message Queuing Telemetry Transport (MQTT) and the Advanced Message Queuing Protocol (AMQP), for Industry 4.0 information models we use the currently trending Asset Administration Shell (AAS) [2]. These forms of communication are also requested for future IIoT platforms as identified in [7] and in the IIP-Ecosphere project[4], which defines the scope of our work.

In [6], Novianti et al. conducted performance evaluation of CNI plugins. They performed several instances of benchmark tests for TCP and UDP and collected CPU and memory usage for each CNI plugin. For benchmarking, the authors used the k8s-bench-suite[5] benchmark tool with some modified parameters. In [4], Kapočius analyzed the performance of 4 CNCF recommended K8s CNI plugins, namely Flannel, Calico (VXLAN, IPIP, and pure IP), Kube-router, and Weave. The author compared the performance in terms of latency and average TCP throughput. Although related, CNI does not support changing the underlying communication protocol as mentioned before. To our best knowledge, there is no work on replacing the communication protocol for K8s.

The paper is structured as follows: In Section 2, we present the approach that we used to replace the communication protocol of K8s. Then, we discuss the experiment that we conducted and its results in Section 3. Finally, in Section 4, we conclude the paper and give an outlook on future work.

2 Approach

In this section, we introduce the utilized communication protocols, discuss the architecture of K8s as well as our approach on modifying the communication protocol without changing K8s.

MQTT is a transmission data protocol suitable for machine to machine communication, Wireless Sensor Networks, and Internet of Things applications [3]. AMQP is a similar protocol that delivers multi-function messages to distributed receiver queues [3]. In contrast to such payload/channel-based protocols, AAS is a virtual and active representation of Industry 4.0 components in a network. An AAS stores digital information of an asset and makes asset related services available through an API with access control [2].

As shown in Figure 1, the architecture for a K8s cluster consists of a master and several worker nodes. The master node manages and controls the K8s cluster, while the worker nodes execute the containerized applications. The default communication between the master node and worker nodes is shown by the black arrow lines in Figure 1, where components in the workers request updates regarding their status in the cluster from the master node.

[4] https://www.iip-ecosphere.eu/
[5] https://github.com/InfraBuilder/k8s-bench-suite

To replace the underlying communication protocol of K8s, we follow a proxy approach including one proxy type for the master node and one for workers. Each proxy is equipped with the three protocols mentioned above, from which we can select the actual protocol at startup. Figure 1 illustrates the communication steps: (1) the worker proxy receives a request from a K8s worker component. (2) the worker proxy translates the request to the active protocol and passes it on to the master node proxy. (3) the master node proxy processes the request using the K8s Java Client library, which, in turn, calls the K8s REST API. (4) the master node proxy translates the received REST response to the active protocol and sends it back to the originating worker proxy. (5) the worker proxy passes the response on to the originating K8s component. It is worth mentioning that the K8s protocol involves various formats, e.g., protobuf, and that the proxy design shares common functionality to ease the integration of different IIoT protocols.

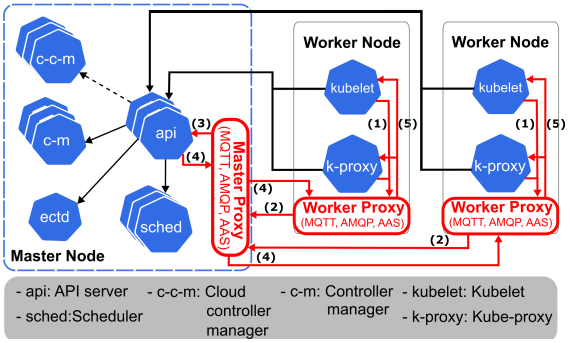

Figure 1: Basic K8s cluster (proxies in red).

3 Experiment and Results

Protocol translations to realize proxies may lead to performance impacts. In this section, we present an experiment to gain insights into the performance of our proxies. Then, we discuss the results.

Experimental environment: We used three VMware virtual machines (VM), each VM has 4 virtual CPU cores, 16 GB of RAM and Linux Ubuntu Server 20.04 installed. The VMs are connected by local network with an average bandwidth of 17,7 Gbits/sec. We established a K8s cluster with one VM acting as master node and the other VMs as worker nodes. We used Java 13 to develop the proxies. We used RabbitMQ (v5.15.0) and Eclipse Paho (v1.2.5) as protocol clients for AMQP and MQTTv5 as well as Apache QPID-J (v8.0.2) and HiveMq (v2020.4) as brokers, respectively. Further, we integrated Eclipse BaSyx (v1.0.1) as open source AAS middleware. As orchestrator, we used K8s (v1.23.3).

Subject and variants: We set up four variants, the original K8s and one variant per proxied protocol. Since K8s does not ship with a (required) CNI, we created a simple networking setup with a bridge for

each node into a single subnet as well as respective port mappings. Alternatives could have been known CNIs like Calico or Flannel, which may introduce unexpected side effects on the experiment.

