Research and Implementation of an Aquaculture Monitoring System Based on Flink, MongoDB and Kafka

Yuansheng Lou[1], Lin Chen[1], Feng Ye[1,2], Yong Chen[2,3] and Zihao Liu[4]

[1] College of Computer and Information, Hohai University, Nanjing 211100, China
[2] Postdoctoral Centre, Nanjing Longyuan Micro-Electronic Company, Nanjing 211106, China
[3] Huai'an Longyuan Agricultural Technology Company, Huai'an 223345, China
[4] Jiangsu University of Science and Technology, Zhenjiang 212000, China
yefeng1022@hhu.edu.cn

Abstract. With the rapid advancement of intelligent agriculture technology, the application of IoT and sensors in aquaculture domain is becoming more and more widespread. Traditional relational database management systems cannot store the large scale and diversified sensor data flexibly and expansively. Moreover, the sensor stream data usually requires a processing operation with high throughput and low latency. Based on Flink, MongoDB and Kafka, we propose and implement an aquaculture monitoring system. Among them, Flink provides a high throughput, low latency processing platform for sensor data. Kafka, as a distributed publish-subscribe message system, acquires different sensor data and builds reliable pipelines for transmitting real-time data between application programs. MongoDB is suitable for storing diversified sensor data. As a highly reliable and high-performance column database, HBase is often used in sensor data storage schemes. Therefore，using real aquaculture dataset, the execution efficiency of some common operations between HBase and our solution are tested and compared. The experimental results show that the efficiency of our solution is much higher than that of HBase, which provided a feasible solution for the sensor data storage and processing of aquaculture.

Keywords: Aquaculture, Flink, MongoDB, Kafka, Big Data.

1 Introduction

With the advancement of IoT (Internet of Things) technologies, the aquaculture industry is also facing a transition from the traditional rough pattern to the refined, intelligent pattern. The IoT has been widely used in all parts of the aquaculture production such as data collection, environmental monitoring, fish fry epidemic detection [1]. Large-scale business data such as ammonia nitrogen, temperature, dissolved oxygen and pH value, is produced and needs to be processed. When storing massive data, the capacity of vertically increasing data nodes is limited, and the performance bottleneck will be caused in the processing of massive data [2]. Therefore, the distributed cloud storage scheme that dynamically accesses new storage nodes through horizontal expansion is an ideal solution. In addition, the heterogeneous and

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22744-9_50

non-structural trend of these IoT data is obvious, and using traditional database systems for storage is no longer appropriate. NoSQL stores support dynamic data model, which can deal with the variety, complexity and later expansion of data acquisition devices in the IoT [3].

On the other hand, the real-time requirement of sensor data processing in aquaculture is very high. Taking water temperature as an example, the water temperature will directly affect the growth and health of fish. Temperature sensors frequently acquire measurement data and transmit them to data centers in the form of streams. In data centers, real-time or near-real-time applications update the display board and issue warnings about changes in water temperature to avoid losses in aquaculture [4]. Therefore, based on Flink, MongoDB and Kafka, this paper proposed and implemented a high-throughput, low-latency architecture for processing real-time streaming data of the IoT of aquaculture.

2 Background and Related Work

2.1 Apache Flink

Many systems generate continuous stream of events. If we can efficiently analyze large-scale sensor stream data, we will have a clearer and faster understanding of the system. In this context, Apache Flink [5] came into being. Flink is an open source stream processing framework that supports distributed, high performance, ready-to-use, and accurate streaming applications. Flink not only provides real-time computing that supports both high throughput and exactly-once semantics, but also provides batch processing. Flink treats batch processing (that is, processing limited static data) as a special stream processing, so that batch and stream processing can be implemented simultaneously. Its core computational construct is the Flink runtime execution engine in Figure 1, which is a distributed system that accepts data flow programs and performs fault-tolerant execution on one or more machines. The Flink Runtime Execution Engine can run as a YARN (Yet Another Resource Negotiator) application on a cluster, on a Mesos cluster, or on a standalone machine. Figure 1 is an architectural diagram of Flink.

