A Design of Serverless Computing Service for Edge Clouds

2021 International Conference on Information and Communication Technology Convergence (ICTC) | 978-1-6654-2383-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICTC52510.2021.9621162

Jaeeun Cho
School of Electronic Engineering
Soongsil University
Seoul, Korea
jayj@dcn.ssu.ac.kr

Younghan Kim
School of Electronic Engineering
Soongsil University
Seoul, Korea
younghak@ssu.ac.kr

***Abstract*— The method of Serverless Computing at the Edge is drawing attention because of the efficiency of using resources. This paper presents a design of Serverless Computing for Edge Clouds, based on an open-source project. One of the methods of Serverless at the Edge is that centralized architecture with a Serverless Platform in the control plane cluster. However, in a centralized architecture, if the resources or the status of a particular Edge is not good, the control plane cluster must detect anomalies and redeploy the Functions considering the environment of other Edges. It will take time to select and deploy to the appropriate Edge. The proposed architecture introduces cross-monitoring which imports monitoring metrics directly from a nearby Edge. Grouping Edge sites as a Zone and cross-monitoring each other in the Zone. So that, Serverless computing-based Functions can operate quickly when adjacent Edges are in poor condition. As a result, the neighboring Edges of the Zone can back up with each other.**

Keywords—Serverless, Autoscaling, Edge computing, Kubernetes

I. INTRODUCTION

The demand for Serverless computing services, which is a cloud computing execution model, is rapid growth. Because it makes developers build applications and services without thinking about the underlying servers[1]. Therefore, Serverless can reduce the time for development. Also, Serverless is event-driven. For that reason, it can save computing resources and costs by scaling only when necessary.

Edge computing introduces to supplement privacy concerns and high latency caused by the centralization of cloud computing[2]. However, while traditional cloud-native approaches to resource management, service orchestration, and scheduling have reached a high level of maturity, they are challenged when dealing with key characteristics of distributed edge systems: compute device heterogeneity, geographic dispersion, and the resulting operational complexity. Serverless computing has emerged as a compelling model to manage the complexity of such systems, by decoupling the underlying infrastructure and scaling mechanisms from applications[3]. Also, Serverless' scale-to-zero property is suitable for IoT use cases with intermittent applications. Therefore the interest in Serverless Edge computing is uprising for dealing with this kind of problem and managing Edge infrastructure. But, Serverless has some issues to be met before entering Edge computing, so Serverless Edge Computing is still developing.

Edge nodes add the additional layer to the IT infrastructure required by enterprises and service providers in the on-premise and cloud data center architecture. So, admins must manage the workloads at edge levels dynamic and automated way like the same way for on-premise or cloud. Additionally, the whole architecture contains different types of computing hardware resources and software applications. Kubernetes comes to the rescue as it characterizes due to infrastructure agnostic capability. It can manage a diverse set of workloads from different compute resources seamlessly[4].

This paper presents a design of Serverless Computing Service for an Edge Cloud, based on an open-source project. Overcome device heterogeneity using Kubernetes, then introduce a Serverless that uses one of the triggers is cross-monitoring.

II. RELATED WORK

Baresi and Mendon[5] proposed a Serverless Edge computing platform based on OpenWhisk. They design the entire system architecture for Serverless Edge computing and implement a load balancer that considers distributed infrastructure.

Reference [3] presents a container scheduling system that enables Serverless Platforms' efficient use of edge infrastructures. That paper introduces four new scheduling constraints to favor nodes based on:

1) *proximity to data storage nodes*
2) *proximity to the container registry*
3) *available compute capabilities*
4) *edge/cloud locality*

Reference [6] analyzed the advantages of bringing Serverless to the Edge and potential obstacles for such accomplishments.

