Visual Web-side Monitoring Platform Implementation based on Layui+SpringBoot Architecture

Jian Xiang[1]
[1]School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou, China, 310023
Email address: freenyspi@gmail.com

***Abstract*—*The system as a whole adopts Layui+SpringBoot architecture, MySQL as the database, and the system structure is divided into agent side, client side, server side and visual web side. The monitoring data is collected by deploying the agent side to the server or application to be monitored and sent to the server side for unified processing. The server side is responsible for processing and storing the monitoring data and generating alarms. The visualization web side provides a visual management platform for operation and maintenance personnel to monitor and configure the monitoring data and related alarms. In summary, this system will provide a more complete monitoring system for the process: from data acquisition to data processing, sending alarms and visual display. And the detailed design for data acquisition, processing, and alarming.*

Keywords*— *Layui, Springbot, Monitoring Platform, Web-side.

I. INTRODUCTION

The China Internet Development Report (2020) shows that by the end of 2020, the number of Chinese Internet users will reach 989 million and the Internet penetration rate will reach 70.40%.[1] The Internet penetration rate will reach 70.40%. This huge group of Internet users reflects the current Internet environment which is developing rapidly. The booming Internet environment has led to an increasing number of Internet enterprises, and the current Internet enterprises often have many product lines and business lines under their companies, and under each business line, there are many application services. These applications that are closely related to each other build the core of an Internet enterprise. Many services are deployed in different host clusters, which depend on each other to provide services to the outside world. These host cluster servers provide the deployment and development environment for the enterprise applications, and the online operation of the applications is often directly related to the profitability of the Internet enterprise. So how to ensure the stability of this huge online service is the key concern of enterprises. The challenge for many developers is to find out the abnormal situation of the online service in time, so as to minimize the impact of the system problems. To ensure the stability of the service, an indispensable point is the perception of service quality, which means that a stable and efficient monitoring system needs to be built. However, the current market lacks an integrated platform for efficient monitoring of online applications and the current operation of the servers where they are located.

II. CURRENT STATUS OF DOMESTIC AND INTERNATIONAL RESEARCH

Due to the rapid growth of Internet businesses, the need for enterprises to monitor their own applications has increased. A large number of monitoring tools, both commercial and open source, have emerged in the past few years, but no clear solution has yet emerged.

Small and micro Internet enterprises choose to directly use the monitoring service carried by the third party that provides the server for monitoring, or even access to the server to check the memory and other related conditions. The problem with relying on external cloud-based monitoring platforms is that the monitoring of different platforms cannot be connected and requires logging into different platforms to view the situation. The traditional way of information monitoring requires remote login to view the server, and in order to ensure that the disk has enough space to record monitoring information, it is necessary to check the memory and CPU information inside the local resource manager in time to do a good job of transferring and clearing the information.[3] In order to ensure that there is enough space to record the monitoring information on the disk, we need to check the memory, CPU and other information inside the local resource manager in time, and do a good job of transferring and clearing information.

Most enterprises choose to use open source frameworks for secondary development to achieve enterprise application and server monitoring, such as Nagios, which can only monitor service and host-related information, but cannot view the history, and cannot perform buried monitoring of business.[3] Prometheus achieves this by periodically capturing monitoring data from the monitored service via HTTP protocol, but because of this, Prometheus is more containerized monitoring and is deficient in the breadth of monitoring, while Zabbix is a mature, highly integrated commercial monitoring product with open source, distributed, and out-of-the-box features, and provides a good Web front-end interface for system-level availability and performance monitoring [4]. However, its secondary development is more complex and technically demanding for the average enterprise.

And large companies with extensive expertise, such as Google, Facebook or Netflix, can develop solutions that fit their scale [5]. Such as Xiaomi, a domestic company that open-sourced its self-developed monitoring system OPEN-FALCON.

