Analysis of the Application of Network Data Configuration Management in Universities Based on Internet Platforms

Liu jingnan, Lv haiyang, Sun qiaoyan
University of Indonesia
Indonesia
Hghfyret454@outlook.com

ABSTRACT: *With the rapid development and popularization of Internet technology, the management of network data configuration in universities faces new challenges and opportunities. The application of university network data configuration management based on Internet platforms has emerged, aiming to improve the efficiency and quality of university network data configuration management. This article will analyze this application, and explore its advantages, challenges, and solutions. University network data configuration management based on internet platforms can quickly process and store a large amount of data, improving the efficiency of data configuration management. This application can be accessed anytime and anywhere, making it convenient for managers and users to access the required data anytime and anywhere. Data configuration management through internet platforms can reduce manpower and material investment, as well as management costs.*

Keywords: Internet Of Things, Big Data, General Platform

Received: 18 March 2023, Revised 10 July 2023, Accepted 29 July 2023

DOI: 10.6025/jisr/2023/14/3/57-63

1. Introduction

The Internet of Things was first proposed by American university professors in the study of RFID in 1999. Internet of Things, based on the concept of the Internet, extends its client side to any item or object through radio frequency identification (RFID), infrared sensors, global positioning systems, laser scanners and other information sensing equipment, and connects anything to the Internet according to an agreed agreement, so as to carry out information exchange and communication and realize intelligent identification, location, tracking, monitoring and management. Cloud computing has always played an important role in the history of Internet of things technology. As the most basic and important way in big data processing, cloud computing provides reliable management and analysis support for the big data generated by the Internet of Things. The number of global sensor nodes is expected to reach 20 billion in 2020 and will produce an unprecedented scale.

Compared with traditional data, cloud computing’s high lateral scalability can flexibly expand the computing power of the system according to specific requirements and can adapt well to the processing needs of the big data of the Internet of Things. Therefore, on the basis of the current cloud computing technology, it is of great practical value to implement the data processing platform of the Internet of Things, which also has great significance for the development of Internet of Things technology.

2. State of the Art

The German company *ParStream* is the online analysis service of the Internet of Things big data, the first industry-class online analysis platform for dealing with mass and high-speed data in the Internet of Things [1]. This platform can help the company obtain the analysis results from big data more quickly and thoroughly, thus supporting the company’s decision-making and system analysis. As a result of good performance and good user experience, the company’s products have been spread in Europe, the United States and other countries and regions [2]. Domestic is relatively backward in this regard, and there is no national-level related projects. There are online data analysis platforms, such as Baidu, music Federation, Ucloud and so on, providing sensor data reception, online data analysis and results display and other common functions based on HTTP. However, our country is still weak in terms of performance and user experience, and there is no relevant open-source technology framework on the market [3]. Alibaba, the country’s largest cloud computing application enterprise, launched the “flying platform” in December 2013. Innovative exploration has been carried out from the angles of product, price, service and cooperation with third parties. Significant breakthroughs have been made in related technologies, thus giving Chinese companies a universal computing platform to rival a single group of 5000 servers that can only be achieved by top companies worldwide, such as Google and Facebook [4]. In addition, because of its technical accumulation in search, Baidu Inc. also has strong strength in the cloud computing infrastructure and massive data processing and has gradually opened the IssS, PaaS SaaS and other multi-level platform services. Baidu cloud OS, with individuals as the centre, organise data and applications, supports Webapp, and provides the function of App pages [5].

3. Methodology

The system is a data processing platform of the Internet of Things based on cloud computing, which implements the process of data reception, storage, processing and display, and the two development interfaces. Users can monitor the sensor node status and simple configuration through the Web interaction community and enjoy the powerful storage and computing capabilities cloud computing provides [6]. At the same time, after fully considering the special needs of users, the system encapsulates the various kernel libraries used in the platform development. Moreover, based on the PHP language, which is easy to learn, users can quickly integrate the customised functional modules in the system according to their own needs. The main services provided by the system are the Web server and the TCP server, among which, the Web server provides users with cross-platform application configuration and application data display capabilities [7]. However, the TCP server provides the receive service of the sensor data. Cloud computing clusters are responsible for data processing in the system. The “middle layer” of the system is responsible for the interaction among the various modules of the system, which is the middleware of each module and implements a *Redis-based* message queue. Through the message queue, the module realises the sharing of data and the call of service and simultaneously reduces the degree of coupling. There is only a need to define the interaction protocol and data interface without knowing the underlying implementation details to enhance the system’s maintainability. The design of the system data storage and processing module is at the bottom of the whole system and is responsible for the storage and calculation of the core data [8]. The intermediate layer interacts directly with this layer. The intermediate layer delivers the computing task to the Hive module in the cluster module, and after the Hive module gets the task information, after compiling and parsing, the task is converted to MapReduce computing and is executed in Hadoop, and the execution results are obtained. Another important role of this layer is data upload [9]. The intermediate layer issues the data upload message to the cluster layer at the right time, and the data upload module is responsible for processing the message. After receiving the message, the module extracts data from the cache in the middle layer. It uploads it to the file system *Hdfs* of the cluster to complete the data persistence and provide the calculation for the Hive module. The specific architecture diagram is shown in Figure 1.