Existing experiments show that by avoiding flow processing bottlenecks and utilizing Flink's stateful stream processing capability, the throughput can reach about 30 times of Strom. At the same time, the exactly-once and high availability can be guaranteed [6]. Therefore, Flink is chosen as the data processing platform in this paper.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22744-9_50

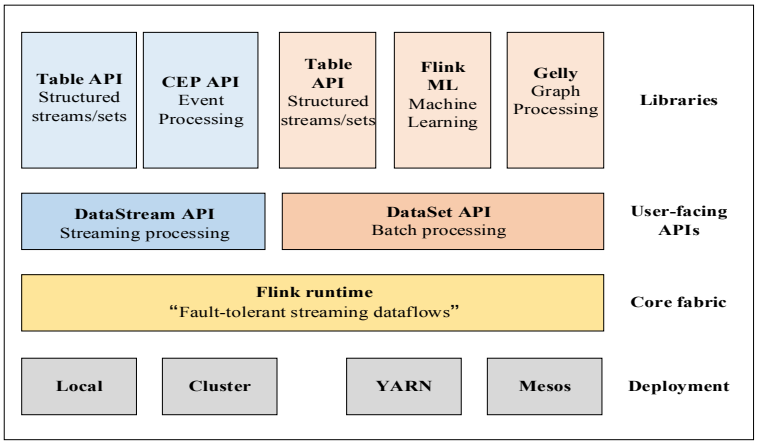

Fig. 1. The architecture diagram of Apache Flink

2.2 MongoDB

MongoDB[7] is a distributed, document-oriented, open source document database. MongoDB is the most similar to relational database among non-relational databases. It has rich functions and can even use many SQL statements for relational query, etc. Moreover, it has a variety of performance advantages of non-relational databases, such as convenient deployment, painless expansion and storage without mode. MongoDB supports a variety of common development languages, has a number of open source MongoDB frameworks, and simplifies data layer operations.Compared with the traditional MySQL database, MongoDB is much more efficient than MySQL in addition, deletion, selection and other operations[8]. In addition, MongoDB is a fast scalable database that can greatly increase the performance of Flink.

2.3 Kafka

Apache Kafka [9] is a distributed publish-subscribe messaging system. It can handle a large volume of data which enables you to send messages at end-point. Apache Kafka is developed at LinkedIn & available as an open source project with Apache Software Foundation. In this article, Kafka was chosen as the message broker. It supports Apache Flink very well. In addition, it provides precise primary semantics to ensure that each record is eventually delivered to its consumer, even in the event of a failure, and that no duplication is created in the process [10].

2.4 Related Work

At present, MapReduce programming model and Hadoop architecture are the popular processing technologies for big data in aquaculture, but Hadoop is a typical batch processing architecture for big data, and cannot meet the real-time requirements for streaming data processing [11-13]. In [14], by implementing Lambda architecture, the real-time computing platform is combined with the offline batch processing mechanism to make the agricultural big data processing framework have both the

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22744-9_50

function of streaming data and the function of historical data mining. But this architecture requires two programming of the same business logic in two different APIs: one for batch computing and one for streaming computing. For the same business problem, there are two code bases, each with different vulnerabilities. Such systems are more difficult to maintain.

In terms of data storage, NoSQL is the mainstream storage scheme of IoT data. In [15], through testing and comparing the storage performance and data processing performance of MongoDB, HDFS and MySQL, experiments show that MongoDB has higher scalability and higher availability, and can store IoT data efficiently. Therefore, MongoDB is a feasible sensor data storage scheme. In [16], HBase, which is a distributed column-oriented database based on Hadoop file system, is used to store stream data. It can also be scaled horizontally like MongoDB. Therefore, this paper tested and compared the performance of HBase and MongoDB under the IoT data storage scenario.

In summary, a high throughput, low delay and easy maintenance architecture is urgently needed to deal with the stream data generated by sensors.

3 The Architecture of the Aquaculture Information Monitoring System and Implementation

3.1 Architecture

According to the characteristics of aquaculture IoT data and considering the key factors such as reliability, flexibility, scalability and load balancing, the system can be divided into four layers: data collection layer, data processing layer, business logic layer and application display layer. Figure 2 shows the whole architecture of the system.

In the data collection layer, Kafka is a pub-subscribe messaging system. We create different topics for different data stream. Data collection layer pushes JSON objects of sensor data to corresponding topics. Kafka's ZooKeeper can realize dynamic cluster expansion. Once the zookeeper changes, Kafka client can perceive and make corresponding adjustments in time. This ensures that when brokers are added or removed, they are still automatically load-balanced among themselves.

