Proposal of an edge serverless computing platform with an asynchronous communication model

LI YANZHI[1] MIDORI Sugaya[1]

Abstract: In recent years, the number of IoT devices that access to the networks is growing rapidly. The outrageous number of network connections and large amount of data transmission generated by these devices bring new challenges to cloud computing. Edge computing has received much attention as one of the solutions to the data transmission bottleneck in cloud computing networks. Currently, edge computing, represented by content delivery networks, has achieved great success in static data distribution and has started to provide computing services in edge computing nodes in a serverless computing paradigm. However, the synchronous computing model used in existing Edge serverless computing platforms has the problem of difficulty in running time-consuming computational procedures such as AI inference. To address this problem, we propose an edge serverless computing platform based on an asynchronous computing model. This study decouples the computational process from the communication process through asynchronous computing, which increases the flexibility of computational resource scheduling, and in turn permits more efficient use of edge computing resources. Also, the serverless computing model and the global computing library introduced in this study can be used to hide hardware differences, which allows different hardware to be deployed in the edge site to improve the computing performance. Finally, we discuss the approach to deploying computational programs on this platform, which validates the possibility of using asynchronous computing techniques to deploy computational programs on this platform.

Keywords: Edge computing, Serverless, Function as a Service, IoT, Heterogeneous computing

1. Introduction

In recent years, the number of devices connected to the network has increased with the spread of IoT devices and mobile phones. The large number of connections to cloud servers increases the pressure on cloud servers to handle connections and data transfers. In order to solve the bottleneck of connection and data transfer on the cloud server side, edge computing has been attracting more and more attention[1]. Contents Delivery Network is a kind of edge computing, and many service providers already offer services. In addition to the static content caching that comes standard with CDNs, companies such as Cloudflare[2] are providing additional computing services on edge nodes in Serverless Computing[4]. However, these computing services are usually based on an event-driven synchronous compute model, which limits the maximum compute time allowed to prevent a single client from taking up compute resources for a long time, leading to service outages. Such limitations prevent Serverless Computing platforms from running time-consuming computational procedures such as AI inference.

This research aims to allow computations of longer duration to be performed in edge computing in the form of serverless computing. To achieve this objective, we designed a serverless computing platform based on an asynchronous computing model. The remaining part of this research is: In Section 2, we introduced the related work, and in Section 3, we pointed out the problem in previous work and gave our proposal. In Section 4, we introduce the design of our platform, and in Section 5, we give two examples of using this platform. Finally, Section 6 gives this research's conclusion and feature work.

2. Related Works

As one kind of edge computing, CDNs have successfully reduced the traffic and connection loads on cloud servers. CDN providers such as Cloudflare are now trying to provide edge computing services using their infrastructure in the model on the left side of Fig 1. In this model, when a client (Requester) issues a request, the context is forwarded synchronously to the backend (Worker) for processing. This model performs well for short-time tasks. However, when dealing with long execution time tasks, the unstable network connection may cause failures[5]. Also, the unpredictable execution time may break the load balancing across nodes.

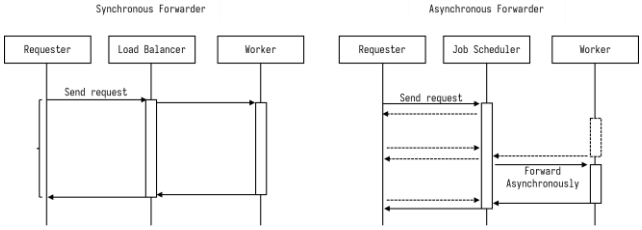

Figure 1 Synchronous (left) and Asynchronous model(right)

Workload managers such as Slrum[3] provide a solution to deal with tasks with the long execution times by using an asynchronous forwarding model, as shown on the right side of Fig. 1. In this model, after a client issues a request, the context is asynchronously forwarded to the backend for processing. The client no longer needs to keep a network connection active to wait for the result. Furthermore, it gives extra flexibility for management tasks with long execution times.

1 Shibaura Institute of Technology
3-7-5 Toyosu, Koto-ku, Tokyo 135-8548, Japan

3. Problem and Proposal

The existing serverless edge computing platforms usually use the synchronous forwarding model. This model has problems with long-time computing, such as AI inference. In this research, we propose a serverless edge computing platform with an asynchronous computing model to satisfy the needs of computing tasks with a long execution time. Our proposal suggests using an asynchronous computing interface as Slrum and providing serverless computing functions as an existing edge computing system to hide the hardware differences.