The layer only exposes two external interfaces to the upper layer, the task delivery interface and the data upload interface and correspond to two modules, namely, the Hive module and the uploader module. There are three sub-modules in the Hive module:

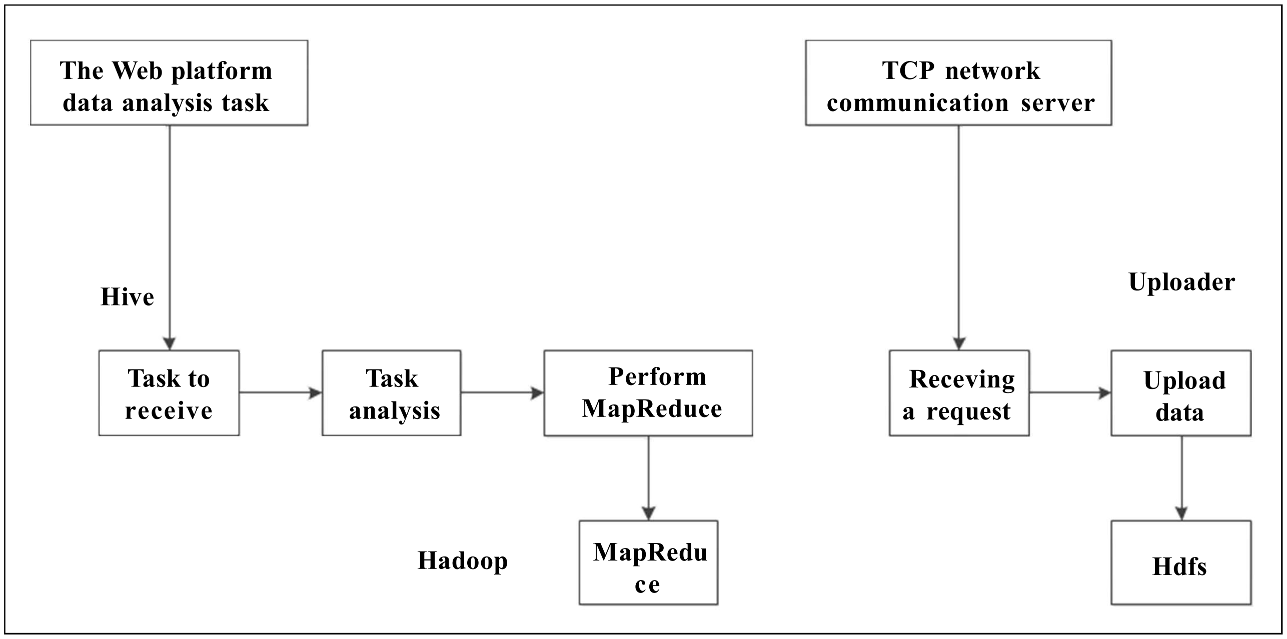

Figure 1. System architecture design

• **Task acceptance:** the interface call is exposed externally to obtain detailed information about the task.

• **Task analysis:** the main task of the task is the SOL query, which is responsible for converting *SQL* queries into MapReduce programs [10].

• **Task execution:** the MapReduce program is submitted to Hadoop for execution, and the task execution results are obtained so as to inform the task status of the intermediate layer.

In the Uploader module, there are two sub-modules:

• **Request receives:** interfaces are exposed externally to capture data upload events.

• **Data upload:** after being activated, the data is automatically extracted from the cache in the middle layer and uploaded to the designated location of *Hdfs* for persistent storage [11].

