Model Construction and Data Management of Running Log in Supporting SaaS Software Performance Analysis

Rui Wang[1,2], Shi Ying[1,2], Chengai Sun[3], Hongyan Wan[1,2], Huolin Zhang[1,2], Xiangyang Jia[1,2]
1. State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China;
2. School of Computer, Wuhan University, Wuhan 430072, China;
3. College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China)
{wangrui1989,*yingshi}@whu.edu.cn, sun910213@163.com, {why0511,huolinzhang,jxy}@whu.edu.cn

Abstract—**Changes in operating environment may result in the performance degradation to a SaaS software. Analyzing running log is an efficient method to locate this problem. However, as a long-running software, SaaS may generate huge log data which is difficult to analyze, and it lacks a systematic approach to implement management of the running log. These all threaten the timeliness of SaaS performance analysis. In this paper, we define a log format to standardize multi-source heterogeneous log data and construct a log model to support SaaS software performance analysis, where the two performance metrics of average response time and request timeout rate in the model are calculated by statistical measurement. Furthermore, a log management framework is given to support real-time big log data collection, access, calculation, storage and service, and the technology implementation of the framework is also given. Finally, a case study is given to illustrate and validate the effectiveness of the approach.**

Keywords-big data; running log; log model; SaaS software performance

I. INTRODUCTION

SaaS software provides software as service to consumers, while service quality is the key factor for consumer satisfaction. Performance as an important QoS attribute of SaaS software, directly affects the user experience. In the dynamic, scalable running environment provided by cloud computing, if the average time that SaaS software responds to service request, especially from tenant is too long, when the service does not meet the service level agreement, and even losses availability, we believe that it is suffering a performance issue (PI). When this happens, it tends to cause consumer dissatisfaction, even cause the loss of consumers, and finally damage the interests of service providers.

After SaaS software is deployed and put into operation, it will provide the highest possible service throughput with the lowest possible response latency in various running environments and usage scenarios. Due to the development quality of the software system, the complex changes of the running environment and its resources, the diversity of usage scenarios and loads, etc., it will cause the software to suffer a variety of performance issues in the runtime. In addition, high performance is the basic requirement of SaaS software. With the increasingly large number of users, business process execution time will get longer and longer, and the possibility of SaaS software performance issues will also grow.

Therefore, at the runtime of SaaS software, operation and maintenance managers need find performance issues timely, exhaustively, accurately, and take measures to ensure that the system can restore the invalid service to the available state in time, make the software continue to provide high performance services. In this case, through corresponding methods and facilities getting a general idea of the performance state of a system becomes a necessary condition to achieve above demand.

The running log of the SaaS software is the data that records the information about the status, events, processes or changes in the software and its operating environment, the usage behavior of users, the occurred events, the interactive messages, and other aspects. Running log is widely applied in the various tasks managed by software system, such as software failure analysis, running environment analysis, usage behavior analysis, etc. The operational information extracted from the running log can help operation and maintenance managers to analyze software status, there are two main means: One is to detect the anomaly of the software by analyzing the log features, for example, Wei Xu[1] proposed an automatic method that extracts information from console logs to detect and visualize runtime problems in large scale distributed systems. Deqing Zou et al.[2] proposed a method that classifies log according to different failure types, which can assists system administrators to monitor the operation of the system and diagnose the failure. Fu X et al.[3] presented a methodology and a system, named LogMaster, for mining correlations of events that have multiple attributions based on system logs and predicting system failure. The other is to analyze the performance of the software by analyzing the performance counters, for example, Keith A. Bare et al.[4] proposed an online diagnostic framework ASDF that transparently monitors and analyzes different time-varying data sources (e.g., OS performance counters, Hadoop logs), and narrows down performance issues to a specific node or a set of nodes. Zhiling Lan et al.[5] proposed a set of techniques to automatically analyze collected data from per node, to detect the nodes acting differently from others in the large scale distributed system, in another reference[6], they adopted a divide-and-conquer approach to address the scalability challenge emerging in this problem and explored the use of non-parametric clustering and two-phase majority voting to improve detection flexibility and accuracy.

However, the above studies mainly focus on the analysis of the characteristics of the log itself to detect software anomalies or monitoring the data in the PaaS or IaaS layer to analyze the

*Corresponding Author
DOI reference number: 10.18293/SEKE2017-128

software performance, few studies have been done on the performance monitoring and analysis of Web application components even fine-grained services in the SaaS layer. The features of SaaS software in the cloud computing environment, lead to large and complex log data related to performance which have high requirement for analysis and processing. So it is difficult to identify and diagnose the performance issues of SaaS software and has high timeliness requirements. Although the application level of big data technology continues to improve, the research on the analysis and processing technology of big log data is not yet mature, and the research on the analysis of SaaS software performance issues based on running log is especially lacking. Therefore, how to scientifically and effectively analyze and utilize big log data, correctly obtain performance-related log data, further diagnose occurred performance issues, and how to use big data technology efficiently record, process and analyze big log data become a challenging problem in log data analysis and processing.