In the data processing layer, Flink receives and analyzes data in different topics, and analyzes infinite data streams by calling the DataStream API. The stream data processing program is implemented with the Java language. The processed data can flow to another message queue or be directly updated in the local database by the Flink program for historical query. To improve the fault tolerance, we adopt Flink's State management mechanism. State is the state of calculation when the stream data is saved from one event to the next event, and the calculation state can continue to accurately update the state after the failure or interruption. For storing massive amounts of sensor data, we use MongoDB for data persistence.

In the business logic layer, for the data that needs to be displayed in real time, this layer constantly obtains the stream data processed by Flink for real-time update. For

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22744-9_50

historical queries, this layer provides various query interfaces for MongoDB database tables. Spring is used to manage object dependencies through dependency injection.

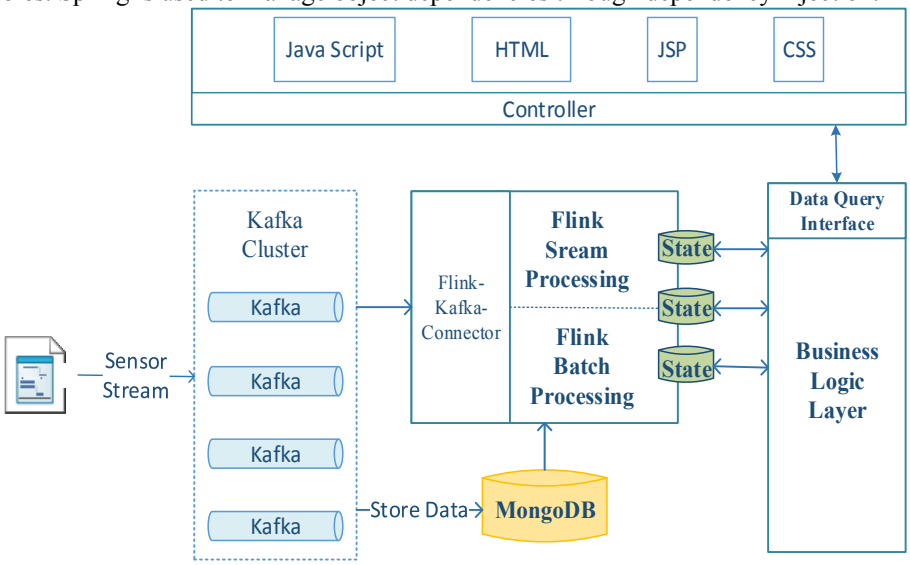

Fig. 2. The framework of aquaculture system

The application display layer uses JavaScript, HTML, CSS and JSP to implement the front-end interface. When the client makes a request, Tomcat receives the request and forwards the request to the DispatcherServlet for processing if it matches the mapping path configured by the DispatcherServlet in web.xml. The DispatcherServlet is used as the front-end controller for request distribution, and the controller is called by the browser request. Controller processes business requests and returns the corresponding view page.

3.2 System Implementation

At present, aquaculture information monitoring system adopts 30 million sensor data of an aquaculture farm to simulate stream data, and Flink is used to realize real-time update and historical query function of dashboard sensor, as shown in Fig.3. The green part of the line chart indicates that the temperature is in normal value, and the red part indicates that the temperature is beyond normal value. By means of visualization [17], users can easily see the maximum, minimum and average temperature in different time periods.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22744-9_50

