Korean DataBase Conference 2020 (KDBC 2020)

2020년 9월 24일(목) ~ 25일(금)

서귀포 칼 호텔

Table of Contents

Exit

주최 한국정보과학회 데이터베이스 소사이어티 KIISE Database Society of Korea

주관 강원대학교 Kangwon National University 정보통신연구소 Institute of Telecommunications and Information

후원 en·core DATA DRIVEN WORLD

KDBC2020 논문집

2020년 9월 24일(목) ~ 25일(금), 서귀포 칼 호텔

프로그램 (Program)

// Program at a glance //

시간	Session A (다이아몬드)	Session B (로즈홀)	포스터세션 (온라인)
첫째 날: 2020년 9월 24일(목)			
12:00 ~ 13:00	**등록 / 점심 (살레 레스토랑)**		
13:00 ~ 14:30	**Session 1A** (Data Mining and Analytics) 좌장: 김한준	**Session 1B** (Database Engines) 좌장: 권준호	
14:30 ~ 14:40	휴식		
14:40 ~ 16:00	**Session 2A** (Languages and Recommendation) 좌장: 권혁윤	**Session 2B** (Machine Learning and Distributed Databases) 좌장: 문양세	**포스터발표** (온라인)
16:00 ~ 16:30	휴식		
16:30 ~ 18:00	**KDBC Business Meeting** (다이아몬드)		
18:00 ~ 18:20	휴식		
18:20 ~ 18:30	**개회식** (다이아몬드)		
18:30 ~ 18:40	휴식		
18:40 ~ 20:00	**Banquet** (살레 레스토랑)		
둘째 날: 2020년 9월 25일(금)			
09:30 ~ 11:00	**DASFAA 참관** (크리스탈)		
11:00 ~ 11:10	휴식		
11:10 ~ 13:00	분과워크샵		
13:00 ~ 14:00	점심 (살레 레스토랑)		

- 구두발표 13분 (발표 10분, 질의응답 3분)
- 포스터발표 5분 (온라인동영상, 질의응답 이메일)

Session 2B: Machine Learning and Distributed Databases 14:40 ~ 16:00

좌장: 문양세 (강원대학교)

KVDOS: Key-Value Data Orchestration System for Edge Computing Environment
Khikmatullo Tulkinbekov and Deok-Hwan Kim

Multiple Instance Domain Adaptation for Sensor-Based Human Activity Recognition
Aria Ghora Prabono, Bernardo Nugroho Yahya and Seok-Lyong Lee

ECG Clustering Method based on Convolutional Autoencoder for Auto-Labeling System to generate Training Dataset
Soyeon Oh and Minsoo Lee

심전도 신호 기반의 폐쇄성수면무호흡증 심각도 분류 모델
Seo-Young Kim, Dae-Woong Seo and Young-Kyoon Suh

데이터 불균형이 합성곱 신경망의 다수 클래스 멤버쉽 비율에 미치는 영향
Hongjun Choi and Dong-Wan Choi

포스터세션 14:40 ~ 16:00

(온라인)

시계열 교차검증과 특징 추출 알고리즘을 이용한 당일 최대전력수요 예측 기법
Jihoon Moon, Sungwoo Park, Seungmin Jung and Eenjun Hwang

공간정보 중심의 공공데이터셋 연계 방법
Yun-Young Hwang, Sumi Shin and Jin-Hee Yuk

Apache Storm에서 네트워크 통신을 최소화하는 하이브리드 잡 스케줄러
Siwoon Son and Yang-Sae Moon

An Extended Angle Comparison Method for Evaluation of Similarity of Dance Pose
Jae-Jun Lee and Aziz Nasridinov

Simulation of Container Terminal for Generating a Test data set by Data Fidelity
Muhammad Tayyab Ali, Zhong Yi and Bonghee Hong

언어 모델과 클러스터링을 이용한 단어 의미 중의성 해소
Hyeongju Ju and Chulyun Kim

KVDOS: 에지컴퓨팅환경의 키-벨류 데이터 오케스트레이션 시스템

KVDOS: Key-Value Data Orchestration System for Edge Computing Environment

Khikmatullo Tulkinbekov, and Deok-Hwan Kim*
Department of Electrical and Computer Engineering, Inha University, South Korea
mr.khikmatillo@gmail.com, deokhwan@inha.ac.kr

Abstract

A reliable data management has been of the most crucial issues in big data systems over the years. Especially on embedded boards with limited hardware resources, the performance of applications and end-to-end latency highly relies on the database management in the server. In this study, we introduce the key-value data orchestration system (KVDOS) for the edge computing environment. The proposed system handles different types of data sent by end-users and performs the management part using Cassandra wide-column store. We also propose the efficient data backup to improve the reliability. The preliminary experiments show that, the proposed system outperforms the traditional databases in terms of reliability and performance.