For these above problems, in view of the SaaS software and its cloud computing environment, combined with the demand for SaaS software performance analysis, this paper studies how to construct log model and manage the log data supporting SaaS software performance analysis. We first define the log format to normalize the log distributed on each virtual machine node, then construct log model and calculate the performance metrics in the model via statistical measurement method, and finally design and implement the whole framework for managing raw log data and log model data.

This paper is organized as follows. In Section 1, we introduce the research background of this paper. In Section 2, we introduce the method of constructing log model. In Section 3, we explain our log management framework and discuss the technology implementation of the framework. Our case study is presented in Sections 4. We conclude our work in Section 5.

II. Log Model Construction Based on Statistical Measurement

A. Performance Metrics

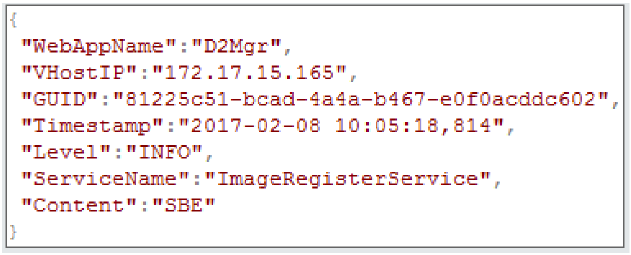
Performance is generally used to define and measure the time constraints and resource allocation degree of a software system efficiency. Common performance metrics include response time, throughput, resource utilization, etc., the first two reflect the performance status of the application level, the latter reflects the performance status of the server level.

In this paper, from the application level, the two metrics of response time and throughput are obtained through collecting and constructing of the performance metric data from SaaS software running log. Performance monitoring of SaaS applications belongs to high-level monitoring[7]. In order to describe the performance of SaaS software at high level, the performance metrics are transformed into the values of statistical measurement. In this paper, according to the definition of two traditional software performance metrics, response time and throughput, we obtain the two performance metrics supporting SaaS software performance analysis: 1) Average Response Time (ART), that reflects the time efficiency that service processes requests. The longer the ART, the slower service processing requests, the lower the performance; 2) Request Timeout Ratio (RTR), which reflects the degree of service overload. The higher the RTR, the higher load, the lower the performance.

B. Log Format

Context information at SaaS software runtime is generally obtained by the way of logging. We define logging rules as follows: 1) Service Invocation Begin (SBE), the SBE event has to be logged as the first instruction (begin) of the service. The event, if logged, provides the trace that the entity initiated the invocation of the service; 2) Service Invocation End (SEN), the SEN event notifies the termination (end) of the service and has to be logged immediately before each normal exit point of the service. The event, once logged, provides the trace that the entity completed the invocation of the service; 3) Service Timeout Error (STE), the STE entry is written in the event log when a service timeout is detected and reported to system.

Log components generally employ binary instrumentation[8] to log, but the log records generated by this method has the characteristic of discontinuity. We solve this problem by adding a unique identifier GUID in the log records. We use the JSON format as the log format standard. An example of log format is shown in Fig. 1.



```
{
 "WebAppName":"D2Mgr",
 "VHostIP":"172.17.15.165",
 "GUID":"81225c51-bcad-4a4a-b467-e0f0acddc602",
 "Timestamp":"2017-02-08 10:05:18,814",
 "Level":"INFO",
 "ServiceName":"ImageRegisterService",
 "Content":"SBE"
}
```

Figure 1. An example of log format.

And we adopt the time window method to analyze the SaaS software running log[9]. The size of the time window can be set according to actual demand.

So GUID and TimeStamp are the core fields of the log model supporting SaaS software performance analysis. The work of this paper focuses on analyzing the performance of SaaS software, we need monitor the performance of each service component even fine-grained service method in cloud environment, so the SaaS software running log should also include the location information field about SaaS service component even service method: Web component name and IP address of virtual machine. The basic log information field should include service name (service method name), log level (log type) and log context (context execution information).

C. Log Model

The raw log data generated by SaaS software at runtime has the characteristics of volume, velocity, value and veracity (4V). For big data processing technology, there are many applications in other fields[10-11]. While the log is a stream of data that is a real-time, continuous, time-varying sequence of data items. Data stream model is the logical abstract expression of the data stream, which can improve the processing efficiency of data streams. In order to improve the efficiency of big log data processing and analysis, we propose a log model, it mainly consists of 7 attributes, as shown in Fig. 2.

