ISSN 2278-3091
Volume 8, No.5, September - October 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse11852019.pdf
https://doi.org/10.30534/ijatcse/2019/11852019

Gateway-based Resource Control for Reliable IoT Environments

Siwoo Byun[1]
[1]Dept. of Software, Anyang University, South Korea, swbyun@anyang.ac.kr

ABSTRACT

Edge computing refers to decentralized computing technology to reduce cloud computing's overload or security problems that redirect local data to a central data center. Edge computing is emerging as a technology that complements cloud computing in an IoT environment where huge amounts of data are generated in real time.

This study introduces architectures of IoT sensor network, data control schemes, fog computing and edge computing technology. This study also reviews the requirements of IoT service from the aspects of transmission, storage and computing, and analyzes data balancing and data recovery.

This paper also proposes a data management scheme called DaRM, utilizing dual IoT gateways to provide efficient and stable data service in IoT environment. DaRM exploits both edge gateway and fog gateway to improve data reliability and communication efficiency. DaRM can transmit compressed sensor data by reducing overhead in advance through delayed replication and column-based data compression.

Key words : IoT edge gateway, sensor data, column-based compression, resource management, fog computing

1. INTRODUCTION

Recently, IoT(*Internet of Things*) sensor network has received significant attention in smart system areas[1-4]. Small IoT devices can be designed with on-board calculations, wireless communications and sensor detection abilities(Figure 1). Recent work has also begun exploring the potential applications for measuring various IoT environments.

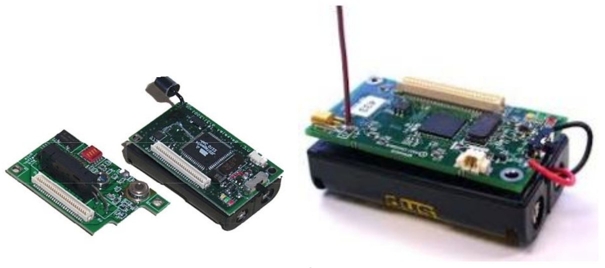

Figure 1: Examples of Tiny Sensor Nodes

IoT applications include energy usage monitoring and planning energy conservation in buildings, military and private surveillance, natural habitats monitoring for understanding environmental dynamics, and collecting data for learning environments.

IoT sensor network is different from the traditional stable networks since IoT applications automatically operate unattended and IoT sensor devices use limited battery and narrow wireless channel.

Most IoT applications are *data-centric* since sensor nodes are designed from the point of measured data rather than identified data such as ip address of conventional networks. That is, measured data is most important in sensor networks. From this architectural point of view, sensor network is treated as a huge database called *sensor database*.

2. RELATED WORKS

2.1 Components of IoT network environment

In general, an IoT network consists of three components: a sensor device, an IoT gateway, and a cloud network, each meaning a data source, a data communication network, and data processor[5,6].

1) Sensor device: Many sensors are placed in wide areas of IoT environment. These sensors produce huge volume of measured data which is a core part of IoT services. The device serves as a human-computer interface that delivers users' requirements to IoT network. These sensors and devices are all be interconnected so that they send sensor data and provide various IoT application services.

2) IoT gateway: IoT gateway collects measured data from sensor devices and forwards it to cloud servers. The sensor devices need to preprocess measured data before they send the data to cloud servers. For example, IoT gateway performs preprocessing of measured data to reduce data redundancy and unnecessary communication overhead.

3) Cloud network: In general, cloud servers have enough resources such as CPU, memory and storages to support IoT applications. The cloud server receives sensor data and user requirements and sends the service results back to the end user after required data processing.

2.2 IoT Edge Computing

Figure 2 shows the base architecture of IoT edge computing [7-9]. IoT Edge computing servers are closer to end users than cloud servers which is located far away. Although IoT edge computing servers provide weaker computing power than cloud servers, but better quality of service (QoS) and lower network latency for end users. In general, the structure of IoT edge computing can be divided into three types: *front-end*, *near-end* and *far-end* [6], as shown in Figure 2.