Keywords: Edge Computing, Kubernetes, Database, LSM-tree, Cassandra

1. Introduction

In recent years, as the result of dramatic increase of big data generated by IoT devices, handling and processing the data has been becoming in the center of studies. Nowadays, almost all hardware units staring from high performance computers to daily life gadgets are becoming a part of big data communication network. Thus far, central cloud computing protocol has been considered as the reliable solution for handling the send/receive requests from IoT devices. However, the high popularity of the internet communication has increased the demand for the new system with higher transmission and lower end-to-end latency. Edge computing paradigm emerged into big data world as the solution for performance issues of centralized data centers. Locally installed edge servers diminish the load for cloud servers by dividing the data transmission and filtering tasks for small areas.

Fig.1 shows the usage of edge servers as an example in connected car communication. The figure shows the process of sending the traffic information in *Intersection 2* to connected cars on *Intersection 1* with

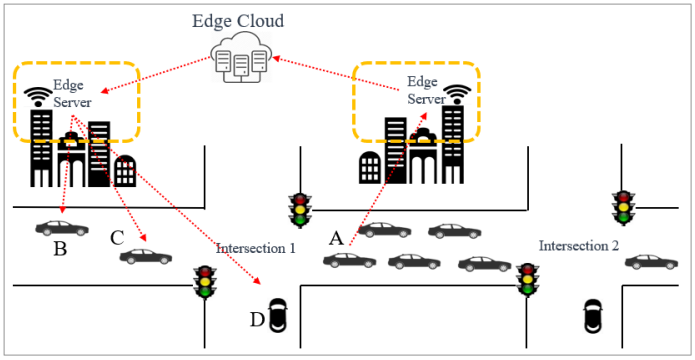

Fig. 1. The implementation and usage of edge servers

locally installed edge servers. Despite the major advantages for task distribution, edge computing comes with hardware resource limitations. The edge server is usually installed as antenna or service routers covering a limited area. This idea requires small hardware units for computing and data storing purposes and cannot afford to provide much space for high-performance big servers. Usually the embedded boards with build-in GPUs are used commonly as the main processing unit in edge computing. On the other hand, the lack of connectivity to external drives of embedded boards increase the latency of communication and highly effects on system reliability. In this study, we point these issues on edge

* *Corresponding author:* Deok-Hwan Kim

computing environment and propose the ongoing project on key-value data orchestration system (KVDOS). The KVDOS is targets the connected car communication as shown in Fig. 1 for the simulation environment and focuses on the data management part in only the selected (with yellow dashes) area. By introducing the KVDOS, we claim three major contributions as mentioned below:

- Data handling with Cassandra for high performance
- Data pre-processing interfaces for steady latency
- Data backup for reliability

The preliminary experiments proved that the proposed system outperforms the traditional data management techniques in terms of performance and reliability.

2. Background and Motivations

In traditional data applications, SQL-based relational databases have been considered as the best solution. Relational database management systems (DBMS) like MySQL would be a potential choice if the data consistency, and availability are main factor of the application. They also provide compatibly high reliability in a single-cluster environment. However, relational databases fail in performance which is the most crucial point in today's development. Moreover, the high demand for multi-cluster architecture for data management decreases brings up the reliability concerns for the SQL-oriented databases. NoSQL [2] databases on the other hand, comes with lightweight solution for the performance and reliability issues. For example, log structured merge (LSM)-tree [1] based key-value (KV) stores, performs independent data processing by indexing each item with its key. KV stores fully deny the data relations in tables and do not support SQL queries but provide only three *put*, *get*, and *delete* interfaces for data management. These alternatives create a tradeoff selecting the database for its usability or performance.

Cassandra [5] comes as a solution by combining the string features of both types of databases. It is an LSM-tree based wide-column store with supports table data management. It also introduces a new interface, Cassandra query language (CQL) that is the simplified form of SQL, free of relations. However, Cassandra fails in performance on edge server environment, where the data size random. CQL supports only the blob data type which is not consistent for big values. Also, lack of connectivity to external drives on embedded boards, the edge server

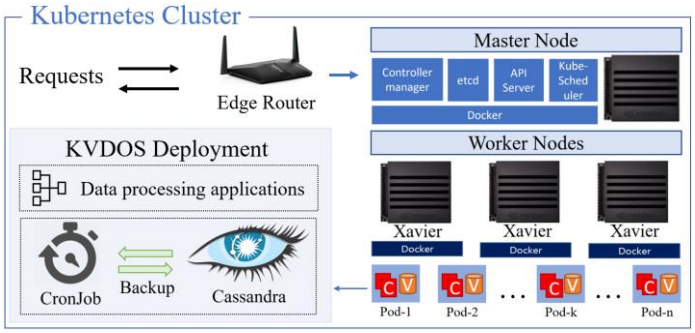

Fig. 2. Edge server architecture

requires a continuously backup. Because of LSM-tree structure of Cassandra data backups cannot be performed easily on memory. We use the Cassandra as the main data store for its adaptability in distributed systems. We introduce solutions for each vulnerable points of Cassandra by proposing KVDOS. Our system fully adapts on edge computing environment.