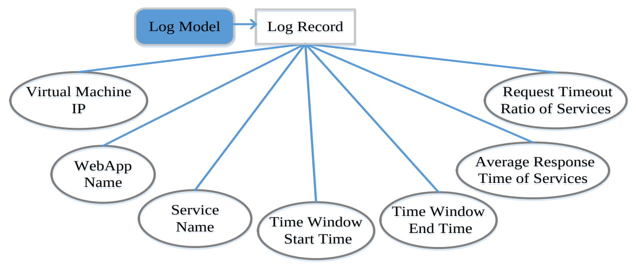


Figure 2. Log model supporting SaaS software performance analysis.

We calculate the two performance metrics of Average Response Time and Request Timeout Ratio in the model by statistical measurement method.

1) *Average Response Time Calculation*

Response time is the end-to-end time that a task spends to traverse a certain path within the system[12]. Average Response Time reflects the user expectation on time that software responds to request. The ART is computed as

$$ART_w = E[T_{rw}] = \sum_{i=1}^{n}(t_{ri} - t_{si})/n \qquad (1)$$

where W is the time window, it can be a given time interval, such as 1 minutes, 1 hours, 1 days etc.; T_{rw} is the response time of request r in W; n is the number of requests in W; t_{ri} indicates the arrival time of the request i, that is the time of SBE; t_{si} indicates the service response time of the request i, that is the time of SEN. This calculation does not take into account the time consumption about network transmission from client to server. Equation (1) is applied only in the case that each (SBE, SEN) pair in a time window.

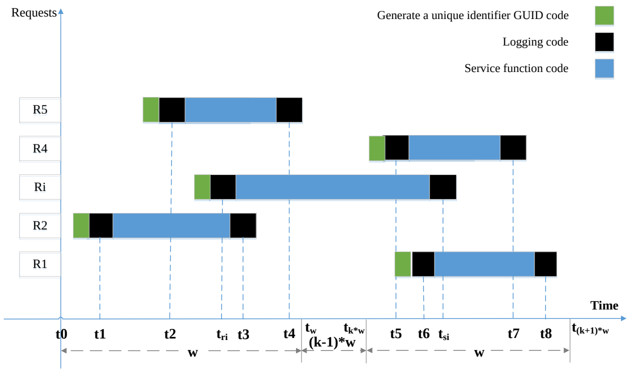


Figure 3. The requests of across the time windows.

For example in Fig. 3, request Ri across multiple time windows from request arrival to service response. We consider the proportion of time spent on each time window. We divide the Ri into three parts, calculate the ART for each part. The ART of request Ri in the time window $[t_0, t_w]$ is $\frac{t_w - t_{ri}}{(t_w - t_{ri})/(t_{si} - t_{ri})}$, in the time window $[t_{kw}, t_{(k+1)w}]$ is $\frac{t_{si} - t_{kw}}{(t_{si} - t_{kw})/(t_{si} - t_{ri})}$, in the time interval $[t_w, t_{kw}]$ is $\frac{(\mathrm{k}-1)\mathrm{w}}{(\mathrm{k}-1)\mathrm{w}/(t_{si} - t_{ri})}$.Therefore, the ART for all requests in the time window$[t_0, t_w]$ is $\frac{(t_3 - t_1)+(t_4 - t_2)+(t_w - t_{ri})}{1+1+(t_w - t_{ri})/(t_{si} - t_{ri})}$, in the time window $[t_{kw}, t_{(k+1)w}]$ is $\frac{(t_7 - t_5)+(t_8 - t_6)+(t_{si} - t_{kw})}{1+1+(t_{si} - t_{kw})/(t_{si} - t_{ri})}$.

Combined with the above example, we modify (1) as

$$ART_{w_j} = \frac{\sum_{i=1}^{n_j} T_{r_i W_j}}{\sum_{i=1}^{n_j} K_{r_i W_j}} \qquad (2)$$

where W_j is a time window from t_{jw} to $t_{(j+1)w}$, i.e. $W_j = [t_{jw}, t_{(j+1)w}]$; n_j is the number of requests within W_j; $T_{r_i W_j}$ is the time proportion of ith request r_i in W_j, i.e.

$$T_{r_i W_j} = \begin{cases} t_{si} - t_{ri}, & t_{jw} < t_{ri} < t_{si} < t_{(j+1)w} \\ t_{si} - t_{jw}, & t_{ri} < t_{jw} < t_{si} < t_{(j+1)w}; \\ t_{(j+1)w} - t_{ri}, & t_{jw} < t_{ri} < t_{(j+1)w} < t_{si} \end{cases}$$

$K_{r_i W_j}$ is the number of ith request r_i in W_j, i.e.

$$K_{r_i W_j} = \begin{cases} 1, & t_{jw} < t_{ri} < t_{si} < t_{(j+1)w} \\ \frac{t_{si} - t_{jw}}{t_{si} - t_{ri}}, & t_{ri} < t_{jw} < t_{si} < t_{(j+1)w} \\ \frac{t_{(j+1)w} - t_{ri}}{t_{si} - t_{ri}}, & t_{jw} < t_{ri} < t_{(j+1)w} < t_{si} \end{cases}.$$

2) *Request Timeout Ratio Calculation*

Request Timeout Ratio describes that the service accepts multiple requests simultaneously, but responses to some requests exceed the user expectation in a given time window, then the percentage of these requests is the request timeout ratio.