- Advantages: (1) offering pure pay-per-use,(2) mitigating always-on resource provisioning, (3) consistency with the event-driven nature of IoT, (4) being easy to parallelize stateless functions, (5) fulfilling storage isolation, (6) fine-grained scalability for resource-limited edge devices, (7) moving inline with bursty workloads and CPU-bound applications, (8) existing custom-made platforms, and (9) viability of cloud to edge integration.
- Obstacles: (1) cold startup provoking long latencies, (2) impractical cost-efficiency designed for the cloud, (3) unsuitability for continuous workloads and edge AI applications along with no support for GPU considerations (4) lack of location- and-energy-awareness, (5) unconsidered distributed networking, (6)

inefficient data shipping,(7) brief resource specification, (8) reliability and fault tolerance concerns, (9) possibility of provoking resource inefficiency, (10)security concerns, (11) immature function triggering and (12) lack of simulation tools.

III. BACKGROUND

A. Deploying Kubernetes for Edge

Reference [4] presents 3 approaches to deploying Kubernetes for Edge.

- Whole Clusters at the Edge: The whole Kubernetes cluster is deployed within Edge nodes. This option is ideal for a requirement where Edge node might have fewer capacity resources or single server machine and do not want to consume more resources for control planes and nodes. K3s is the reference architecture suited for this type of solution.
- Control plane resides in the cloud and managing the Edge nodes containing containers and resources: This approach is referred from KubeEdge architecture. This architecture allows support for different hardware resources present at the Edge and enables optimization in Edge resource utilization.
- Hierarchical cloud Edge in which Virtual Kubelet is used: Virtual Kubelets reside in the cloud part. It basically contains the abstract of nodes and pods deployed at the Edge. Virtual Kubelets gets a supervisory control for Edge nodes, containing containers. Using Virtual Kubelets enables flexibility in resource consumption for Edge-based architecture.

B. HPA

The HPA(Horizontal Pod Autoscaler) automatically scales the number of Pods in a replication controller, deployment, replica set, or stateful set based on observed CPU utilization[7]. However, it is limited in the number of metrics available and cannot reduce the number of PODs to zero. As shown in Fig. 1, if you want to use other metrics you can use Custom Metrics, which needs a suitable Metrics Adapter each, but only one metric can use at a time.

C. KEDA

KEDA(Kubernetes-based Event-Driven Autoscaler) is an open-source component for Kubernetes that supports event-driven autoscaling. The key roles of KEDA are the scaling agent and Kubernetes metric server. KEDA works alongside HPA and can extend functionality without overwriting or duplication. It can extend your PODs by event queue-based as well as CPU and memory-based, and also reduce the number of PODs to zero in the absence of events. As shown in Fig. 2, it has a wide range of scalers that can both detect if a deployment should be activated or deactivated, and feed custom metrics for a specific event source. No separate Metrics Adapter is required, and multiple metrics can be used simultaneously[8].

IV. PROPOSED ARCHITECTURE

To deploying the Edge Computing architecture using Kubernetes, envisioned placing an entire cluster at the Edge, the first method introduced in the Related Work. This architecture is more convenient if you have a hybrid cloud architecture that you are using. Private clouds can replace with Edge, or Edge can be linked together to existing hybrid clouds.

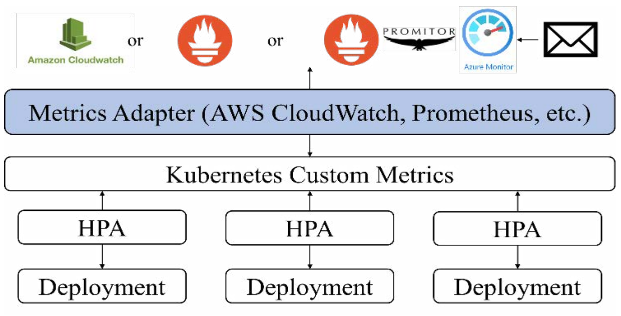

Fig. 1. Autoscaling using Custom Metrics.

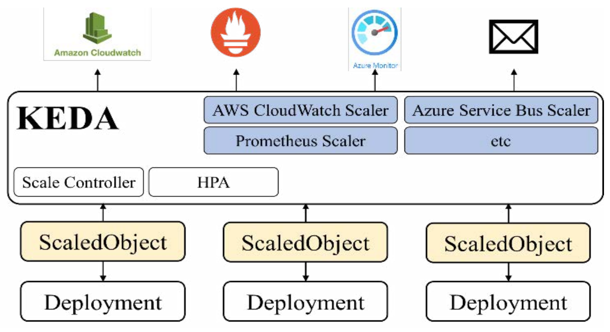

Fig. 2. Autoscaling with KEDA.