III. Research content and methodology

The purpose of this project monitoring platform is to solve the monitoring problems faced by the current development and operation and maintenance personnel, using SpringBoot + Layui technology architecture, mainly for real-time monitoring of applications, servers, databases and networks. It is mainly used for real-time monitoring of applications, servers, databases and networks. It can detect abnormalities and alert in time in the face of online applications and visualize the running status of servers, so as to achieve reasonable and efficient maintenance of the online environment, timely detection of online application abnormalities and timely alerting to avoid losses.

The overall front-end of the project uses the easy-to-use Layui + ECharts framework to achieve a visual display of the data. The back-end core uses the most popular SpringBoot framework, and the log management uses the most common SLF4J and Logback frameworks to achieve the collection and processing of operation logs. The system uses MySQL relational database for data storage and Alibaba Druid for connection pooling. the security involved in the monitoring system is implemented using SpringSecurity, SpringSession framework.

IV. System Implementation

A. Monitoring Client

The client is used to monitor various applications and needs to be integrated into the application that needs to be monitored to achieve monitoring of the application running status, JVM, application server and other related information. But the client only has the monitoring function, and the specific trigger needs to be triggered with other ways to trigger. The current O&M monitoring system can be triggered by the SpringBoot project relying on the SpringBoot proxy and configuring the annotation @EnableMonitoring at startup, or by running the server proxy.

The main functions of the client include business buried monitoring data collection, Java VM data collection, server data collection and heartbeat packet task for sending data to the server. These four functions are designed as task scheduler method, which is triggered when monitoring is needed to achieve the purpose of monitoring.

The specific implementation of business buried monitoring supports business monitoring of both normal Java applications and SpringBoot applications. When the application to be monitored relies on the SpringBoot proxy and the relevant code is injected in the code, it triggers a business buried monitoring task on the client side: a CPU/IO intensive thread pool is built according to the business specific needs of the monitored application. The business monitoring buried task can be configured to agree on the time interval between two buried monitoring points, thus avoiding exceptions of the same cause suddenly being triggered in large numbers in a short period of time, resulting in a large number of repetitive and meaningless alert messages in a short period of time.

The difference between CPU-intensive threads and IO-intensive threads is that IO-intensive threads use more time for context switching, resulting in low CPU utilization, so when creating a thread pool, the number of core threads needs to be set larger, while CPU-intensive threads are just the opposite, for CPU utilization is already high, the number of core threads can be set close to the actual number of threads.

The number of core thread pools in the project for both types is calculated as follows.

CPU-intensive blocking factor is relatively small: number of core threads = Ncpu / (1 - 0.2)

IO-intensive blocking factor is relatively large: number of core threads = Ncpu / (1 - 0.8)

The code for the construction of the thread pool is implemented as follows, for a CPU-intensive example.

The client's server monitoring task is mainly to collect information about the server it is on, and then send the server information to the agent when the server monitoring task is on.

The overall process of server monitoring task is as follows: firstly, the monitoring task is mobilized and opened, then it determines whether the server monitoring is opened, and if it is opened, it loads the configuration file information, gets the configuration about server monitoring, initializes the server related monitoring properties and server specific information. Then the Scheduled Executor Service multi-threading mechanism creates a thread pool, processes each monitoring task in the thread pool, schedules it according to the frequency specified in the configuration, builds a data packet in the timed schedule, stores the server specific information in the data packet and sends it to the agent together. Thus, real-time monitoring of the server is realized.

The specific server information obtained includes: operating system information, memory information, CPU information, network card information, disk information, and process information.

Java VM monitoring task is mainly when SpringBoot agent is triggered, the current operation of the monitored application is monitored, including: class loading information, memory information, GC information, JVM runtime information, thread information and other information.

Enterprise O&M monitoring platform contains several modules, in which the server agent and SpringBoot agent will be run on various cloud servers or everywhere physically, and some of these agents are only monitoring some abnormal information, and the frequency of abnormal information is low, then the agent is in a silent state most of the time, so how to ensure that the agent is running normally, and Then how to make sure that the agent is running normally and not offline by sending a heartbeat packet task to[9] .