In time window W_j, the RTR is the ratio of the number of timeout requests N_{tW_j} to the total number of requests N_{aW_j}:

$$RTR_{W_j}(\Delta t) = \frac{N_{tW_j}}{N_{aW_j}} \qquad (3)$$

Because it is a ratio, the metric is less influenced by isolated extreme values, which makes it more independent of time window length. But N_{tW_j} and N_{aW_j} can not be directly obtained through the program, we modify (3) as

$$RTR_{W_j}(\Delta t) = \frac{N_{tW_j}}{N_{aW_j}} = \frac{\sum_{i=1}^{n_j} N_{tW_j}^{r_i}}{\sum_{i=1}^{n_j} N_{aW_j}^{r_i}}, \sum_{i=1}^{n_j} N_{aW_j}^{r_i} \neq 0 \quad (4)$$

where $N_{tW_j}^{r_i}$ indicates whether ith request r_i timeouts within W_j,

$$N_{tW_j}^{r_i} = \begin{cases} 1 & t_{jw} < t_{ri} < t_{si} < t_{(j+1)w}, t_{si} - t_{ri} > \Delta t \\ 0 & \mathit{otherwise} \end{cases},$$

1 indicates that the request timeouts within the window.

$N_{aW_j}^{r_i}$ indicates whether ith request r_i is within W_j,

$$N_{aW_j}^{r_i} = \begin{cases} 1 & t_{jw} < t_{ri} < t_{si} < t_{(j+1)w} \\ 0 & \mathit{otherwise} \end{cases},$$

1 indicates that the request is within the window.

A request timeout belongs to the STE event, we estimate the expected duration Δt of the requested service timeout by EWMA[13]. The timestamps of the SBE and SEN events, logged at each exception-free invocation, are used to profile Δt. The EWMA statistic as described in the following:

$$\Delta_n = (1-\alpha)\Delta_{n-1} + \alpha\Delta_l \qquad (5)$$

where Δ_l is the last duration estimate (obtained by subtracting the timestamp of the SBE from the one of the SEN event at the last service invocation); α is the weight, the weight is small for taking into account the history in the changing tendency of the Δ_n series. A requested service timeout is detected if the SEN event is not observed within $\Delta t = n_s \cdot \Delta_n$ time units since the SBE. Then the STE event will be written in the log and reported to system.

The structure of log model is fixed as shown in Fig. 2, but log model data is calculated in real time with the generation of raw log data, it has the characteristic of rapid growth. In addition, the log model is for different roles, and the requirements for the log model are different due to different demands of the roles. In Fig. 2, we define the core attributes of a log model that supports the diagnosis of performance issue that response time is too long. In the actual diagnosis, we can add the specific attribute items to meet the different requirements of specific diagnostic tasks of different roles.

III. Log Management Supporting SaaS Software Performance Analysis

A. Log Management Framework

The log data management framework adopts the idea of modularization[14].

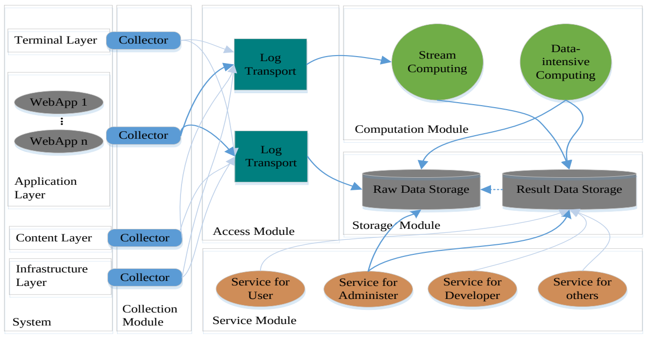


Figure 4. The log management framework supporting SaaS software performance analysis.