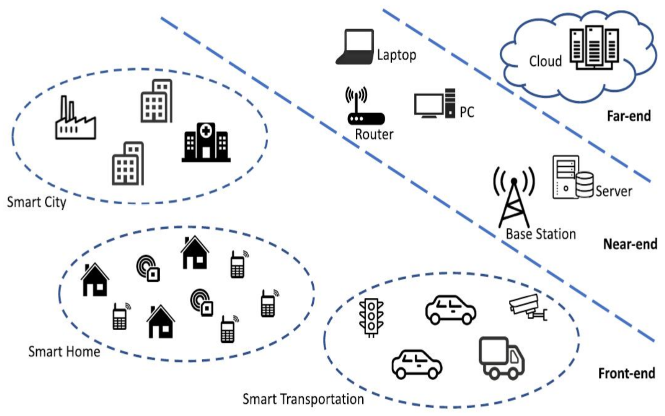

Figure 2: Architecture of edge computing networks

1) Front-End: The end devices such as sensors and actuators, are placed at the front-end in IoT edge computing. Front-end computing can provide end users with more interaction and better responsiveness. Because of the limited capacity of the end devices, most of the requirements cannot be met at the front-end, so the end devices must forward the request to the server.

2) Near-End: Gateway located in the near-end supports most network services. Edge servers have a various requirements such as real-time data processing, data caching, and computation offloading. Migrating most of the data processing and storage capabilities to this near-end environment allows users to achieve even better performance on data computing and storage with a slight increase in latency.

3) Far-End: Since the server is located far from the end device, the transfer delay is severe. On the other hand, far-end servers can provide much more computing power and storage. For example, far-end servers can provide massive parallel data processing, big data mining, replicated data management and machine learning.

2.3 Compression for IoT data

Traditional row(record) based storage systems have to read a lot of unrelated data from data repository, so they have performance limitations when handling a number of concurrent, diverse queries. In contrast, in column-based storage systems, only the columns related to the query need to be taken from the data repository which is densely compressed to improve read efficiency[10,11].

In particular, column-wise storing skill enables significant improvements in data compression, reduces the number of storage drives while maintaining high performance. As a simple example, each city name, such as Los Angeles, can be mapped to an integer value such as 1397, which requires storage of two bytes instead of 11 bytes. Run-length encoding is another simple compression technique that uses counts of repetitions, which can be used to reduce space needs [10].

Compared to simple row-oriented databases, column storage methods need to be reconstructed for most database access standards (e.g., ODBC, JDBC). However, the overheads can be mitigated by some techniques, such as memory buffering, tuple movement, and partition merging. Moreover, regarding the computational efficiency of pipeline query or vectored query processing, the column storage method is much more suitable to take advantage of the advanced functions such as pipeline CPU, CPU branch prediction, CPU cache. In fact, column storage systems show superior CPU and cache performance in benchmarks such as TPC-H [11].

2.4 Replication for IoT Data

Although current database can be applied to general network or mobile environment, it is difficult to apply it to IoT sensor network environment that has incomplete characteristics. That is, IoT-based applications can suffer from unreliable and delayed services because sensor networks have unreliable wireless channels and narrow bandwidth. One traditional way to reduce the potential for this undesirable data service is to replicate data across multiple nodes.

As an important example, if body heat is detected in an emergency disaster recovery area, the location data of the survivors should be quickly sent to the control center regardless of any disturbance (wireless communication failure, sensor failure, etc.). In this situation, data replication of sensors can contribute to improving data availability.

Traditional replica management exploits eager or lazy approaches. While the eager approach uses a synchronous technique, lazy approach uses asynchronous technique that can spread the renewal operation to all other node sites after the local renewal is completed. Traditional eager approaches include a primary copy technique based on main site consent and a quorum consensus technique. Quorum consensus is a coordination technique that proceeds renewals after obtaining the necessary consent, and is recently used in blockchain applications[12,13].

3. PROPOSED SCHEME

3.1 IoT Performance Demands

1) Transmission: The total response time is the sum of the transfer time and the processing time. Since IoT devices generate huge amounts of data continuously, excessive network delays are not allowed. Specific examples are vehicle-vehicle communication and vehicle-infrastructure communication, and the related response time is very critical. Unlike traditional clouds such as Google Cloud, IoT edge computing offers numerous distributed nodes and is close to end users. Thus, IoT application can take advantage of the shorter transfer times of edge computing gateways.

2) Storage: Since IoT data is the most important part of big

data generation, it should be uploaded to edge or cloud storage. The benefits of uploading to edge storage is the short upload time, but it has some drawbacks. That is, edge nodes are difficult to ensure security, integrity of the original data, and information protection. Moreover, as compare to the cloud centers, the edge nodes have no long-lived storage.
3) Computation: Most IoT devices have limited computation and energy resources. Thus, IoT devices transmit the data to more powerful computing nodes to analyze the data. Since the computation power is severely limited, edge nodes reduces the CPU workloads by the offloading the computation tasks.