• The *TCP* server module design was developed using PHP. PHP itself is not good at developing high-performance network communications servers. Still, with the latest extension library *Swoole*, *PHP* can also carry out more complex large-scale network application development. The primary function of this module is to deal with terminal node requests, establish *TCP* connections, and handle terminal data upload, which is an important function module of the platform [12]. The module is divided into connection maintenance, engineering management, data frame management and other modules. A good encapsulation is carried out among the sub-modules. Thus, users can use the system's most basic functions and build more powerful functional modules according to their own requirements. The module is mainly divided into three layers: the *TCP* connection pool of *Swoole* is in the outermost layer, the application layer of the module is in the middle layer, and the middle layer is at the bottom. The main functions of each layer are as follows:

1. The SwooleTCP connection pool is maintained primarily by *Swoole* itself. When the *Swoole* receives the connection request, it creates a new connection in the connection pool, starting with the server *onConnect* event. In the *onConnect* event, the connection is passed to the next layer in detail according to the port number [13]. Another function of this layer is to update the status of the TCP server to the intermediate layer in real-time, and other system structures can obtain the server's current state through the middle tier.

2. Application layer: when each connection module receives a connection request, the connection information is stored first to achieve communication in the connection module. Secondly, the frame resolution module of the application module is invoked to parse the data frame. The parsed result object array is then stored in the frame queue of the connection module. Finally, the system provides the user with the end of the main process [14]. Then, the method exposed to the outside world can be run. Users extend their application modules to rewrite the method. In the run method, the user can connect the array objects in the module frame queue and also send the data to the sensor connection through the connection module. At the same time, to facilitate the operation of the user, this operation implements most of the commonly used methods, such as the encapsulation of intermediate interaction method, encapsulation of interaction method with database, sending mail and text messages and other functions. In essence, all functions of this module are carried out around the method of operation, and the purpose is to prepare data for the running method and also to facilitate the scheduling of the whole system. The built-in applications are on the same floor as the common applications, which are essentially the same as user applications. The difference is that these applications are protected, the access of which is controlled by permissions. The most important of these applications is the management process of the server, which can remotely start, restart, or close the server by modifying the application and issuing other important instructions to the server. In addition, since *Swoole* is implemented entirely by PHP, the memory of the global variable can be easily resident. Therefore, a global registration tree is implemented in the system, and the common factory design patterns and singleton design patterns are combined to implement application-level services, thus reducing the overhead of building and destroying objects at each time. When the application is activated for the first time, the memory is resident, containing links and data frames cached by each link so as to implement object services and speed up processing.

3. The middle layer is a part of the layer responsible for interacting with the TCP server. The main functions of this section are: the data frame submitted by the run method is cached, and the uploaded *Hdfs* event is triggered when the configured threshold is reached; the running parameters and current status of the server are recorded in real-time; it can be used as data sharing between each connection module and application module [15].

4. Result Analysis and Discussion

TCP data-receiving module is an important module of the platform, which is responsible for the reception and pretreatment of the sensor data, and directly determines the network communication performance of the platform. Through testing, the performance transformation rule of the TCP server was explored, and then the best configuration of the TCP server in the current testing environment was summarized.

An important indicator for a network communication server is response time, the time it takes for the server to process user requests; in this platform, it is the time that the system processes the data frame. It is assumed that there are n sensors in the specified period; at the same time, the data frame of size s Byte is uploaded to the server. After receiving and processing the data, the server returns it as it is. The size is R Byte. Because the server consumes time on data processing: $r < n*s$ That is, the data returned by the server will be less than the sum of data sent by the n sensors. It is not difficult to say that when n approaches infinity, the server's processing time approaches infinity, while the server returns processing data, and the R approaches 0. So the server data processing ratio is $P = r/n*s$. The closer the P is to 1, the better the server performs. In the testing period, the data sent by the sensors is processed, and when P is close to 0, the server pressure is too large, causing congestion or even downtime.

The main performance parameters of the target server were: 4 cores CPU, power 1200MHZ, 64 bit processor, 128G memory. After receiving the data, the server can parse the data and return it directly, and the performance will be better than the actual production environment.

In the experiment, when n was very small, each time increased 10 times, when the n value was bigger, the increment slowed down, 1, 10, 100, 200, 500, 600, 700, 800, 900, and 1000 were selected. To minimise the impact of each set of data connections and server fluctuations, the test time was 30 each time. The test results are shown in Table 1.

