Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2020.010204
ISSN 2790-0916 Vol. 1, Issue 2: 28-36

Consistent Hash Algorithm in Distributed Monitoring System

Adityan Kumare*

Myanmar Institute of Information Technology, Myanmar

**corresponding author*

Keywords: Consistent Hash Algorithm, Distributed System, Monitoring System, Data Management

Abstract: The cloud monitoring system allows administrators to know the historical performance of each component of the platform at any time and control the usage of various resources. Monitoring can remind operation and maintenance managers in a timely manner when a fault occurs, quickly find problems, and better solve errors. The purpose of this paper is to study the application of consistent hash algorithm in distributed monitoring system. The architecture and design concept of distributed monitoring are analyzed, and a set of good distributed performance monitoring system of cloud platform is developed according to the characteristics of the rapid development of cloud platform. The whole system can be divided into data acquisition unit, real-time alarm unit, historical data storage unit and global control unit. The specific implementation process of the main functional units is given. Finally, the system is tested. The experimental results show that the consistent hash algorithm has a good balance effect in the distributed monitoring system.

1. Introduction

It is very important to build a distributed monitoring system in the cloud platform environment. The system needs to be able to scale horizontally with the development of the cloud platform, without running bottlenecks, and the deployment configuration should be simple enough [1-2]. It can monitor the usage of various resources, locate the occurrence of various anomalies in real time, and notify fault alarms in various ways. At the same time, historical data can be persistently stored, which can be used to analyze and understand the historical operation of various components of the platform [3-4].

In the era of Internet information, thanks to the rapid changes in technology and the surge in the amount of information, the development of Internet companies is in full swing. How to survive and

develop in an increasingly competitive environment is a challenge that every Internet company needs to face. [5-6]. Mettler M proposes a decentralized monitoring architecture for large-scale multi-block MPSoCs. To minimize the performance and power consumption overhead of RV, a lightweight and non-intrusive hardware solution is proposed. It features a new dedicated trace interconnect that assigns and classifies detected events based on timestamps. Each tile monitor has a consistent view of globally ordered event traces that can verify the behavior of the target application using logic and timing requirements [7]. Kazemi Z implements and embeds advanced monitoring algorithms directly into the DCS structure for the first time. Thus, the need for an additional computer connected to the DCS will be eliminated, which brings several advantages from a performance perspective. In the proposed method, the advanced monitoring technique is first simplified and divided into several functions. Next, Dynamic Link Libraries (DLLs) are created and used to quickly execute different functions in DCS. Finally, using the generated DLL and defining functions in WinCC, the monitoring algorithm is executed in real time [8]. Building a low-cost, high-performance, scalable distributed monitoring system is of great significance for providing efficient and stable data forwarding, storage, and analysis services [9].

With the continuous increase in the area of the monitoring system and the complexity of the business, the traditional single-server architecture can no longer meet the system development requirements. Starting from the basic frame of the monitoring system, this paper proposes a fully integrated, hybrid architecture distributed system, and designs a load balancing strategy according to the characteristics of the monitoring system itself. The general definition and performance characteristics of the distributed server cluster of the monitoring system are designed. All widgets can be deployed on different devices in a distributed manner, and can be scaled horizontally through cluster deployment. The monitoring system of distributed server cluster architecture proposed in this paper has good research significance and technical value for improving the overall performance and service quality of the monitoring system.

2. Research on the Application of Consistent Hash Algorithm in Distributed Monitoring System

2.1. Distributed Framework

The distributed structure of the decentralized and fully symmetric architecture uses the idea of the consistent hash algorithm to locate the physical location of the file in the storage node, thereby canceling the role of Master. As shown in Figure 1, the entire system has only one role as a storage node, and does not distinguish between metadata and data blocks. All data requests are obtained through the calculation of consistent hashing. This architecture avoids single point of failure and makes data more evenly distributed on different nodes [10-11].