By sending heartbeat packets to the server at regular intervals to notify the server that the agent is still running normally, and when the agent no longer sends heartbeat packets, the server will decide that the agent is offline.

So the processing flow of the task of sending heartbeat packets is similar to the Java VM monitoring task, but it lacks the task of determining whether monitoring is on or not, and the information in the packets sent to the server is just the basic information of the current application: application instance name, application id, application description, IP address, computer name, time, heartbeat frequency, etc.

B. Monitoring Agent Side

There are two types of monitoring proxies, the server proxy and the SpringBoot proxy.

The server proxy, as the name suggests, is used to monitor the server and provide the corresponding monitoring data to the monitoring server. The server agent relies internally on the client to implement the server information collection function. The types of data collected by the server proxy include: the application's own alert information, Java virtual machine information, and server information. The server proxy then sends heartbeat packets to the server to ensure that the current proxy is alive.

The overall process of sending monitoring information from the server agent to the server is as follows: firstly, the configuration of the server agent is modified, then it is packaged and deployed on the server to be monitored and run, and then the server agent collects the monitoring data through the dependent client and sends it to the server for processing and subsequent visualization.

SpringBoot proxy, when there are related SpringBoot projects need to be monitored, you can import this dependency, so as to achieve the monitoring of the application and business buried monitoring functions.

If you need to perform business burial-related monitoring, you need to inject code at the location that needs to be monitored, call the client-side business burial monitoring task, and the server-side receives alerts for further processing, so as to achieve business monitoring. The specific example is shown below

C. Server Side

The main tasks of the server side of the operation and maintenance monitoring project are: to receive the relevant monitoring data sent by the agent side of the customer and the client side of the processing and storage; to alert the generated alarms by email.

Since the number of agents and clients may be countless, but the number of database connections is online, the project does not design the database connection and storage at the client side, but chooses to connect to the database and perform data processing and storage work uniformly after the server side receives the monitoring data. The specific process is: the monitoring data sent by each agent and client to the server side is accepted by the server side, and then the data is processed and stored in the MySQL database according to the different types of monitoring data.

For the generated alarms, the alarms are processed asynchronously on the server side and emails are sent to the corresponding alarm recipients. The overall processing process is as follows: when the server side receives the alert packet, it will first make a pre-judgment to determine whether the alert conditions are met, and if so, it will improve the alert data and then store it in the database to generate the alert data. After that, it makes a logical judgment whether to send the alert, and if the conditions for sending the alert are met, it splices the alert emails and sends them, and then stores the sent email records into the database.

D. Alarm Configuration Module

Common alarm configurations have the ability to report time series at a configurable granularity, provide statistical aggregation capabilities (e.g., maximum, minimum, average), define thresholds, define alarms or actions to be taken in a given situation, and provide filtering capabilities.[10] The alarm configuration capability of this system is weak, and the configuration management module is subdivided into environment management, group management, monitoring configuration and alarm definition.

As the role of servers in an enterprise will differ and there will be different types of servers. The deployment of the same application is divided into different environments [11]. So operations and maintenance personnel need to subdivide the servers and applications to facilitate management. The environment and grouping configuration is used for server and application monitoring. After the alarm configuration module is configured with the specified environment and grouping information, the monitored servers and applications can be classified from the environment and grouping, which makes it more convenient for the operation and maintenance personnel to classify the deployment environment according to the targeted monitoring.

Monitoring configuration for monitoring platform some hardware monitoring items for alarm configuration. It includes whether to alarm, alarm category, alarm mode, and corresponding contact person. The monitoring points can be turned off with one click or specified certain alarm points to be turned off without generating alarms. Monitoring points include: network, database, CPU, memory, disk in the server. Alarm definition: It is used to define the alarms of the business burial point, configure the alarm code at the business burial point, and configure the alarm message to be generated when the alarm is triggered, then when the alarm at the business burial point is triggered, the alarm email message will be sent according to the configured alarm template.

