Design of Connected Data Lake System based on Micro Cloud Storage

Sun Park
School of Electrical Engineernig and Computer Science, GIST
Gwangju, South KOREA
sunpark@smartx.kr

Byungrae Cha
School of Electrical Engineernig and Computer Science, GIST
Gwangju, South KOREA
brcha@gist.ac.kr

Yunseok Cha
Research Institute
GenoTech Corporation
Gwangju, South KOREA
dxcha@naver.com

Jaeyun Mo
Research Institute
GenoTech Corporation
Gwangju, South KOREA
ahwo22@gmail.com

Suhui Oh
Research Institute
GenoTech Corporation
Gwangju, South KOREA
genotech2012@daum.net

Jongwon Kim
School of Electrical Engineernig and Computer Science, GIST
Gwangju, South KOREA
jongwon@gist.ac.kr

***Abstract*—The increasing use of ICBM (IoT & Cloud & Big Data & Mobile), which is the basis of ICT diversity, has led to an increase in various large amounts of data. This increase in data is transforming the society into a data-oriented society. Edge Cloud is increasing to manage increasing data efficiently. However, it is difficult to efficiently integrate distributed data between Edge Cloud and Cloud. This paper proposes a Connected DataLake system that can solve these problems.**

Keywords—Cloud, micro cloud storage, data lake, message queue, stream processing

I. Introduction

In a world where technology and life change differently from one day to another, which revolutionary change is named Industry 4.0, smart factory, and manufacturing innovation 3.0 strategy. On the other hand, the maturity of ICT (Information & Communication Technology) convergence has been diversified into autonomous vehicles, smart medical, health care, drones, robots, smart culture & life. With the increasing use of ICBM, a large variety of ICT data is increasing. The increase of this data is making society become a data-oriented society. Increasing volumes of data and a data-oriented society are increasingly important to storage.

An increase in IoT (Internet of Things) devices is a factor that explosively accelerates data growth. In other words, data generated from IoT and various things are increasing day by day, which Edge Cloud for efficient management of such data is increasing. This trend has increased the need to reliably and flexibly manage the massive data stored in the Edge Cloud storage. There is increasing interest in integrating between Edge Cloud and Cloud to efficiently manage growing data in the Edge Cloud. However, it is difficult to efficiently integrate large amounts of data between Edge Cloud and Cloud. In order to solve this problem, there is a need of research for scalability, high availability, fault-tolerant, high throughput, low latency, fast recovery, re-scaling, end-to-end consistency, and transactional integration are needed [1, 2].

In this paper, Connected DataLake is designed that a large amount of data stored in various edge clouds is integrated into a micro cloud storage for supporting scalability, high availability, fault-tolerant, high throughput, low latency and fast recovery.

II. Connected DataLake System

A concept of Connected Data Architectures is an interconnected Data Pools which can connect to pools in cloud data centers. That can flow to the pools at the edge and everywhere in between. All data connected and flowing all the time [3]. The Connected DataLake in this paper limits the data connection between the edge cloud and the micro cloud storage. The micro cloud storage in this paper is constructed private cloud storage based on Ceph [4].

The Connected DataLake system proposed in this paper consists of MQT (Massage Queue Transport) module and STR (Stream Processing Router) module as shown in Fig 1. The MQT module transfers a large amount of data from the distributed Edge Cloud to Micro Cloud storage. The SPR module manages data transmission securely and flexibly. The MQT module and SPR module are supported as containers in MCS (Management Center Server).

The MQT module transmits large amount of data securely and flexibly from distributed edge clouds to the SPR module. When failures or crashes occur in the Edge Cloud, it takes control of the SPR which it rolls back massive data from the micro cloud storage to the Edge Cloud. The MQT module is developed using the Kafka framework witch it consists of Producer Module, Broker Cluster, and Consumer Module. The Producer module sends large amounts of data from the Edge Cloud to the Broker Cluster. the Producer module divides the data into sizes that can be transmitted. Then the divided data is serialized and posts it to the broker's topic. Broker Cluster distributes published data to cluster nodes. The Consumer module fetches the posted data in the Broker Cluster and sends it to the SPR module [5].