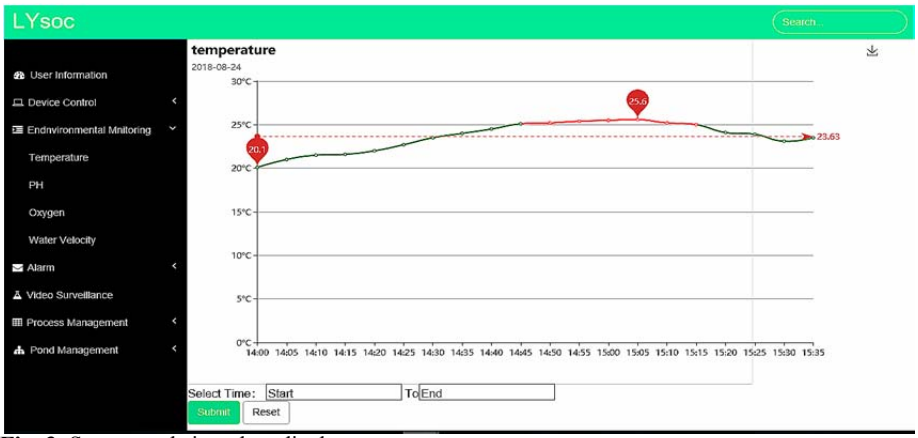

Fig. 3. Sensor real-time data display

4 Experiments and Discussion

In order to verify the performance of this storage scheme, the insertion and query performance of MongoDB and HBase were tested in different orders of magnitude. In order to verify the performance of the storage scheme in this paper, the insert and query time of MongoDB and HBase were tested under different orders of magnitude scenarios.

MongoDB and HBase clusters are deployed on four PCs of the same type. The PC's configuration is: Intel(R) Xeon(R) CPU E5645@2.40GHz dual-core 24 CPU, Kingston DDR3 1333MHz 8G, 500GB SSD Flash Memory. Operating system are Ubuntu 16.04 64-bit and Linux 3.11.0 kernel.MongoDB version is 3.6.3 and HBase version is 1.2.6. The CPU utilization of each primary node is monitored by Ganglia.

4.1 Data insertion performance verification

The experimental steps are as follows.

1) 30 million pieces of sensor data are randomly selected from the monitoring system of the IoT, and then 3 groups of data are randomly selected from these data. The data volumes are 10,000 pieces, 100,000 pieces and 250,000 pieces of data respectively.
2) The data is sent to the Topic through Kafka Producer. The MongoDB cluster with 4 nodes obtains data from Kafka Broker.
3) Build a HBase cluster with 4 nodes under the same hardware configuration, and get data from the same Kafka Broker in 2).
4) Record the insertion time and CPU utilization, and repeat the insertion five times to get the average.

The insert performance comparison between MongoDB and HBase is shown in Figure 4.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22744-9_50

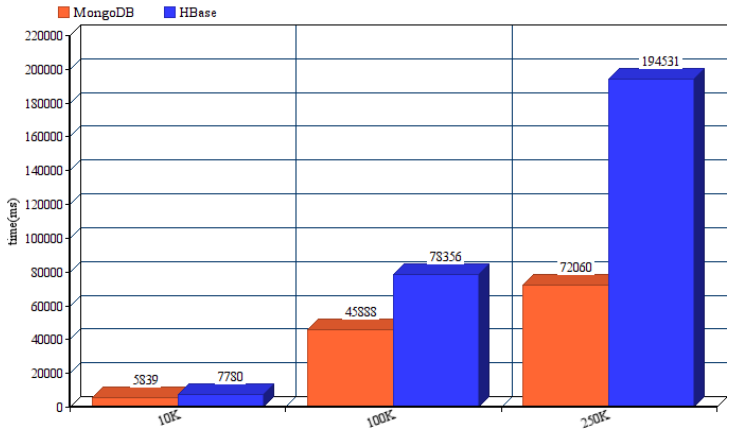

Fig. 4. Performance comparison of data insertion between MongoDB and HBase

It shows that when the data scale is small, the insertion performance of the MongoDB cluster and the HBase cluster is close; however, when the data volume is 100,000 pieces of data, MongoDB cluster data insertion performance is obviously better than that of HBase cluster; when the data volume is 250,000, the insert operation time of MongoDB cluster data is only 37.04% of that of HBase cluster. Ganglia recorded the CPU utilization in the process of data insertion, and the results are as follows.

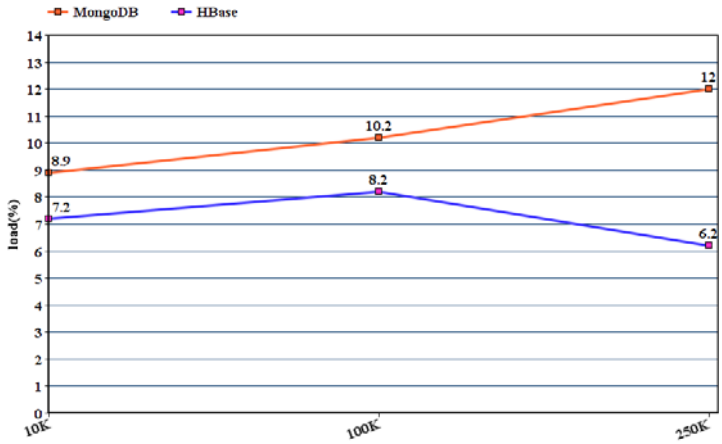

Fig. 5. CPU utilization comparison of data insertion

It can be seen that with the increase of data quantity, the CPU utilization of MongoDB slightly increase, but the change range is not large, and the change of CPU utilization of HBase is relatively stable.

