

Using Cross Monitoring to scale on the Edge

Iosif Koen
Fabian Zeiher

Serverless computing





Serverless computing

- Cloud computing execution model
- Developers build applications easier
- Event-driven
 - Small functions are executed on demand
 - Save:
 - Computing resources
 - Cost

Edge computing





Edge computing

- Distributed computing paradigm/template
- Brings computation and data storage closer to data sources
 - Like IoT devices, sensors and other Edge devices
- Main Goals:
 - Faster data processing because of **reduced distance**
 - **Reduced network congestion** (Improved performance)
 - Local data processing for **better security**

A Design of Serverless Computing Service for Edge Clouds

Jaeeun Cho

School of Electronic
Engineering, Soongsil
University

Younghan Kim

School of Electronic
Engineering, Soongsil
University

A Design of Serverless Computing Service for Edge Clouds

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

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)



Our focus today: *Scale-up on the edge*

- Processing power on edge devices is usually limited
 - High load at unexpected times
- ➡ Distribute workload across edge devices



Sharing workload across edge devices

Serverless approach

- Encapsulate functions inside containers
- All functions are available in all edge nodes (initially scaled to 0)
- Workload can be moved to other nodes

Centralized scaling solution





Centralized scaling solution

Tasks of the Central control plane cluster:

- **Monitor load** on all edge devices
- **Monitor available resources** on all edge devices
- Select appropriate edge devices for scale-up



Issues with a centralized solution

- Reliability
 - Connectivity to cloud could be unreliable
- Complexity
 - Node selection for scale-up can become quite complex
- Performance
 - High load on the control plane

Decentralized Scaling Solution

The background features a series of dark gray, three-dimensional geometric planes that create a sense of depth and perspective, receding towards the right. Two trapezoidal shapes are highlighted: a light green one positioned higher and further back, and a blue one positioned lower and closer to the foreground.



Decentralised scaling solution

Utilize cross-monitoring between edge nodes:

- **Monitor load** on neighboring edge devices
- **Know available resources** on own device
- Take workload from overloaded neighboring nodes



Technical Implementation

KEDA

- Can scale up or down, according to some external events

Prometheus

- Monitoring tool
- Scrape other Prometheus instances



Code snippet

```
triggers:
- type: prometheus
  metadata:
    # Required fields:
    serverAddress: http://<neighbor-node-host>:9090
    query: sum(rate(http_requests_total{deployment="my-deployment"}[2m]))
    threshold: '100.50'
    activationThreshold: '5.5'
    # Optional fields:
    namespace: example-namespace
    customHeaders: X-Client-Id=cid,X-Tenant-Id=tid,X-Organization-Id=oid
    ignoreNullValues: true
    unsafeSsl: "true"
```





Reflection

- *Missing*: How is input/output data securely transported between nodes?
- A lot of additional components are running on the edge devices, performance?
 - Not suitable for low powered IoT devices
- **Quite theoretical**



Future Work

- Prototype an actual implementation of this concept
- Test and evaluate this solution in a real world example
- Create a framework for this architecture



Related Research / Applications

t.b.d.

If you live your life on the edge
monitor your neighbors.