3. KVDOS

Fig.2 shows the internal architecture of edge server using 4 Jetson AGX Xavier embedded boards and Kubernetes [3] orchestrating the resources. Master node handles the overall controlling and scheduling tasks for all other worker nodes and the client requests. The Kubernetes is responsible for virtualization of all hardware resources of nodes according to application requirements. All nodes are connected to the same network and can be accessed using by router IP. Router maps the nodes and applications to the fixed port for consistent accessibility. KVDOS runs as a deployment controlling different tasks for data management.

The overall diagram of KVDOS implements four data processing layers, where each layer focuses on enhancing the availability, consistency, performance and reliability of the system as shown in Fig. 3. Withing these layers it includes three main features as a set of applications running on Pods as described below:

- Cassandra is used as the main data store and handling unit. As discussed earlier, Cassandra combines the usability and performance features of relational and NoSQL databases. Moreover, Cassandra is a good solution for distributed data management. It comes with full support in node management and task distribution in multi-cluster environment. To keep the balance between reliability and cost of hardware, KVDOS deploys Cassandra server on replicated three Pods. We set up the table in Cassandra as it requires

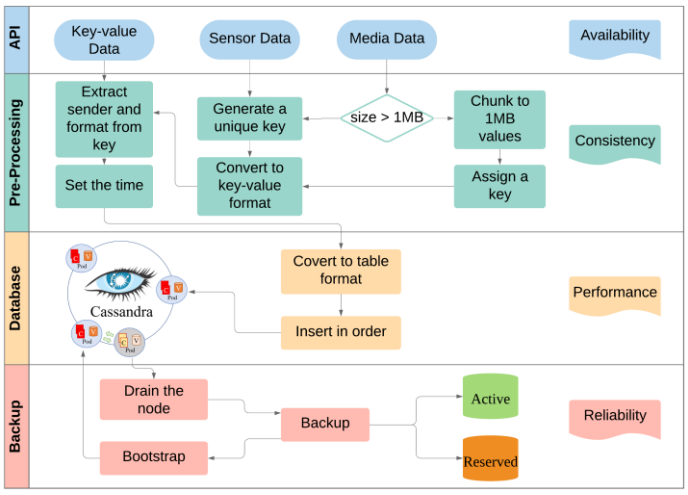

Fig. 3. KVDOS diagram

KV format data and its metadata such as sender id, data type, format and received time. Each row is identified by the key.

– KVDOS provides three overloaded interfaces to handle different types of data sent by connected cars. The default one for all types of data is key-value interface (KVI). The key is required to be formatted in specific order that contains sequentially written metadata. The KVI first extracts the metadata and specify the received time. Then the data will be sent to database layer to be stored in Cassandra in desired format. The most common data types generated by connected cars are sensor and media data. They are handled by the sensor data interface (SDI) and media data interface (MDI) respectively. Sensor data is usually numbers or text those sizes are small. The SDI generated a unique key according to received information and sends to KVI after converting to KV format. The media data generated by camera and might contain small or big size images and videos. Cassandra supports a blob data type to store byte-stream. Even though the blob is expected to hold up to 2GB data, the MDI avoids more than 1MB data in order to keep the best performance of Cassandra. Big files are divided into chunks and managed separately.

– The final layer improves the system reliability by performing scheduled backups using the CronJob scheduled task manager. Cassandra stores data in both memory and disk components. So, backup requires to drain the node to flush all data to disk. While the node is isolated from Cassandra, its job will be distributed to the other nodes. The last backup will be available in reserved volume and current database is stored to active volume. In the next backup, the current volume replaces the reserved one and backup runs circular. After backing up all database including snapshots and configurations, the node will be bootstrapped again to Cassandra network. In this scenario, Cassandra repartitions the nodes and updates membership.

TABLE 1
COMPARISONS

Category	MySQL	LevelDB	Cassandra	KVDOS
Single node	Yes	Yes	Yes	Yes
Multi node	No	No	Yes	Yes
Performance	Low	Good	High–	High+
Chunking	No	No	No	Yes
Backup	No	No	No	Yes

By contributing the features as discussed above, the KVDOS fully adapts on edge computing environment. The pre-processing interfaces serve to keep a steady performance of Cassandra over different types of data. Moreover, the scheduled data backups significantly improve the system reliability.