In this paper, the consistent hash algorithm of virtual nodes is adopted, which introduces the concept of virtual nodes and dynamically determines the number of virtual nodes according to the current state of the server, so as to achieve the purpose of evenly distributing server resources [12]. Using the consistent hash algorithm of virtual nodes, the mapping relationship between the ordinary consistent hash algorithm and the key is divided into two steps:

1) Calculate the mapping relationship between the key and the virtual node;

2) Query the real physical node corresponding to the key according to the mapping relationship between the virtual node and the physical node [13-14].

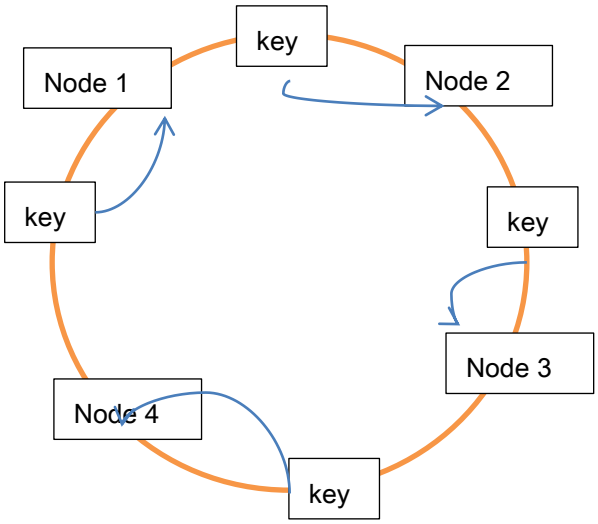

Figure 1. Decentralized structure

In the aspect of the corresponding allocation between virtual nodes and physical nodes, the concept of dynamic weight is introduced. The dynamic weight of a server s is determined by its current load status [15]. Select the server with the most load as the baseline wbase., and other servers to be selected are compared with wbase to obtain wn, and the specific calculation is shown in formula (1).

$$w_n = w_{base}(1-P_n)/(1-P_{base}) \tag{1}$$

Assign n*E (where E is the average multiple of virtual nodes) nodes to physical servers {S1, S2,..., Sn} according to weights {W1, W2,..., Wn}, and their corresponding virtual nodes The number is {N1, N2,..., Nn}. Since the hash has good dispersion, the interval ∂ of the virtual node corresponding to the physical machine Sm is shown in formula (2):

$$\partial = [\sum_1^m N_i, \sum_1^{m+1} N_i] \tag{2}$$

2.2. Overall Architecture Model

Figure 2 shows the overall architecture of the entire distributed monitoring system:

Agent: deployed on all nodes that need to collect data. This module periodically collects the built-in general monitoring indicators of the node where it is located, and runs the collection plug-in to collect extended indicators [16-17]. And it can realize automatic configuration without manual intervention and report data to the Deliver node for subsequent data forwarding operations. The module has high execution efficiency and will not affect the performance of the monitored equipment.

Deliver: Use the consistent hashing algorithm to distribute the indicators collected and reported by all agents evenly and efficiently to the real-time alarm module and the historical data storage module for subsequent alarm judgment and data persistence. In order to achieve high availability of historical data storage, the system implements a total of two consistent hashing algorithms to

achieve double writing of data. When a cluster fails, it will not affect the reading of the entire system data [18].