Fig. 4 shows the log management framework that supports the processing of big log event stream data in real-time. It consists of five modules: collection module, access module, computation module, storage module and service module, the functions of each module are described as follows:

1) Log Collection Module

This module is responsible for collecting data in real-time from virtual machine nodes on the cloud platform. As the log data is multi-sourced and heterogeneous, each collector logs the data generated by different objects and normalizes the collected log data according to the log format defined in this paper;

2) Log Access Module

This module is responsible for easing the situation that the speed of data collection and data processing is not synchronized, and solving the cost problem that immediately process collected data. So it is necessary to transfer the log data to the storage module or computation module for centralized processing;

3) Log Computation Module

In this module, one is data-intensive computing applied to analyze massive raw log data to obtain the required information. The other is stream computing applied to process the coming data in real-time. Furthermore, this module is extensive for its computing frameworks based on actual analysis demands;

4) Log Storage Module

This module includes two kinds of storage systems. One is raw data storage, which stores historical data orderly and permanently for future data mining and analyzing. The other is result data storage, which provides rapid access to data for different role analysis, evaluation or prediction;

5) Log Service Module

This module consists of services that can meet the different data demands of different roles. Each service reads required data from the Result Data Storage for all roles in each layer. In addition, to meet the demand of an administrator for diagnosing performance issues, the raw log data in Raw Data Storage are needed.

B. Implementation of Log Management Framework

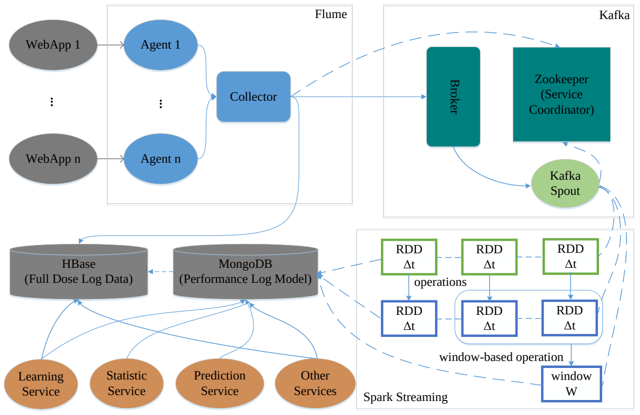


Figure 5. The log management framework implementation supporting SaaS software performance analysis.

As shown in Fig. 5, event log generated from the SaaS application WebApp deployed on the virtual machine in cloud platform will be collected by Agent of the Flume collection process of each virtual machine, then it flows into the Flume sink node of a virtual machine and generates two log event stream branch: The first flows directly to the HBase database for storage; The second will flow into the Spark Streaming for calculation, and the results are written into the MongoDB database.

1) Log Collection Module Implementation

We collect the log context of SaaS software at runtime by interceptor technology, as shown in Fig. 6.

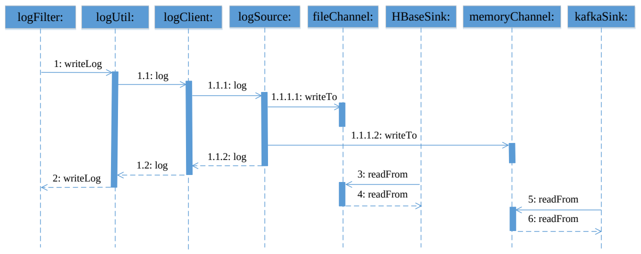


Figure 6. Log collection sequence diagram.

2) *Log Access Module Implementation*

We adopts Kafka mainstream framework as distributed message queue middleware, as shown in Fig. 7.

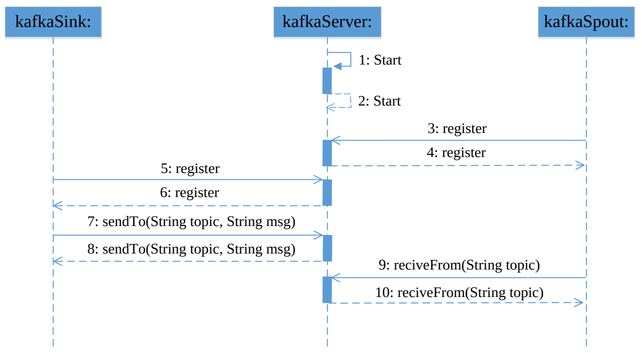

kafkaSink:
kafkaServer:
kafkaSpout:
1: Start
2: Start
3: register
4: register
5: register
6: register
7: sendTo(String topic, String msg)
8: sendTo(String topic, String msg)
9: reciveFrom(String topic)
10: reciveFrom(String topic)

Figure 7. Kafka's producers and consumers publish subscribe message sequence diagram.

3) *Log Computation Module Implementation*

We implement the real-time calculation of log data stream based on Spark Streaming[15], as shown in Fig. 8.