E. Web-side Display Monitoring Information Module

As a platform to provide users with monitoring and alarm data visualization, the web terminal will display all the collected monitoring and alarm data information. Most of the display modules support comprehensive search, view detailed data information, delete data, and clear historical data.

The specific display module includes: the home page of each data summary situation and real-time alarm information.

Server module: It displays the status of the servers connected to the server proxy, allows you to conduct a comprehensive search, view the detailed data information of each server, configure the environment and group information of the server and the server description. The application module is similar to the server module, displaying the status and basic information of the application connected to the SpringBoot proxy.

In addition to the monitoring provided by the agent and the client, there are some network and database monitoring that need to be configured in the monitoring platform interface before they can be monitored and generate alarms when the conditions are met. The database module supports inputting database url and user and password to monitor the number of database connections and capacity. The network module is similar to the database module, just enter the ip that needs to be monitored.

F. Logging Module

The logging module includes recording and displaying exception logs and operation logs. The operation log is used to record the specific operations of all users on the monitoring platform. Each operation log will record the operation user, call interface information, relevant parameter information, operation time, etc. The exception log records the abnormal information generated by the operation and monitoring project itself and generates alarms.

The operation log is implemented by means of annotations, which are configured to each method in the control layer. The annotations are used to automatically generate an operation data when an object invokes the annotated method, which is then stored in the log table.

The implementation logic of the custom annotation is: define the log data to be recorded as operation type, operation module, and operation description respectively. The implementation of custom annotations requires the configuration of three annotations: @Target annotation of type METHOD, @Retention annotation of type RUNTIME, and @Documented. then set to write the operation logging tangent, the scope of the tangent is the web control layer, from the tangent weave in point through the reflection mechanism to obtain the method at the weave in point. Then get the requested class name and method name, and then get the specific information carried by the annotated operation and convert it to parameters, store it in the log object, and establish a database connection for data storage.

The specific implementation code of the operation log section is as follows.

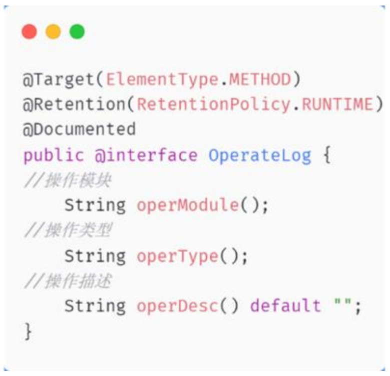
```
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface OperateLog {
//操作模块
    String operModule();
//操作类型
    String operType();
//操作描述
    String operDesc() default "";
}
```

Fig. 1. Log customization annotation

The specific operation information generated is: Operation Module: User Management Module # User, Operation Type: Add, Operation Description: Add User

Examples of use are as follows.

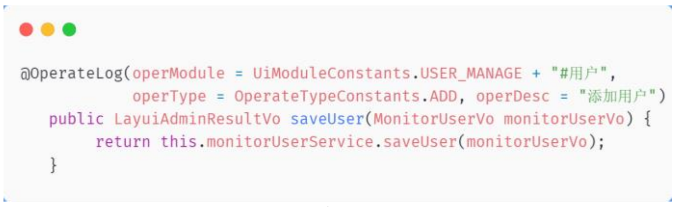
```
@OperateLog(operModule = UiModuleConstants.USER_MANAGE + "#用户",
            operType = OperateTypeConstants.ADD, operDesc = "添加用户")
    public LayuiAdminResultVo saveUser(MonitorUserVo monitorUserVo) {
        return this.monitorUserService.saveUser(monitorUserVo);
    }
```

Fig. 2. Example of log annotation usage

The exception log is recorded by capturing global exceptions on the web side to get the exception information, and then the same as the operation log, set up the operation log cut-scene, the cut-scene range is the web servicer layer, the captured information is obtained at the cut-scene weave entry point, and the exception information is packaged to generate alarms and sent to the server side while stored in the database.