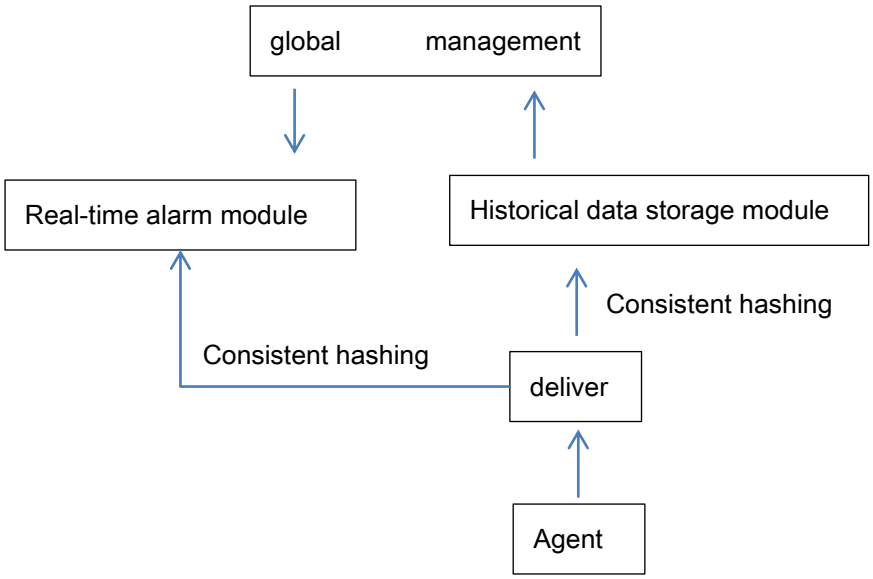

Figure 2. Overall architecture diagram

3. Investigation and Research on the Application of Consistent Hash Algorithm in Distributed Monitoring System

3.1. Real-time Monitoring of Data

Fig. 3 is the realization frame diagram of the business data real-time monitoring system.

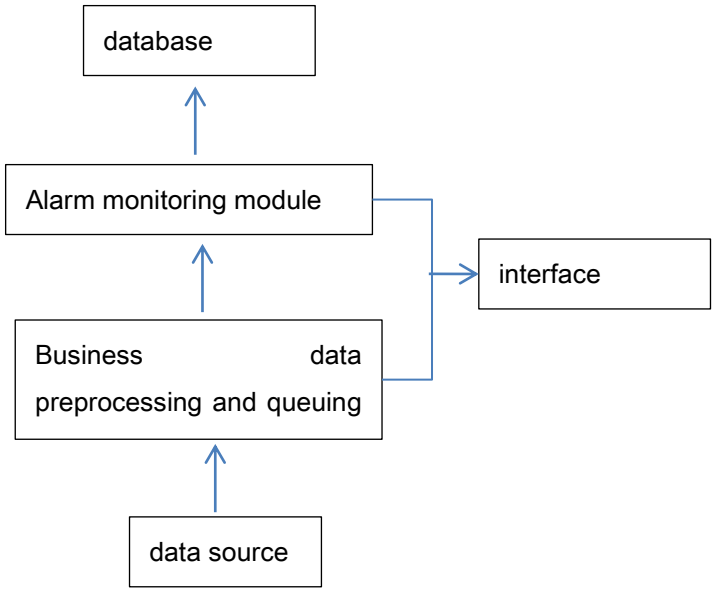

Figure 3. Sub-modules of the real-time monitoring system for business data

In this paper, the data source is generated by the program, and multiple data sources are introduced to generate data at the same time. Each data source is generated in the way of Poisson distribution. See the generation method. When the business data in the ISN reaches the business layer through the transport layer, it is in the form of data frames. Therefore, the business data needs to be preprocessed when stored in the database or through the alarm detection module. As a distributed and parallel network, the ISN will have various The business data of the level arrives at the business layer at the same time. The business data preprocessing and queuing module mainly completes the analysis of the generated data frame, and realizes the interpretation of the messages in the data frame according to the regulations of the business layer on the business data frame. After the interpretation is completed, according to the priority carried in the data frame, it enters the priority queuing module, which is mainly used to control the arrival order of data and perform sequential uploading. At the same time, the data that has been analyzed will be displayed on the interface in real time, that is, the real-time monitoring function of the whole network data will be realized.

The alarm detection module detects and filters the uploaded data according to the information carried by the arriving data and pre-agreed rules. If the data is identified as alarm data, an alarm is generated and an administrator is notified. At the same time, the alarm data will be displayed on the interface in real time at the same time, that is, the real-time monitoring function of the alarm data is realized.

3.2. Performance Test

Due to the limited number of SIP devices, this paper uses the simulated terminal written to test the performance of the distributed server cluster system. The simulation terminal program is divided into a SIP client simulation program and a SIP device simulation program. Both of them have the functions of registration, login and heartbeat information. The heartbeat information is sent once every 20s. Simulates end devices to send real-time requests. Since the load balancing algorithm of the server cluster is a static algorithm, the SIP device simulation program is set to not send data to the server for the convenience of testing; in addition, the authentication function of the central signaling control server is also removed. In this paper, 300 SIP client simulation programs are used to initiate real-time requests to 300 SIP devices. After 2 minutes, the number of registered SIP terminals and the number of SIP session tasks on the central signaling control server and service data server are respectively checked.