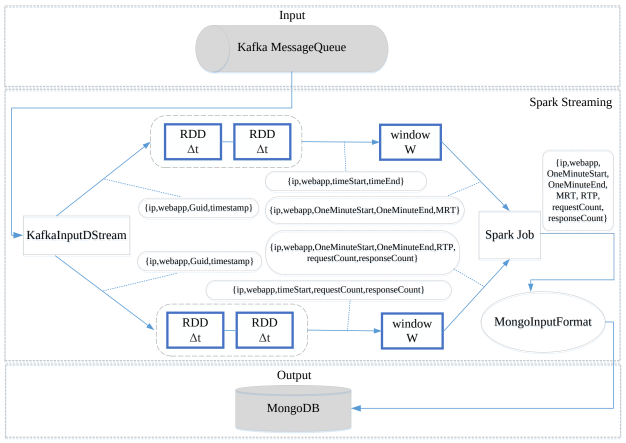

Input
Kafka MessageQueue
Spark Streaming
RDD Δt
RDD Δt
window W
{ip,webapp,timeStart,timeEnd}
{ip,webapp,Guid,timestamp}
{ip,webapp,OneMinuteStart,OneMinuteEnd,MRT}
{ip,webapp, OneMinuteStart, OneMinuteEnd, MRT, RTP, requestCount, responseCount}
KafkaInputDStream
Spark Job
{ip,webapp,OneMinuteStart,OneMinuteEnd,RTP, requestCount,responseCount}
{ip,webapp,Guid,timestamp}
{ip,webapp,timeStart,requestCount,responseCount}
RDD Δt
RDD Δt
window W
MongoInputFormat
Output
MongoDB

Figure 8. Log data stream real-time computing process.

Spark Streaming provides some basic log data statistics interface, we use its basic statistical functions: total, mean and percentage. For the window operation of Spark Streaming, we use incremental methods to improve the efficiency of statistics.

4) *Log Storage Module Implementation*

a) log data storage

Raw log data generally has no high-frequency access and has no high requirement for real-time, so we consider storing it on Hadoop Distributed File System (HDFS). This paper stores the raw log data according to the log level in the log record;

b) log model data storage

Log model data will be accessed frequently by the log service module. Combined with the characteristics of the log model itself, this paper chooses to store it in the extensible, high-performance distributed NoSQL database MongoDB. In order to improve the efficiency of query processing, we establish B-tree index[16] at the SBE time for log model data with the increasing computational requirements.

5) *Log Service Module Implementation*

The module provides the basis for operation and maintenance managers to find software performance issues, and improve the issues effectively. For the implementation of this module, from discovering performance issues to timely improving performance issues, demand of the roles in SaaS software may involve three services: historical log data statistics service, real-time monitor service and prediction service.

IV. Case Study

We evaluate the effectiveness of our method by an Integrated Disaster Reduction Application System (IDRAS). It relies on the basic idea of cloud service platform and service oriented architecture (SOA), has a series of Web components with independent functions, as shown in Fig. 9.

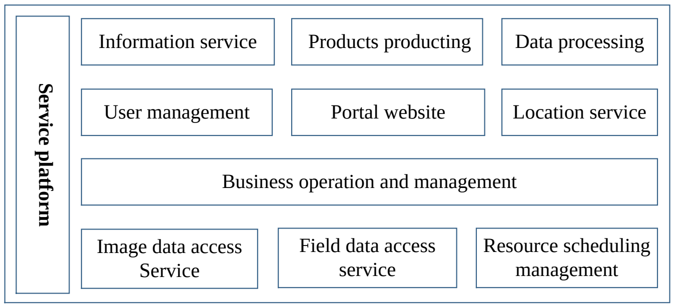


Figure 9. The service platform of IDRAS.

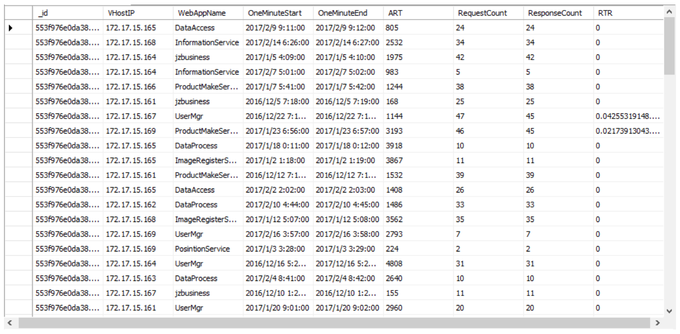
When the disaster occurs, all the SaaS software services in the system work together to provide services. In such a complex production environment, SaaS application will produce hundreds of MBs of log data with millions of events in them. Full manual analysis of these log data is not a realistic option, yet they need to be used daily by the operation and maintenance managers to monitor and resolve performance issues. Therefore, it is of great significance to provide a unified log management system for the massive running log generated by IDRAS to support its performance analysis. And our log management system can handle these hundreds of MBs of log data even larger.

We create a collection called LogModel in the MongoDB, the document structures of the collection include Key, Value, and Type. Table I elaborates the types and descriptions of each key in the log model in the document structure.