4. Preliminary Experiments

The initial evaluations are performed on the edge server environment as discussed in section 3. We have compared the performances of MySQL, LevelDB [4], Cassandra and KVDOS. The overall results of experiments are shown in table 1. From the table, MySQL and LevelDB are limited only on single node environment where the Cassandra is applied on multi-nodes too. This proves that it was the best choice for implementation of KVDOS. Moreover, Cassandra outperforms other databases with its architecture. But its throughput is not steady for all values and KVDOS covers it with adapted data interfaces. Also, Data chunking and backups are provided only by the KVDOS which improves its reliability significantly.

We have also performed YCSB [6] microbenchmark evaluations as shown in Fig. 4. YCSB is one of the most popular database benchmarking tools that provides six test workloads. All databases are loaded with 200GB data in mixed size values starting from 50bytes to 16MB and run workloads. From the results, we can see that MySQL fails in performance by all criteria when it highly relies on heavy queries and relations. Moreover, we witness the Cassandra outperforms LevelDB even though they are both share the LSM-tree architecture. The reason is, LevelDB is limited on single node environment and Cassandra

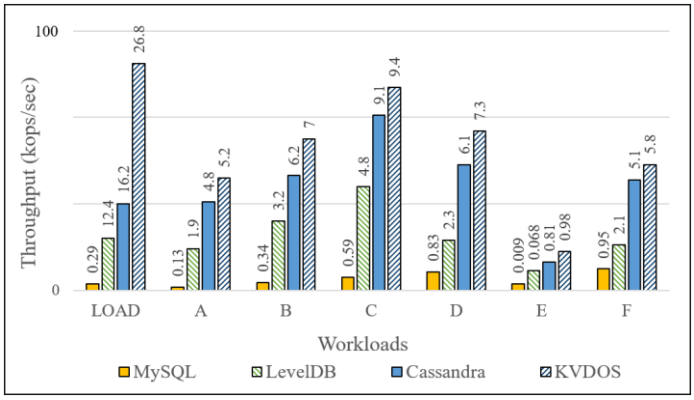

Fig. 4. YCSB macrobenchmark evaluations. *(Workload A has 50% reads and 50% updates, Workload B has 95% read and 5% updates, Workload C has 100% reads, Workload D has 95% reads and 5% new insertions, Workload E has 95% range queries and 5% new insertions, and Workload F has 50% reads and 50% read-modify-writes. Keys are chosen from a Zipf)*

shows better performance running on three nodes. Finally, the KVDOS keeps the best performance of all in all workloads. Especially, we can face a considerable better throughput in data writes since it performs data pre-processing to keep the steady best performance over mixed workloads.

5. Future Work

We plan to continue the development of the KVDOS by enlarging its coverage to data exchanges among multiple edge servers. In this scenario, data will be offloaded to centra edge cloud and broadcasted to other edge servers. Moreover, we plan to focus deeper inside Cassandra architecture and resolve write amplification problem in LSM-tree. This is expected to increase the system performance dramatically resulting to reduce the end-to-end latency in edge computing. Finally, more extensive experiments will be performed to evaluate the system performance by different factors.

6. Conclusions

Efficient data management has been considered in the center of studies over the years. Especially on edge computing environment, it becomes the core issue in terms of reducing latency. Traditional relational databases fail in adaptability in data-oriented systems where the throughput is the main factor. This study proposes the KVDOS that fulfills the edge computing requirements using Cassandra wide-column store. The KVDOS offers three API interfaces for different kinds of data generated by connected cars. Moreover, the scheduled data backups enhance the reliability in edge server. The preliminary experiments proved that the KVDOS outperforms the traditional data systems in terms of performance and reliability.

Acknowledgements

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1B07042602) and in part by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00064, Intelligent Mobile Edge Cloud Solution for Connected Car), (No.2019-0-00240, Deep Partition-and-Merge: Merging and Splitting Deep Neural Networks on Smart Embedded Devices for Real-Time Inference) and in part by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government(MOTIE) (N0002428, The Competency Development Program for Industry Specialist).

References

[1] P. O'Neil, E. Cheng, D. Gawlick, and E. O'Neil, "The log-structured merge-tree (LSM-Tree), " Acta Informatica, vol. 33, pp. 351-385, 1996.
[2] NoSQL Wikipedia. [Online]. Available: https://en.wikipedia.org /wiki/NoSQL, 2019.
[3] Kubernetes. [Online] Available: https://kubernetes.io/docs/home/
[4] Leveldb main page. [Online]. Available: https://code.google. com/p/leveldb/, 2016.
[5] A. Lakshman, and P. Malik, "Cassandra: A decentralized structured storage system," in ACM SIGOPS Operating Syst, vol. 44, pp. 35-40, 2010.
[6] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, Benchmarking Cloud Serving Systems with YCSB. In Proceedings of the ACM Symposium on Cloud Computing (SOCC '10), Indianapolis, Indiana, June 2010.