Fig. 3 shows a centralized architecture with a Serverless Platform in the control plane cluster. This architecture can use when monitoring the resources in each cluster and Functions are deployed accordingly or when availability is critical.

However, in a centralized architecture, if the network connectivity or the status of a particular Edge is not good, the control plane cluster must detect anomalies and redeploy the Functions considering the environment of other Edges. It will take time to select and deploy to the appropriate Edge. The role of scheduling policies becomes major because the resource availability of the Edge must check to determine the amount of deployment to the Edge. The more things to

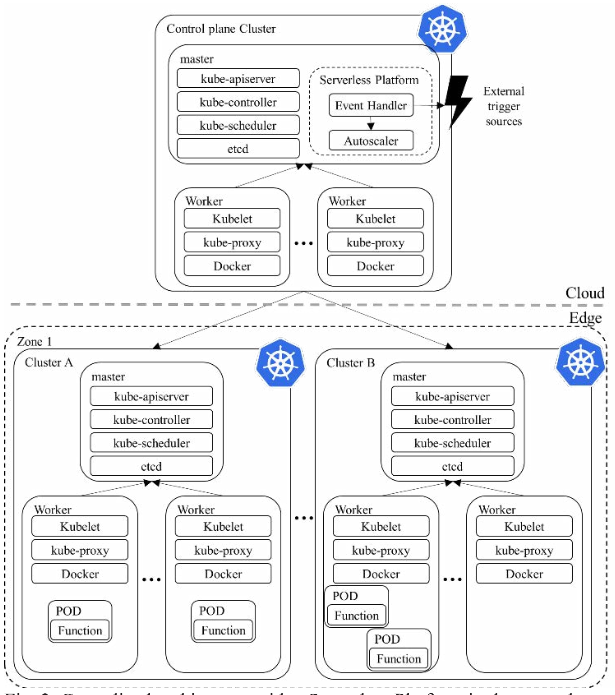

Fig. 3. Centralized architecture with a Serverless Platform in the control plane cluster.

consider, such as Edge resources, network connectivity, and distance from users, the more complex the scheduling algorithm becomes.

Therefore, this paper proposes an architecture that links one of the multiple event sources used as triggers to a monitoring metric at a nearby Edge. The monitoring metrics for nearby Edge define as cross-monitoring. Fig. 4 shows the proposed architecture. This architecture allows responding more quickly because import metrics directly from a nearby Edge to the Edge without going through a control plane cluster.

Fig. 5 is a flowchart that is showing how scaling works. Suppose the same Functions deploy in both clusters A and B. Each Function has two triggers, which are Event and cross-monitoring. From cluster A's side, A's Event Handler is polling events periodically. If an Event occurs or B's status is not good, Function scaling is happening. Repeat event polling. Likewise, from cluster B's side, B's Event Handler watches A's status is good or not. As a result, adjacent Edges in the Zone can be backed up by each other.

The plan is that implement the architecture using KEDA and Prometheus. In KEDA, containers don't use HTTP routing to consume events. Instead, they directly consume events from the source. In addition, multiple event sources can use as well as a single event source. And Prometheus Federation allows a Prometheus server to scrape selected time series from another Prometheus server[9]. Therefore, KEDA and Prometheus will fit well with the proposed architecture.

V. CONCLUSIONS

This paper introduces the advantages and problems of Serverless Edge Computing. Also, it proposes an architecture that introduces cross-monitoring that can react quickly when adjacent Edges are in poor condition. This architecture allows responding more quickly because import metrics directly from a nearby Edge to the Edge without going through a control plane cluster. The plan is that implement the architecture using KEDA and Prometheus. Then, specify and experiment with the types of metrics that can be used as cross-monitoring.