4. Implementation and Analysis of Consistent Hash Algorithm Applied in Distributed Monitoring System

4.1. Consistent Hash Implementation

In the Deliver workflow, the Agent sends the built-in monitoring indicators and extended monitoring indicators to the Deliver module through RPC and HTTP methods. The first job for Deliver is to parse the monitoring indicators, calculate their hash values, and rearrange the relevant data. Then put it into the queue of the corresponding receiving node for subsequent sending.

First, the Deliver module calls the initHashRing() function to initialize the consistent hash ring. This function internally passes the judgment cluster and storage cluster address information in the configuration file to the two sub-functions compareHashRing and storageHashRing. Take compareHashRing as an example, after receiving the compareAddrs array, take out the IP address of

each backend receiving node, and convert it into a string type for hash value calculation. The calculation of the hash value uses the 32-bit cyclic redundancy check algorithm crc32. Each hashkey returned by the calculation is stored in an array and sorted, and each backend IP address and corresponding hashkey are stored in the hashkeyMap.

The next step will start generating the hashRing. The generation of the hash ring calls the open source consistent library of stathat company. The library efficiently implements methods related to consistent hashing algorithms, and deeply encapsulates the underlying implementation of related data structures. First, use newConsistentHashNodeRing() to generate a blank hash ring, specify the ring size and the virtual node multiple replicasNum, and use the setNode() method to add the above-generated hashkey array elements to the hash ring one by one. At this point, the operation of adding all Compare backend instances to the consistent hash ring is completed. Storage instances are implemented the same way.

After initializing the hash ring operation, Deliver enters the data forwarding phase. Deliver parses each received monitoring indicator, obtains the corresponding endpoint value, metric value, and tag value, and combines the three to generate a string representing the monitoring indicator. Also perform 32-bit cyclic redundancy check on this key to get a unique hash value hashkey, call the getNode() method to get the hashkey value of the nearest node on the hash ring, and finally read the hashkeyMap to get the corresponding back-end node IP address, and send this monitoring indicator data to the corresponding sending queue.

4.2. Performance Analysis of the System

Table 1 shows 300 SIP device simulation programs and 300 SIP client simulation programs, and a total of 600 SIP simulation terminals are finally registered with the central signaling control server distribution diagram. Because the configurations of the three central signaling control servers are basically the same, the number of registered SIP terminals on Node1, Node2, and Node3 in the figure is also relatively average. The test results show that the central signaling server cluster designed in this paper has a good load balancing effect.

Table 1. Analysis of load balancing effect

server	The number of SIP sessions of the service data server	Number of registered SIP terminals in the central signaling control server
Node1	211	163
Node2	189	158
Node3	200	179

Figure 4 shows the distribution of 400 SIP sessions on the service data server. It should be noted that the 400 SIP sessions here are counted based on the SIP client simulation program. Since the SIP client simulation program randomly sends real-time requests to the 400 SIP device simulation programs, there are 400 SIP device simulation programs. Some did not receive a request, while the rest of the SIP emulator took on at least one request. As can be seen from the figure, the number of SIP sessions on the three service data server nodes is roughly the same, indicating that the service data server cluster in this paper also has a good balance effect.