3.2 IoT Data Management

Cloud computing-based storage is implemented as multi-layer systems which combine commercial server and disk drive groups. To meet QoS requirements, edge-based storage also utilizes load balancing and fault recovery technologies. In particular, failover techniques are essential to cope with data failures from multiple sources (e.g., software, hardware, packet loss, noise, and power problems).
1) Storage balancing: Since IoT devices have very limited storage space, all data collected or generated should be sent to the storage server. If all devices simultaneously transfer data to cloud storage, this will be a significant burden to the network. Instead, sending data to several edge storage based on the characteristics of the edge topology reduces the long-distance traffic.
For this, load balancing and resource allocation technologies for edge-based storage are essential. It monitors different storage demand rates and uses data replication for traffic and storage balancing. Additionally, selecting the nearest edge storage node and weighted data control, can reduce the storage access time.
2) Recovery policy: Typically, periodic pinging or heartbeat is conducted by monitoring systems to check storage system health and availability of edge nodes. For example, network devices may not be available, the operating system of the edge node may be damaged, the storage hardware may fail, the system recovery process may shut down the entire system for maintenance. Nonetheless, in edge storage systems, the other available edge nodes will act as redundant storage.
In the IoT environment, a huge number of devices generate intensive request for storage. Clearly, correctness of sensitive data such as personal health data, energy consumption records, speed of smart vehicles is essential.
Distributed storage systems use replication to increase reliability. In a distributed storage system, the data is divided into several pieces and the pieces of data overlap each other. As a result, the data stored in each piece can be reconstructed from other relevant pieces. Edge storage is essentially a distributed storage system and not only logically distributed, but also physically distributed. Thus, edge storage architecture significantly mitigates the risk of data loss.

3.3 Efficient Data Management using IoT gateway

IoT edge-gateway shares fast local network with front-end sensor devices. Thus, IoT edge gateways provide much faster transmissions as compared to cloud servers connected to the slow Internet. However, the IoT edge gateway has limited memory and storage space as compared to the cloud servers. Therefore, it is important to design efficient data management by taking advantage of superior network conditions, taking into account limited resource conditions.
In particular, storage devices lead to heavy system workloads because they constantly have to store large amounts of data regardless of its importance from many sensor devices. Therefore, the burden of storing this intensive data can be greatly reduced by *offloading* unnecessary data in advance. In other words, if non-critical data is not prevented from being put into the next operation, the overloaded storage operations degrade edge-gateway performance severely.
However, although the gateway may determine the importance of the data right away, the strategic judgment of the cloud should be included. Therefore, proposed scheme prefers not to use immediate removing strategy, but to use delayed removing strategy that waits the confirmation of the cloud. For example, all the sensor data is kept in a gateway's bulk storage such as hard disk during one day at very low cost, and the data that is not needed later is linked to garbage places such as trash cans. That is the invalidated data is transferred to the waste repository, erased at idle time, and recycled later. Although most sensor data are initially processed with the maximum I/O performance, but if the storage device or memory becomes overloaded, the following data is not processed at that speed. Therefore, it is needed to prevent this overloading and to keep data availability and reliability. Our approach aims to increase data distribution speed across the entire network and to increase data availability and reliability by adding an IoT gateway called *fog-gateway* in the entry point of the cloud servers.
Edge gateway and fog gateway have similar hardware design and specification, but have different placement locations. While edge gateways are connected to local network which has broad bandwidth, their storage efficiency is low since the acquired data are stored without compression. Fog gateways, on the other hand, utilize data compression for high storage efficiency, while they are connected to slow Internet.
If the edge and fog gateways are efficiently combined, they provide faster network transmission and higher data efficiency for overall IoT network. That is, it is effective to increase the communication efficiency of edge gateways, to increase the storage efficiency of fog gateways, to offload sensor data that is not needed on edges, and to send it to the cloud.
In this respect, this study proposes a new management framework called DaRM (Data-aware Resource Management) to improve the availability and reliability of IoT data (Figure 3).