4. System Design

The Fig 2 shows the overview of our proposed system. There are four prominent roles in this system: The Requester is the role that requests the use of compactional resources, and the Worker is the role that offers the compactional resources. The Developer is the role that will develop the computing program called by the Requester and run on the Worker. The Manager is the central role of this system. It has three main tasks: 1) Handle the job contents sent by the Requester, schedule and forward the job contents to the Worker asynchronously. 2) Offers extra functions for computing, such as data storage for computing contents, including input and output. 3) Distribute the program developed by the Developer to the Worker and neighbour Edge Node. 4) Fetching data from neighbor Edge Node when Requester requests.

4.1 System Objects

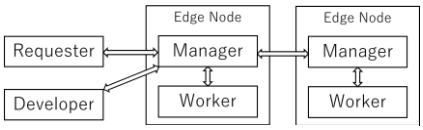

Figure 2 System Overview

We suggest these objects be implemented for creating a serverless computing platform with a permission system.

1. User: The Developer's information, such as password.
2. Project: The organization of several Developers they can create functions together
3. Function: The script that will be executed on the Worker
4. Job: A running function instance issued by the Requester.
5. Worker: The workers connected to this node.
6. BLOB Data: Binary data that can be used as input or output of the function also can be used to share data between Jobs.
7. KV: The Key-Value store namespace for each Function, Project and User.

Also, we create some rules for the object federation between Edge Nodes. We suggested that User, Project, and Function will be distributed to federation nodes activity when modified, and the Data and Job will only be Fetched from federation nodes when needed. The Worker will only serve resources for the Manager connected, so the Worker object will never be transferred. For KV storage, we suggested having Global KV distributed to other nodes activity and Local KV only available in the same Edge Node.

4.2 API Design

We designed an object-oriented HTTP API for this system. The API design was shown in Table 1. The classes are described in Section 4.1, and the object_id is an ID based on SnowFlake ID.

GET /:class	Get object list
POST /:class	Create new object
GET /:class/:object_id	Get object attributes
POST /:class/:object_id	Set object attribute
DELETE /:class/:object_id	Delete an object
POST /:class/:object_id/:method	Call instance methods

Table 1 API design

5. Application Example

To discuss if this system can achieve our goal, we discuss some application examples here.

5.1 AI inference

One example is offloading AI inference computing to the Edge. The fine-tuning AI model can be saved as blob data. Since the data ID may be changed, we suggested saving the data ID in the KV store with the Developer decided key. The function will first fetch the model blob data ID from the KV store, then fetch the model blob by data ID, run the inference, and return the result to the Manager.

5.2 Data Pre-Processing

Another example is running pre-processing before the data is uploaded to the cloud. For example, when trying to detect some object from the camera, Requester can upload the picture to Edge and call the function that will detect the object from the picture. The function will be executed starting to detect the object. Once the object has been detected, the function will start to upload the information to the cloud, and the picture without the information can just be deleted.

6. Conclusion

This study proposed a serverless edge computing that can handle computes that request for a long execution time. The proposed system uses an asynchronous computing interface to satisfy the long-duration execution and uses a serverless computing model to hide the difference in hardware. Finally, we give two examples of using this platform for running long-duration computing and federation with cloud computing. Future research is considered in 1) Implement this platform as a distribution system for each edge node. 2) Discussion about the library function that the system will offer. 3) The proposed system is a large-scale system which still needs further discussion about quantitatively evaluate this system.

Acknowledgements This work was supported by JST, CREST Grant Number JPMJCR19K1, Japan.

References

[1] Ju Ren, Deyu Zhang, Shiwen He, Yaoxue Zhang, and Tao Li. 2019. A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms: Transparent Computing, Mobile Edge Computing, Fog Computing, and Cloudlet. ACM Comput. Surv. 52, 6, Article 125 (November 2020), 36 pages. https://doi.org/10.1145/3362031

[2] Cloudflare Workers. https://workers.cloudflare.com/.

[3] SlurmMD. https://www.schedmd.com/

[4] Y. Li, Y. Lin, Y. Wang, K. Ye and C. Xu, "Serverless Computing: State-of-the-Art, Challenges and Opportunities," in IEEE Transactions on Services Computing, vol. 16, no. 2, pp. 1522-1539, 1 March-April 2023, doi: 10.1109/TSC.2022.3166553.

[5] T. Lynn, P. Rosati, A. Lejeune and V. Emeakaroha, "A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms," 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong Kong, China, 2017, pp. 162-169, doi: 10.1109/CloudCom.2017.15.