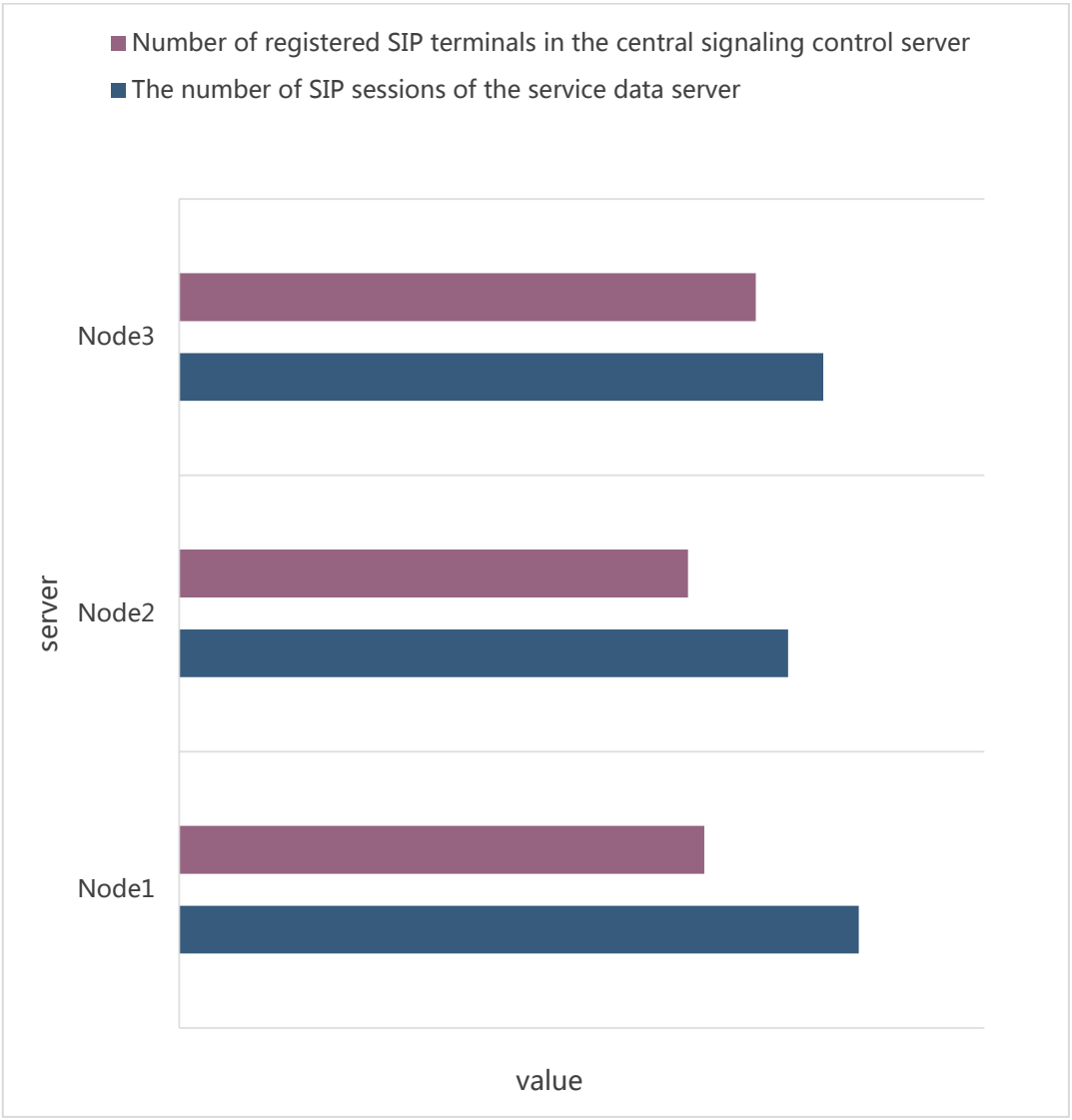

Figure 4. Session distribution map and terminal registration distribution

5. Conclusion

This paper gives the design, implementation and performance analysis of the data monitoring system, and there are some follow-up work that can be optimized: For the performance analysis of the single-node system and the over-node system, it is necessary to increase the analysis parameters. This paper only analyzes the load rate at present., you can further study such as packet loss rate, data migration rate, etc. When analyzing the load balancing scheduling algorithm, only the load balancing scheduling algorithm of the consistent hash algorithm is used, which can increase the comparison of some other methods and improve the persuasiveness of the article. When implementing the data monitoring system, due to the limitation of experimental conditions, the scale and data volume of the data monitoring system built in this paper are small, and the real-time/timing synchronization module realized may have potential data consistency problems. The alarm detection module has not been implemented yet and needs to be optimized later.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Katsaros D. Distributed ledger technology: the science of the blockchain (2nd ed.). Computing reviews, 2018, 59(11):596-597.