V. System Testing

Software testing is required to ensure the usability of an application system before it goes live. Software testing can ensure the degree of realization and usability of system functions, and fix or risk avoidance for the vulnerabilities found in the system by testing. Moreover, effective software testing can make the various developmental functions of the software itself to be effectively improved, so that the user experience can be enhanced.

Server agent monitoring function test: Test whether the server agent can successfully detect the server-related information in the monitoring platform after it is deployed on the server to be monitored.

First, deploy the server proxy on Tencent cloud server, check the web side, found that this server has been successfully monitored and get detailed server information SpringBoot proxy side monitoring alert function test: build a SpringBoot project, implement the interface with business buried function, test whether the web side can monitor and display this project, and trigger the interface, test whether can be correctly alerted and send email to the specified user. Build SpringBoot project, design business buried function interface /test configure monitoring server IP address and other related information to run the project, after triggering the /test interface to view the web side, found that the project was successfully detected and successfully sent an alert email.

VI. Summary

The whole system so far is actually only considered to build a general framework for operations and monitoring, will be some of the more basic monitoring content design and development completed. In the development process, we encountered many problems and difficulties, the first is the selection of technology. The first is the selection of the technology, because of personal preference for back-end development, so the front-end design can only choose Layui, and how to get server data monitoring tools, through multiple comparisons and learning costs after the calculation of the selection of the older sigar. second is the communication between multiple services and alarm processing, after taking into account the concurrency, after trying to change the alarm processing to asynchronous form. The initial version of the operation and maintenance monitoring system has been developed, but there are still many shortcomings, such as in the customization of alarms in terms of independent configuration, the definition of alarms is still lacking. We hope that we can continue to improve and optimize this system in the future, so that we can really provide a well-functioning monitoring system for enterprise operation and maintenance personnel.

REFERENCES

[1] Mou Hanjie. China Internet Development Report (2021)" released [N]. Farmer's Daily, 2021-07-27(2).
[2] Kong, Xiangwen, Shen, Chen-Nan, Zhang, Chong-Yang, Ren, Shi-Hui. Research on enterprise-level server monitoring and data-based operation and maintenance system[J]. Digital Technology and Applications,2021,39(06):141-143.DOI:10.19695/j.cnki.cn12-1369.2021.06.45.
[3] Wang Yuhui. The design and implementation approach of software operation and maintenance monitoring system[J]. Electronic Components and Information Technology,2020,4(11):165-166.DOI:10.19772/j.cnki.2096-4455.2020.11.078.
[4] Bi Yao. Research on railroad ticket monitoring system based on Zabbix[J]. Railroad communication signal engineering technology,2021,18(12):62-66.
[5] Tamburri, Damian A.; Miglierina, Marco; Nitto, Elisabetta Di (2020). Cloud applications monitoring: an industrial study. Information and Software Technology, 127(), 106376-. doi:10.1016/j.infsof.2020.106376.
[6] Tian Wenhong,Zhao Yong. Optimal Scheduling of Data Center Resources:Theory and Practice [J]. Computer Security,2014(03):56.
[7] Yu Jian,Bao Qi. Design of population information system based on ThinkPHP[J]. Information Technology and Informatization,2022,(02):9-12.
[8] Zhu Rong,Zheng Jianhua. SpringBoot-based waste classification science and curriculum platform[J]. Computer Knowledge and Technology,2022,18(09):22-24.
[9] Kong, Xiangwen, Shen, Chen-Nan, Zhang, Chong-Yang, Ren, Shi-Hui. Research on enterprise-level server monitoring and data-based operation and maintenance system[J]. Digital Technology and Applications,2021,39(06):141-143.DOI:10.19695/j.cnki.cn12-1369.2021.06.45.
[10] Fatema, Kaniz; Emeakaroha, Vincent C.; Healy, Philip D.; Morrison, John P.; Lynn, Theo (2014). A survey of Cloud monitoring tools: taxonomy, capabilities and objectives. journal of Parallel and Distributed Computing, 74(10), 2918- 2933. doi:10.1016/j.jpdc.2014.06.007