TABLE I. LOG MODEL DOCUMENT STRUCTURE DESCRIPTION

Key	Type	Description
_id	ObjectId	Primary key field
VHostIP	String	Mobile IP or fixed IP
WebAppName	String	such as war files
OneMinuteStart	DateTime	Start time
OneMinuteEnd	DateTime	End time
ART	Int32	Average Response Time
RequestCount	Int32	The number of requests in 1min
ResponseCount	Int32	The number of responses in 1min
RTR	Double	1-(ResponseCount/RequestCount)

We visualize the log model data results supporting IDRAS performance analysis through MongoVUE, Fig. 10 shows a small part of the data.



	_id	VHostIP	WebAppName	OneMinuteStart	OneMinuteEnd	ART	RequestCount	ResponseCount	RTR
▶	553f976e0da38....	172.17.15.165	DataAccess	2017/2/9 9:11:00	2017/2/9 9:12:00	805	24	24	0
	553f976e0da38....	172.17.15.168	InformationService	2017/2/14 6:26:00	2017/2/14 6:27:00	2532	34	34	0
	553f976e0da38....	172.17.15.164	jzbusiness	2017/1/5 4:09:00	2017/1/5 4:10:00	1975	42	42	0
	553f976e0da38....	172.17.15.164	InformationService	2017/2/7 5:01:00	2017/2/7 5:02:00	983	5	5	0
	553f976e0da38....	172.17.15.166	ProductMakeSer...	2017/1/7 5:41:00	2017/1/7 5:42:00	1244	38	38	0
	553f976e0da38....	172.17.15.161	jzbusiness	2016/12/5 7:18:00	2016/12/5 7:19:00	168	25	25	0
	553f976e0da38....	172.17.15.167	UserMgr	2016/12/22 7:1...	2016/12/22 7:1...	1144	47	45	0.04255319148....
	553f976e0da38....	172.17.15.169	ProductMakeSer...	2017/1/23 6:56:00	2017/1/23 6:57:00	3193	46	45	0.02173913043....
	553f976e0da38....	172.17.15.165	DataProcess	2017/1/18 0:11:00	2017/1/18 0:12:00	3918	10	10	0
	553f976e0da38....	172.17.15.165	ImageRegisterS...	2017/1/2 1:18:00	2017/1/2 1:19:00	3867	11	11	0
	553f976e0da38....	172.17.15.161	ProductMakeSer...	2016/12/12 7:1...	2016/12/12 7:1...	1532	39	39	0
	553f976e0da38....	172.17.15.165	DataAccess	2017/2/2 2:02:00	2017/2/2 2:03:00	1408	26	26	0
	553f976e0da38....	172.17.15.162	DataProcess	2017/2/10 4:44:00	2017/2/10 4:45:00	1486	33	33	0
	553f976e0da38....	172.17.15.168	ImageRegisterS...	2017/1/12 5:07:00	2017/1/12 5:08:00	3562	35	35	0
	553f976e0da38....	172.17.15.169	UserMgr	2017/2/16 3:57:00	2017/2/16 3:58:00	2793	7	7	0
	553f976e0da38....	172.17.15.169	PosintionService	2017/1/3 3:28:00	2017/1/3 3:29:00	224	2	2	0
	553f976e0da38....	172.17.15.164	UserMgr	2016/12/16 5:2...	2017/12/16 5:2...	4808	31	31	0
	553f976e0da38....	172.17.15.163	DataProcess	2017/2/4 8:41:00	2017/2/4 8:42:00	2640	10	10	0
	553f976e0da38....	172.17.15.167	jzbusiness	2016/12/10 1:2...	2016/12/10 1:2...	155	11	11	0
	553f976e0da38....	172.17.15.161	UserMgr	2017/1/20 9:01:00	2017/1/20 9:02:00	2960	20	20	0

Figure 10. Log model result data of the service platform of IDRAS.

As can be seen from the data in Fig. 10, SaaS components of UserMgr and ProductMakeService receive 47 and 46 user requests respectively in a certain minute, but they only respond to 45 requests, and their average response time are fluctuating abnormally or fluctuating up over a time period in which the request timeouts, this indicates that the two SaaS components may be experiencing a PI. About what causes this result, the expert can combine the raw log data in the HBase database to analyze whether an exception or error has occurred in a time window in which the request timeouts. The expert can locate the reasons of SaaS software PI through tracking analysis of abnormal or error log.

The analysis results were evaluated by performance experts from IDRAS who has 8 years of experience in SaaS software performance analysis and deep knowledge of the IDRAS infrastructure. In this evaluation the experts focus on evaluating whether the result data generated by our method reflects the software is experiencing PI, the evaluation results as shown in Table II and Table III.