The server's processing data ratio within the sending time interval (100ms) decreased gradually with the increase of the concurrent amount. When concurrency increased to 1000, less than half of the data was processed in time. So, in the current environment, the best control of server concurrency was within 500. However, it was necessary to point out that sensor data acquisition rates rarely reached 100ms levels and thousands of concurrent amounts simultaneously in the actual generation environment. When the acquisition frequency increases to 1s level, in theory, the amount of contemporary can reach about 5000. In addition, the data processing ratio parameter selected in this test was not particularly appropriate, which described the processing ratio at "the same time". In the actual production environment, the delay of 100ms is acceptable, and the concurrency can also be doubled to around 5000 in theory. To sum up, although the performance of TCP servers is gradually declining with the increase of concurrency, it is enough to deal with most actual production environments.

Test number	**Concurrent quantity *n***	**Send data (Byte)**	**The server returns data (Byte)**	**Data processing ratio P**
1	1	798	798	100.00%
2	10	8445	8445	100.00%
3	100	81297	81279	99.97%
4	200	162057	159924	98.685
5	500	264516	246390	93.14%
6	600	251553	226368	89.98%
7	700	275304	202108	73.41%
8	800	277338	196623	70.89%
9	900	424434	219447	51.70%
10	1000	2118258	1002174	47.31%

Table 1. Test results for the server

Hadoop distributed cluster is the base of data storage and processing in this platform. Aiming at the actual generation environment of the system, an experiment was designed and tested, and various characteristics under the Hadoop fully distributed cluster were verified. The functions of data upload, view, dynamic additions and deletions of the server nodes and cluster status in the fully distributed mode of the Hadoop cluster were tested.

For convenience, in the experiment, the server as the main node was the entity machine, and the other two machines as the slave nodes were all virtual machines, among which, the host master (192.168.1.129), the two Slaves; Slave 1 (192.168.1.116), and the slave 2 (192.168.1.116). There was only qualitative analysis, so the virtual machine's configuration was relatively low.

After installing and configuring the three machines, the command *start-dfs.sh* was entered on the master node to start the cluster. After the start of the process, the web operation interface through the master node is shown in Figure 2. As can be seen, there were two active nodes, slave1 nodes and slave2 nodes, respectively. The data was uploaded to the cluster. The test text file test was uploaded to Hadoop, the command bin/Hadoop fs-put test/test was entered, and then, through the web interface, the file was seen.

One of the major advantages of Hadoop is its ability to increase capacity flexibly, that is, to add and delete nodes to the cluster smoothly. In the experiment, the Slave1 node was added to the blacklist of the master node and removed from the cluster. Then it was deleted from the blacklist and then added back to the cluster. Throughout the process, there was no need to restart or close the cluster, and other nodes were not affected. The slave1 node was added to the blacklist of the master node; that was, in the data node-deny-list file, the slave1 was added. Then, the command *bin/hdfs dfsadmin-refreshNodes* was entered so that the cluster re-checked each node, and the web interface was refreshed. The slave2 was still in the In-Service state, while the slave1 was in the Decommissioned state. As for recovery, it was deleted from the blacklist, and then the cluster section was refreshed just as before. Then, tests were then performed, and nodes were added to the cluster. To facilitate testing, the cluster was started with only two nodes, master and slave1. Then, slave2 was added to the slaves' list of masters, and then the data node process was started in slave2. The cluster automatically checked the slave2 node and then used its resources.