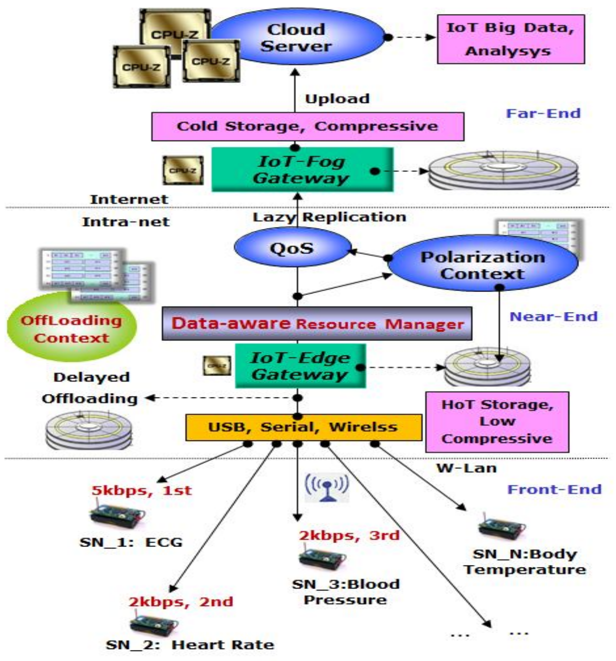

Figure 3: Data-aware Resource Management Model

First, edge gateways perform data-aware offloading to reduce system overheads such as I/O delay, severe transmission delay, memory shortage, and CPU overload. If the unnecessary data are not offloaded in advance, they can eventually result in the traffic congestion due to excessive data workload.

Next, to enhance the data availability and reliability, the edge gateways are interconnected to the fog gateways below the cloud layer.

DaRM analyzes the compression ratio of sensor data and recognizes the compression characteristics. Then, the sensor data are classified to two classes, *compressible* and *incompressible*, according to the compression efficiency. DaRM also controls the transfer rate according to the queue length of a task in edge gateway. DaRM utilizes column-based compression skill, and uses a real-time data compression library called lzo which is not only easy to handle source code but also controls compression ratio and speed by changing compression levels. If DaRM exploits more advanced algorithms, data compression performance can be improved much more.

Data classification is divided into two categories depending on the compression ratio. This means that the characteristics of the sensor data stream are classified into compressible class such as text data, and incompressible class such as voice signals that are not already compressed.

For critical data that need to be replicated, the same data is maintained on both edge and fog gateways. Among the replication management techniques, the delayed methods have the disadvantage of less data consistency due to their asynchronous nature. On the other hand, eager techniques ensure strong consistency. Although eager protocols are possible under stable conditions, they are not suitable in unstable IoT environment which consists of unstable devices and wireless network. In particular, delayed protocol is advantageous in IoT network which needs to transmit data over the long distance Internet. In this respect, this study exploits a delayed technique called *lazy replication* in DaRM scheme(algorithm 1).

Algorithm 1: Handling proposed DaRM scheme

```
1) Upon the receipt of sensor data generated by sensor
   device in the front-end area, DaRM inserts the sensor
   data into the log storage such as bulk hard disk. After
   the predefined period, if offloading process has reached
   to confirm-state, DaRM remove the sensor data marked
   as 'remove'.
2) DaRM reads the log of the sensor data one by one, and
   fetches the compression ratio of the column attribute
   related to the data.
3) DaRM inserts the sensor data into the tail of
   replication-queue with its compression ratio.
4) If the mark of the data in replication-queue is
   high-compressive type, DaRM executes FOG_SEND. If
   the type of the operation is low-compressive type,
   DaRM executes EDGE_WRITE.
5) FOG-SEND:
   5.1) DaRM finds the location of fog gateway related to the
        copy operation by reading the column info in edge
        gateway.
   5.2) DaRM forwards the sensor data to Lzo-Manager.
        The Lzo Compressor returns the compressed
        version of the data to DaRM
   5.3) DaRM forwards the compressed sensor data to fog
        gateway.
6) EDGE-WRITE:
   6.1) DaRM finds the location of fog gateway related to the
        copy operation by reading the column info in edge
        gateway.
   6.2) DaRM reads the Q-Length of work-queue in edge
        gateway.
   6.3) If the Q-Length is less than QL-Congest, then insert
        the sensor data into the data storage of the edge
        gateway.
   6.4) If the Q-Length is greater than QL-Congest, then
        perform 5)FOG-SEND process.
```

DaRM prioritizes the sensor data so that it can be sent to the fog gateway or can be stored first. Urgent IoT service such as health care needs priority classification. High-priority class device has ensured priority on system resources such as network bandwidth and storage space as followings.

high-priority class: {ECG, heart rate, blood pressure},
mid-priority class: {body-temperature, battery level},
low-priority class: {room-temp., humidity, luminance}

In addition, for reliable and guaranteed data transfer, the network bandwidth between the two gateways should be guaranteed. DaRM maintains each priorities and minimum bandwidth, ensuring that urgent class of data are reliably sent even in the event of communication congestion.