[2] Mendelson G, Vargaftik S, Barabash K, et al. AnchorHash: A Scalable Consistent Hash. IEEE/ACM Transactions on Networking, 2020, PP(99):1-12.

[3] Thaiyalnayaki S, Sasikala J, Ponraj R. Indexing Near-Duplicate Images In Web Search Using Minhash Algorithm. Materials Today: Proceedings, 2018, 5(1):1943-1949. https://doi.org/10.1016/j.matpr.2017.11.297

[4] Jose C. Document Security System Using Improved Hash Algorithm on Pre-processing Operation. Journal of Advanced Research in Dynamical and Control Systems, 2019, 11(11-SPECIAL ISSUE):972-978.

[5] Kumar M S, Pvrd P, Rao. Advanced SHA-256 Algorithm for Device to Device Communication. International Journal of Advanced Science and Technology, 2020, 29(7):1189-1191.

[6] Santra R, Obermeyer M. A first encounter with the Hartree-Fock self-consistent-field method. American Journal of Physics, 2020, 89(4):426-436. https://doi.org/10.1119/10.0002644

[7] Mettler M, Mueller-Gritschneder D, Schlichtmann U. A Distributed Hardware Monitoring System for Runtime Verification on Multi-Tile MPSoCs. ACM Transactions on Architecture and Code Optimization, 2020, 18(1):1-25. https://doi.org/10.1145/3430699

[8] Kazemi Z, Safavi A A, Pouresmaeeli S, et al. A practical framework for implementing multivariate monitoring techniques into distributed control system. Control Engineering Practice, 2019, 82(JAN.):118-129.

[9] Ciminello M. Distributed Fiber Optic for Structural Health Monitoring System Based on Auto-Correlation of the First-Order Derivative of Strain. IEEE sensors journal, 2019, 19(14):5818-5824. https://doi.org/10.1109/JSEN.2019.2903911

[10] Salih H S, Egorov S Y. Development Of A Monitoring System For Scheduled Works At Distributed Facilities. Vestnik Tambovskogo gosudarstvennogo tehnicheskogo universiteta, 2020, 26(1):056-063.

[11] Mohammed A, Hu B, Hu Z, et al. Distributed Thermal Monitoring of Wind Turbine Power Electronic Modules Using FBG Sensing Technology. IEEE Sensors Journal, 2020, PP(99):1-1.

[12] Ciminello M. Reliability of structural health monitoring system based on distributed fiber optic and autocorrelation of the first order derivative of strain. IEEE Sensors Journal, 2019, PP(99):1-1.

[13] Monsberger C M, Lienhart W. Design, Testing, and Realization of a Distributed Fiber Optic

Monitoring System to Assess Bending Characteristics Along Grouted Anchors. Journal of Lightwave Technology, 2019, PP(99):1-1. https://doi.org/10.1109/JLT.2019.2913907

[14] Masouros D, Xydis S, Soudris D J. Rusty: Runtime interference-aware predictive monitoring for modern multi-tenant systems. IEEE Transactions on Parallel and Distributed Systems, 2020, PP(99):1-1. https://doi.org/10.1109/TPDS.2020.3013948

[15] Alfredo, Güemes, Antonio, et al. Simulation Tools for a Fiber-Optic Based Structural Health Monitoring System. Transactions of Nanjing University of Aeronautics and Astronautics, 2018, v.35(02):5-11.

[16] Charapko A, Ailijiang A, Demirbas M, et al. Retroscope: Retrospective Monitoring of Distributed Systems. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(11):2582-2594. https://doi.org/10.1109/TPDS.2019.2911944

[17] Abdurakhmanov A S, Fedorova V A. Intellectual mobile system for monitoring environment in the premises. Radio Industry (Russia), 2018, 28(4):41-46.

[18] Sancho J I, Almandoz I, Barandiaran M, et al. Scalable Wireless Wearing Monitoring System for Harsh Industrial Environment. IEEE Transactions on Industrial Electronics, 2020, PP(99):1-1. https://doi.org/10.1109/TIE.2020.3053892