Experimental procedure: As basis we use the K8s test-infra[6], a testing infrastructure of K8s including 7118 tests targeting different aspects of a K8s cluster such as "Conformance", "Linux only", "NetworkPolicy", or "Performance". We selected the 348 conformance tests to validate the functionality of K8s in combination with our proxies. This includes tests on API communication, storage, networking, authentication, or scheduling. We use the total execution time of all conformance tests as a coarse-grained indicator on the impact of our proxies on the overall functionality of K8s. For a more detailed insight, we use the only performance test in test-infra. This test creates a container for each worker, starts the containers, which generate (as a simple application) some logs. The logs are requested and validated by the test. Here, we focus on the individual execution time.

In both cases, we followed the same procedure for a selected variant/test aspect: First, we start[7] master and work nodes proxies. Then, we join the worker nodes into the cluster. After that, we run either the conformance tests once or the performance test once. Next, we remove the worker nodes from the cluster. Finally, we stop[7] master and worker proxies. For the performance test, repeat these steps 100 times to gather also fluctuations. Experiment execution and result collection are automated through Bash scripts[8].

Results: The conformance tests were successfully executed for each variant with an execution time of about 89 minutes and variations of around 6 minutes. We can attribute the variances to the consecutive execution of tests to random fluctuations that occur even in the original K8s, where the execution of one test suddenly delays the start of the next test. As we just aim at a coarse grained insight, infra-test was not realized as a performance experiment in mind and we tried to avoid modifications of K8s/infra-test, we believe that a variance of 5% is acceptable. Further, the proxying also increases the resource usage, as we believe in acceptable manner. E.g., the average CPU load for the AAS proxy is increased by 2.67%/1.35% (master/worker), the memory usage by 4.06%/2.43% and the network traffic by 1.99%/0.51%.

Table 1 shows the statistical summary for the execution time of the repetitions of the performance test for each variant. The proxies only cause minor deviations compared to the original K8s.

Table 1: "Performance" test result in milliseconds

K8s variant	***Min***	***Max***	***Avg***	***Stddev***
Original K8s	5,083	5,163	5,108	0,018
AAS	5,086	5,159	5,108	0,017
AMQP	5,085	5,160	5,108	0,016
MQTTv5	5,085	5,152	5,107	0,017

4 Conclusion

The use of containerized applications in IIoT demands management capabilities as, e.g., provided by container orchestrators. Kubernetes offers many capabilities, but proprietary communication protocols (K8s and other orchestrators) can cause adoption issues when standardization in the IIoT domain progresses.

In this paper, we presented an approach to enable IIoT protocol compliance by replacing the communication protocol of K8s without modifying its code base. We conducted an experiment with four variants (original K8s, MQTT, AMQP, and AAS). Although executed on a small scale cluster, a potential limitation, the results indicate that our approach does not cause major effects on typical functionality or performance of K8s. Based on this, can imagine that protocol openness (without proxy detours) may be a future interoperability feature of orchestrators. Further, we believe that a similar approach can be applied to other orchestrators such as Docker Swarm or Mesos.

In future, we plan to evaluate the scalability with different cluster sizes. Further, aim at integrating to other relevant protocols, e.g., OPC UA.

[6] `https://github.com/kubernetes/test-infra`

[7] Skipped for the original K8S variant.

[8] `https://doi.org/10.5281/zenodo.7158281`

References

[1] E. Sisinni et al. "Industrial Internet of Things: Challenges, Opportunities, and Directions". In: *IEEE Trans Industr Inform* 14.11 (2018), pp. 4724–4734.

[2] N. Chilwant and M. S. Kulkarni. "Open Asset Administration Shell for Industrial Systems". In: *Manufacturing Letters* 20 (2019), pp. 15–21.

[3] N. Q. Uy and V. H. Nam. "A comparison of AMQP and MQTT protocols for Internet of Things". In: *Conf. on Information and Computer Sc.* 2019, pp. 292–297.

[4] N. Kapočius. "Performance Studies of Kubernetes Network Solutions". In: *Conf. of Electrical, Electronic and Information Sc.* 2020, pp. 1–6.

[5] A. Hazra et al. "A Comprehensive Survey on Interoperability for IIoT: Taxonomy, Standards, and Future Directions". In: *ACM Comput. Surv.* 55.1 (2021).

[6] S. Novianti and A. Basuki. "The Performance Analysis of Container Networking Interface Plugins in Kubernetes". In: *Intl. Conf on Sustainable Information Eng. and Tech.* 2021, pp. 231–234.

[7] H. Eichelberger, H. Stichweh, and C. Sauer. "Requirements for an AI-enabled Industry 4.0 Platform - Integrating Industrial and Scientific View". In: *SOFTENG'22.* 2022.