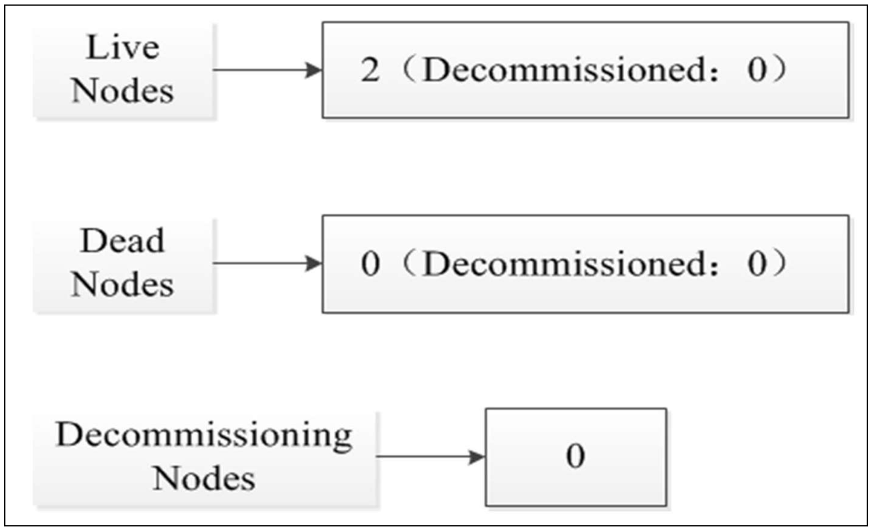

Figure 2. Hadoop cluster status

5. Conclusions

In this paper, the implementation of a universal management platform for Internet of Things data was studied. The functions and characteristics of several key technologies were analyzed. Combined with the analysis and abstraction of the main part of the data management of the Internet of Things, in the main links of data access, storage, processing and display, the configurable management of the Internet of Things data was realized, thus facilitating the second development of the underlying API. At the same time, the main functions of the platform were tested, and the application of the actual project was realized. The main functions of the platform and the characteristics of the second development were shown in detail. Through the research of this paper, some conclusions were obtained as follows: the high-performance TCP server was developed based on PHP, and the functions of data analysis, cache and pretreatment were designed and implemented; in order to solve the effective sharing of data between different servers and different layers, an intermediate adaptation module was developed; in order to enable users to build IOT applications and access sensor data through simple configuration, the platform implemented the online service function, including the application of real-time data monitoring, data analysis and results display module. In order to further enhance the platform's functional characteristics and performance, research and optimization can be carried out in other ways in the future, such as reorganizing the structure of API and reducing the learning cost of the second development.

References

[1] Sun, Q. B., Liu, J., Li, B., et al. (2010). Internet of things: concepts, architecture and key technology research review. *Journal of Beijing University of Posts and Telecommunications*, 33 (3), 1-9.

[2] Liu, Y. (2013). *Internet of things in the era of big data. High Technology and Industrialization,* 5, 10-15.

[3] Shvachko, K., Kuang, H., Radia, S., et al. (2010). The Hadoop Distributed File System. *IEEE Symposium on Mass Storage Systems & Technologies*, 3, 1-10.

[4] Thusoo, A., Sarma, J. S., Jain, N., et al. (2009). Hive - A Warehousing Solution Over a Map-Reduce Framework. V*LDB Proceedings of the VLDB Endowment*, 2(2), 1626-1629.

[5] Lloyd, A. D., Sloan, T. M., Antonioletti, M., et al. (2013). Embedded systems for global e-Social Science: *Moving computation rather than data. Future Generation Computer Systems*, 29(5), 1120-1129.

[6] HDFS. (2007). The Hadoop Distributed File System: *Architecture and Design. Hadoop Project Website*, 11, 1-10.

[7] Song, S. D., Guo, F. (2002). Java-based WEB database connection pool technology research. *Computer Engineering and Application,* 38 (2), 201-203.

[8] Ren, Z. F., Zhang, H., Yan, M. S., et al. (2004). It summarized research of MVC pattern. *Computer Application Research*, 21(10), 1-4.

[9] Cheng, C. R., Liu, W. J. (2009). High cohesion and low coupling in software architecture to build. *Computer System Application*, 18(7), 19-22.

[10] Thusoo, A., Sarma, J. S., Jain, N., et al. (2009). Hive - A Warehousing Solution Over a Map-Reduce Framework. *VLDB Proceedings of the VLDB Endowment*, 2(2), 1626-1629.

[11] Lin, C., Hu, J., Kong, X. Z. (2012). *User Quality of Experience (QoE) Model and the Evaluation Method of Review. Journal of Computers*, 35(1), 1-15.

[12] Castro, O., Ferreira, H. S., Sousa, T. B. (2014). *Collaborative Web Platform for UNIX-Based Big Data Processing. Lecture Notes in Computer Science*, 3, 199-202.

[13] Shan, J. H., Jiang, Y., Sun, P. (2005). Software testing research progress. *Journal of Beijing University: Natural Science Edition*, 41(1), 134-145.

[14] Dong, Y. Z., Zhou, Z. W. (2006). Based on X86 architecture system virtual machine technology and application. *Computer Engineering,* 32(13), 71-73.

[15] Li, D. L., Chen, R. (2009). WebSocket in Web research in the field of real-time communication. *Computer Knowledge and Technology*, 9, 19-20.