4.2 Data query performance

The test steps are as follows.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22744-9_50

1) Respectively, MongoDB cluster and HBase cluster are used to query 10,000, 100,000 and 250,000 pieces of data from 30 million pieces of data set in experiment 4.1.
2) Record the insertion time and CPU utilization, and repeat the query operation five times to get the average.

The insert operation performance comparison between MongoDB and HBase is shown in Figure 6.

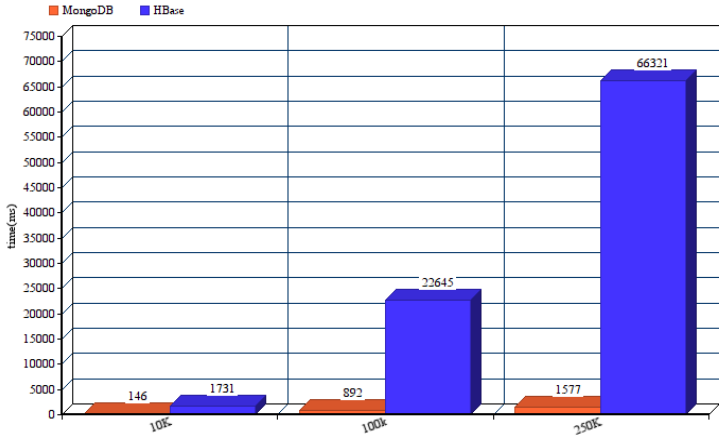

Fig. 6. Performance comparison of data query between MongoDB cluster and HBase cluster

When the data size is small, the data query performance of MongoDB cluster is close to that of HBase cluster, but always lower than that of HBase.With the increase of data volume, the query time of HBase increased greatly, while the growth of MongoDB is small and far smaller than that of HBase. When the query data volume is 250,000, the MongoDB cluster's data query time is only 2.38% of HBase cluster's. As a result, when the amount of data is large, the query speed of MongoDB is faster.

Ganglia recorded the CPU utilization in the process of data query, and the results are shown in Figure 7.

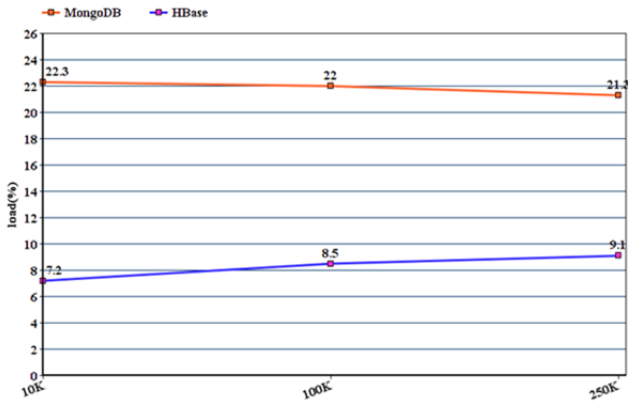

Fig. 7. CPU utilization comparison of data query

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22744-9_50

With the increase of queries, the CPU utilization of MongoDB and HBase increase slowly, and the CPU utilization of MongoDB is significantly higher than that of HBase.

Through the above two comparative experiments, we can see that when the sensor data volume is large, the insertion and query efficiency of MongoDB cluster is much higher than that of HBase. Although the CPU utilization is slightly higher than that of HBase, it is basically between 10% and 20%. Therefore, in the case of high throughput and high concurrency, it is feasible to use MongoDB to store sensor data.