The SPR is a module for stream processing to manage a large amount of streaming data received from the MQT module securely and flexibly. In other words, this module safely transfers large amounts of data from multiple Edge Clouds to micro cloud storage for integration. It traces failure of a large amount of data transmitted from a variety of sources. It stores checkpoint settings and snapshots for rollback control of transmission fail or an error in the Edge cloud. It transmits and stores the object data for flexible management of large amount of transmission data of each Edge cloud. SPR module consists of Streaming Dataflow Module, Distributed Snapshots Module, Failure Tracing Module and Recovery Module. Streaming Dataflow module converts the received streaming data from the Consumer module into a streaming-enabled format. In Streaming Dataflow module, the input streaming data is divided into a map, which is a subtask unit, an ID is assigned to the divided streaming data by using a hash value, and the format is converted according to the size of the window. Distributed Snapshots module manages Snapshots for the state of streaming data for a set period by setting the checkpoint at the state boundary of the period by classifying the state of the converted streaming data. Failure Tracing module traces the streaming data processes in transit, grasps the state of the process, and sends the acknowledged state to the Recovery module. Recovery module restores to the previous status by using the Snapshots checkpoint according to the failure status of the streaming process [6].

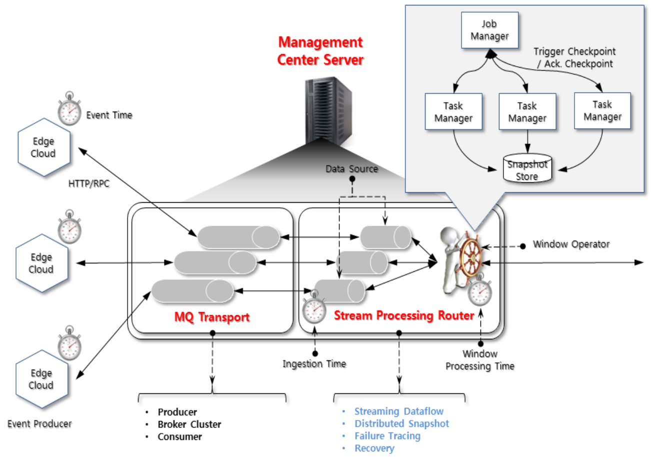

Fig. 1. Concept Diagram of Connected DataLake System

III. CONCLUSION

In this paper, we designed a Connected DataLake system that can solve the difficulties of efficient integration of large number of data transmission between Edge cloud and cloud. The proposed system supports scalability, high availability, fault-tolerant, high throughput, low latency, fast recovery, re-scaling, end-to-end consistency, and transactional integration for the integration of Edge cloud and cloud.

ACKNOWLEDGMENT

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (MEST) (2016R1D1A1B03934823). This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.R7117-16-0218, Development of Automated SaaS Compatibility Techniques over Hybrid/Multisite Clouds). This research was one of KOREN projects supported by National Information Society Agency (18-951-00-001). This research was financially supported by the Ministry of Science and ICT(MSIT), Korea, under the "Construction Project of New Energy Industry Software Convergence Cluster(R&D, ITAS1010170110070001000200200)" supervised by the National IT Industry Promotion Agency(NIPA) and Gwangju Information & Culture Industry Promotion Agency(GITCT).

REFERENCES

[1] A. Meola, "Why IoT, big data & smart farming are the future of agriculture". Business Insider. Insider, Inc. Retrieved 26 July 2018.

[2] G. L. Pedro, M. Alberto, E. Dick, D. Anwitaman, H. Teruo, I. Adriana, B. Marinho, F. Pascal, R. Etienne, "Edge-centric Computing: Vision and Challenges". ACM SIGCOMM Computer Communication Review. 45(5), 2015 .pp.37–42.

[3] M. Harring, "Connected Data Ponds: The Evolution of Data Lakes", https://ko.hortonworks.com/blog/connected-data-ponds-evolution-data-lakes/, July, 2018

[4] Ceph, https://en.wikipedia.org/wiki/Ceph_(software), 2018

[5] Apache kafka, https://en.wikipedia.org/wiki/Apache_Kafka, 2018.

[6] Apache Flink, https://en.wikipedia.org/wiki/Apache_Flink, 2018.