TABLE II. PI EVALUATION 1

PI ID: 1 Date: 2016-12-22 07:15:56 Strategy used: tracing
Manual diagnosis: Page reads/sec high on UserMgr. Cause: server restarted → cache empty so needs to be filled up.
Verification: Is real PI: Yes Diagnosis correctness: 1

TABLE III. PI EVALUATION 2

PI ID: 2 Date: 2017-01-23 06:57:56 Strategy used: tracing
Manual diagnosis: Images download timeout (9000 ms) has expired. Cause: cache I/O busy → read and write exception so needs to optimize the cache mechanism.
Verification: Is real PI: Yes Diagnosis correctness: 1

From the above two evaluation results can be seen, the results data generated by our method reflect the software is experiencing PI, it is consistent with the expert's diagnostic results. Software running cycle is long, we just choose some real-world performance problems that prove the effectiveness of our method, in the next running time, as long as the performance issues occur, we can find the performance experts to confirm, and so far our all results are corroborated with the experts.

V. CONCLUSION

From the demand and research status of SaaS software performance monitoring and analysis in cloud computing environment, this paper proposes a format standard that standardizes the SaaS software running log. Based on the standardized log, we calculate the performance metrics of SaaS software components with the aid of the service average response time calculation model and service request timeout ratio calculation model, and then obtain the log model supporting SaaS software performance analysis. In addition, this paper also presents a management framework and technology implementation of the raw log data and log model data, which supports the performance analysis of SaaS software. Finally, we demonstrate the effectiveness of our approach through a case study. The next step will improve the work of the log service module, and combine the low-level indicators to study the automatic identification of SaaS software performance issues.

ACKNOWLEDGMENT

This work is supported in part by the grants of National Key Research and Development Program of China (2016YFC1202204) and National Natural Science Foundation of China (61373038, 61672392).

REFERENCE

[1] Wei Xu. Detecting Large Scale System Problems by Mining Console Logs. PhD dissertation, UC Berkeley, 2010.

[2] Deqing Zou, Hao Qin, Hai Jin. UiLog: Improving Log-Based Fault Diagnosis by Log Analysis. Journal of Computer Science and Technology, 2016, 31(5): 1038-1052.

[3] Fu X, Ren R, Zhan J, et al. LogMaster: mining event correlations in logs of large-scale cluster systems. 2012 IEEE 31st Symposium on Reliable Distributed Systems(SRDS), 2012:71-80.

[4] Keith A. Bare, Soila Kavulya, Jiaqi Tan, et al. ASDF: An Automated, Online Framework for Diagnosing Performance Problems. Workshop on Software Architectures for Dependable Systems(WADS), 2009: 201-226.

[5] Zhiling Lan, Ziming Zheng, Yawei Li. Towards Automated Anomaly Identification in Large-Scale Systems. IEEE Transactions on Parallel and Distributed Systems(TPDS), 2010, 21(2): 174-187.

[6] Li Yu, Zhiling Lan. A Scalable, Non-Parametric Method for Detecting Performance Anomaly in Large Scale Computing. IEEE Transactions on Parallel and Distributed Systems(TPDS), 2016,27(7): 1902-1914.

[7] Aceto G, Botta A, De Donato W, et al. Cloud monitoring: A survey. Computer Networks. 2013, 57(9): 2093-2115.

[8] Mona Attariyan, Michael Chow, Jason Flinn. X-ray: Automating Root-Cause Diagnosis of Performance Anomalies in Production Software. 10th USENIX Symposium on Operating Systems Design and Implementation(OSDI), 2012: 307-320.

[9] Zhu Baojin. Design and Implementation of Log Filter System for Cloud Computing System. Hangzhou: Hangzhou Dianzi University, 2014.

[10] Xu Z, Liu Y, Mei L, et al. Semantic based representing and organizing surveillance big data using video structural description technology. Journal of Systems and Software, 2015, 102: 217-225.

[11] Liu J, Yu X, Xu Z, et al. A cloud - based taxi trace mining framework for smart city. Software: Practice and Experience, 2016.

[12] Cortellessa V, Di Marco A, Inverardi P. Model-based software performance analysis. Springer Science & Business Media, 2011. 4:10.

[13] Tao Wang, Wenbo Zhang, Jun Wei et al. Fault detection for cloud computing systems with correlation analysis. IFIP/IEEE International Symposium on Integrated Network Management(IM), 2015: 652-658.

[14] Meiyappan Nagappan. A Framework for Analyzing Software Systems Log Files. PhD thesis, NC State Uni., 2011.

[15] Holden Karau, Andy Konwinski, Patrick Wendell et al. Learning Spark: Lightning-fast data analysis[M]. CA: O'Reilly Media,Inc., 2015.

[16] Wang Zhaoyong. Research of Storage Methods for Large-scale bulk log data. Chengdu: University of Electronic Science and Technology of China, 2011