5 Summary and Outlook

This paper summarizes the problems encountered in the data processing of IoT sensor in aquaculture, namely the massive, non-structural and real-time processing, and proposes and implements the aquaculture monitoring system based on Flink, Kafka and MongoDB. Taking sensor data acquired in aquaculture monitoring as experimental data, the data insertion and query performance of MongoDB and HBase were tested and compared. The experimental results show that MongoDB is much more efficient than HBase in storing and querying massive sensor data, which can meet the storage and management requirements of massive and real-time data of the aquaculture IoT. At the same time, MongoDB supports dynamic data model, making the system easier to expand.

The subsequent research will focus on Flink, which combines distributed computing and machine learning technology to calculate the real-time data of the IoT, providing real-time disaster warning and decision-making services for aquaculture.

Acknowledgement

This work is partly supported by the 2018 Jiangsu Province Key Research and Development Program (Modern Agriculture) Project under Grant No.BE2018301, 2017 Jiangsu Province Postdoctoral Research Funding Project under Grant No. 1701020C, 2017 Six Talent Peaks Endorsement Project of Jiangsu under Grant No.XYDXX-078.

References

1. Shetty, S., Pai, R.M., Pai, M.M.M.: Energy Efficient Message Priority Based Routing Protocol for Aquaculture Applications Using Underwater Sensor Network.Wireless Pers Commun 103(2), 1871-1894 (2018).
2. Xu, X., Shi, L., He, L., Zhang, H., Ma, X.: Design and implementation of cloud storage system for farmland internet of things based on NoSQL database. Transactions of the CSAE, 35(1), 172-179 (2019).
3. Edward, S.G., Sabharwal, N. : Practical MongoDB. Apress. Berkeley, CA(2015).
4. Liu, S., Xu, L., Chen, J., Li D., Tai H., Zeng L.: Retracted: Water Temperature Forecasting in Sea Cucumber Aquaculture Ponds by RBF Neural Network Model. In: Li, D., Chen, Y. (eds.) Computer and Computing Technologies in Agriculture VI. CCTA 2012. IFIP

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22744-9_50

Advances in Information and Communication Technology, vol 392, pp.425–436. Springer, Berlin, Heidelberg(2013).
5. Tanmay, D.: Learning Apache Flink. Packt Publishing, Birmingham(2017).
6. Friedman, E., Tzoumas, K.: Introduction to Apache Flink: Stream Processing for Real Time and Beyond. O'Reilly Media, Sebastopol(2016).
7. Chodorow, K.:MongoDB the definitive guide.O'Reilly Media,Sebastopol(2013).
8. Győrödi, C., Győrödi, R.; Pecherle, G., Olah, A.:A Comparative Study: MongoDB vs. MySQL.In: 13th International Conference on Engineering of Modern Electric Systems (EMES),IEEE, pp. 1-6(2015).
9. Narkhede, N., Shapira, G., Palino, T.:Kafka the definitive guide. O'Reilly Media, Sebastopol(2013).
10. Versaci, F., Pireddu, L., Zanetti, G.: Kafka Interfaces for Composable Streaming Genomics Pipelines.In: IEEE EMBS International Conference on Biomedical & Health Informatics (BHI),IEEE, pp. 259-262(2018).
11. Wang, D., Zheng, J., Wang, D., Liu, Y.: Primary research of fishery big data and application technology in China.Shangdong Agricultural Sciences 48(10), 152–156(2016).
12. Yu, Z.: Review of fishery big data.Journal of Anhui Agricultural Sciences 45(9), 211-213(2017).
13. Li, D., Yang, H.: State-of-the-art review for internet of things in agriculture.Transactions of the Chinese Society for Agricultural Machinery 49 (1) ,1-20(2018).
14. Duan, Q., Liu, Y., Zhang, L., Li, D.: State-of-the-art Review for Application of Big Data Technology in Aquaculture.Transactions of the Chinese Society for Agricultural Machinery 49(06), 1–16(2018).
15. Yang, P., Lin, J.: A Scheme for Massive Unstructured Iot Data Processing Based on MongoDB and Hadoop. Microelectronics & Computer 35(04),68-72(2018).
16. Wang, Y., Chiang, Y., Wu, C., Yang, C., Chen, S., Sun, P.: The implementation of sensor data access cloud service on HBase for intelligent indoor environmental monitoring.In: 15th International Symposium on Parallel and Distributed Computing (ISPDC),IEEE, pp. 234-239(2016).
17. ECharts,https://echarts.baidu.com, last accessed 2019/2/1.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22744-9_50