4. CONCLUSION

This research introduced recent IoT network and sensor data control technology. Among them, this paper analyzed the two emerging technologies, fog and edge computing. In detail, the requirements for new IoT computing were analyzed in terms of transmission, storage and computing, and then data load balancing and data recovery methods were presented.

This paper proposed a data management scheme called DaRM, utilizing dual gateways to provide efficient and stable data service in IoT environment. DaRM exploits both edge gateway and fog gateway to improve data reliability and communication efficiency. DaRM can transmit compressed sensor data by reducing overhead in advance through delayed replication and column-based data compression.

ACKNOWLEDGEMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology(2018R1D1A1B07044418)

REFERENCES

1. https://www.ics.uci.edu/~dsm/ics280sensor/readings/data/02-771.pdf, Feb. 25 (2019).
2. P. Bonnet, J. Gehrke, and P. Seshadri, **Towards Sensor Database Systems**, in *Proc. 2nd International Conference on Mobile Data Management*, Hong Kong, January 2011, pp.3-14. https://doi.org/10.1007/3-540-44498-X_1
3. S. Yeon, J. Park, **IoT Platform Analysis and Issues**, in *ETRI Insight Report*, vol.27, pp.1-56, 2016.
4. S.V.R.K.Rao, M.Saritha Devi, A.R.Kishore, Praveen Kumar, **Wireless sensor Network based Industrial Automation using Internet of Things (IoT)**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol.7, no.6, pp.82-86, 2018. https://doi.org/10.30534/ijatcse/2018/01762018
5. A. ElSharif Karrar, M. Fadl Idris Fadl, **Security Protocol for Data Transmission in Cloud Computing**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol.7, no.1, pp.1-5, 2018. https://doi.org/10.30534/ijatcse/2018/01712018
6. Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, Xinyu Yang, **A Survey on the Edge Computing for the Internet of Things**, *IEEE Access*, vol.6, pp.6900-6919, Nov. 2017. https://doi.org/10.1109/ACCESS.2017.2778504
7. S. Yi, Z. Hao, Z. Qin, and Q. Li, **Fog Computing: Platform and Applications**, in *Proc. 3rd Third IEEE Workshop on Hot Topics in Web Systems and Technologies*,2015, pp.73-78
8. Gopika Premsankar, Mario Di Francesco, and Tarik Taleb, **Edge Computing for the Internet of Things: A Case Study**, *IEEE Internet Of Things Journal*, vol.5, No.2, 2018. https://doi.org/10.1109/JIOT.2018.2805263
9. Amir M. Rahmani, T. Nguyen Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, P. Liljeberg, **Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach**, *Future Generation Computer Systems*, vol. 78, no.2, pp. 641-658, 2018. https://doi.org/10.1016/j.future.2017.02.014
10. S. Byun, and S. Jang. **Asymmetric Index Management Scheme for High-capacity Compressed Databases**, *Journal of Korea Academia-Industrial*, vol.17, no.7, pp.293-300, 2016.
11. S. Ahn, and K. Kim, **A Join Technique to Improve the Performance of Star Schema Queries in Column-Oriented Databases**, *Journal of Korean Institute of Information Scientist and Engineers*, vol. 40, no.3, pp.209-218, 2013.
12. E. Tavares de Camargo, E. P. Duarte Jr., F. Pedonez, **A Consensus-based Fault-Tolerant Event Logger for High Performance Applications**, in *Proc. Euro-Par 2017: Parallel Processing*, 2017, pp.415-427. https://doi.org/10.1007/978-3-319-64203-1_30
13. C. Cachin and M Vukolic, **Blockchain consensus protocols in the wild**, in *Technical Report arXiv:1707.01873*, IBM Research - Zurich, July 2017. https://doi.org/10.1109/EDCC.2017.36