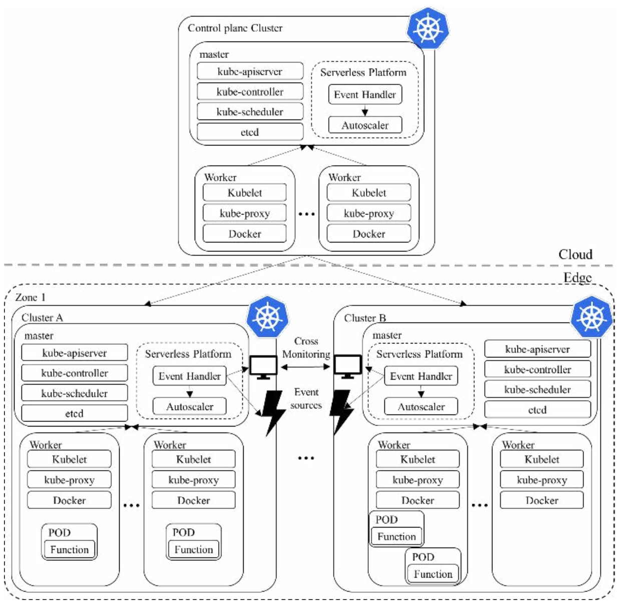

Fig. 4. Proposed architecture.

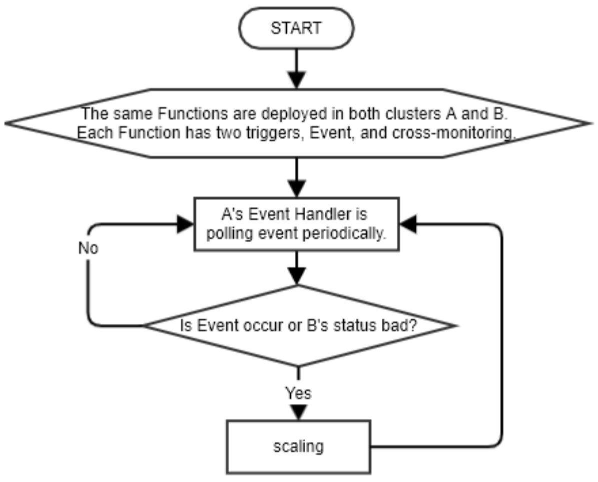
START
The same Functions are deployed in both clusters A and B. Each Function has two triggers, Event, and cross-monitoring.
A's Event Handler is polling event periodically.
No
Is Event occur or B's status bad?
Yes
scaling

Fig. 5. Scaling process flowchart.

ACKNOWLEDGMENT

"This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2021-2017-0-01633) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation)"

REFERENCES

[1] "Serverless Computing – Amazon Web Services," 2021. [Online]. Available: https://aws.amazon.com/serverless/?nc1=h_ls. [Accessed: 06-Aug-2021]

[2] A. Palade, A. Kazmi, and S. Clarke, "An evaluation of open source serverless computing frameworks support at the Edge," in *Proceedings - 2019 IEEE World Congress on Services, SERVICES 2019*, 2019, pp. 206–211.

[3] T. Rausch, A. Rashed, and S. Dustdar, "Optimized container scheduling for data-intensive serverless edge computing," *Future Generation Computer Systems*, vol. 114, pp. 259–271, Jan. 2021.

[4] Calsoft, "A deep-dive on Kubernetes for Edge."

[5] L. Baresi and D. Filgueira Mendonca, "Towards a Serverless Platform for Edge Computing," *2019 IEEE International Conference on Fog Computing (ICFC)*, pp. 1–10, Jun. 2019.

[6] M. S. Aslanpour *et al.*, "Serverless Edge Computing: Vision and Challenges," in *ACSW '21: 2021 Australasian Computer Science Week Multiconference*, 2021.

[7] "Horizontal Pod Autoscaler | Kubernetes," 26-Jul-2021. [Online]. Available: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/. [Accessed: 02-Aug-2021]

[8] "KEDA | KEDA Concepts," 2021. [Online]. Available: https://keda.sh/docs/2.3/concepts/. [Accessed: 02-Aug-2021]

[9] "Federation | Prometheus," 2020. [Online]. Available: https://prometheus.io/docs/prometheus/latest/federation/. [Accessed: 12-Aug-2021]