基于日志分析平台的监控系统的设计与实现

王力群 黄必栋
（南京铁道职业技术学院软件技术系 江苏 南京 210031）

摘 要 介绍基于日志分析平台的监控系统的设计与实现。针对复杂软件系统的监控点类型多样、监控点数量多、监控点易变化、监控数据量大等问题，提出一种基于日志分析平台 ELK 的监控系统的设计与实现方法。通过对日志分析平台 ELK 进行改造，把日志处理中的收集、存储、索引、搜索、分析方法引入到监控系统的设计与实现中，解决了传统监控方法存在的问题，为监控系统的设计提出了新的思路。

关键词 监控系统 日志分析 ELK Elasticsearch

中图分类号 TP3 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2017.12.030

DESIGN AND IMPLEMENTATION OF MONITOR SYSTEM BASED ON LOG ANALYSIS PLATFORM

Wang Liqun Huang Bidong
(*College of Software and Art Design, Nanjing Institute of Railway Technology, Jiangsu 210031, Nanjing, China*)

Abstract This article introduces the design and implementation of monitoring system based on log analysis platform. There are many problems in complex software systems, such as the multi-type monitor point, the excessive monitoring points, the easy change of monitoring points and the large amount of monitoring data. Therefore, we propose a design and implementation method of monitoring system based on log analysis ELK platform. By reforming the log analysis ELK platform, the method of collection, storage, indexing, search and analysis in log processing is introduced into the design and implementation of monitoring system, which solves the problem existing in the traditional monitoring method. It presents a new idea for the design of the monitor system.

Keywords Monitor system Log analysis ELK Elasticsearch

0 引 言

稳定、可靠、高性能的软件系统是以用户大量使用及反馈为基础，不断满足新需求，不断完善和改进系统功能与性能，不断演化系统架构与技术而形成。如互联网系统，大量用户的高并发访问和使用，能及时反馈并发现问题，从而使系统不断地进行调整、完善以满足新的改进需求。而监控系统在发现问题、提高系统稳定性、分析系统性能等方面发挥着巨大的作用[1-2]。

监控系统能实时收集系统运行指标，通过配置好的监控规则及时发现系统异常，并通过有效途径进行报警和通知。系统运行指标可分为系统底层运行指标和业务运行数据指标，通用的监控系统软件一般监控系统底层运行指标，如开源监控系统 Zabbix 可以监控系统底层异常，包括进程异常、IO 异常、内存异常、网络异常等[3]。而由于业务系统架构的多样化、不同业务系统间的差异大等特征，业务运行指标的监控没有通用的软件系统可以使用，通常业务监控系统需要进行定制化的搭建。业务运行指标数据一般通过 BI 系统、业务日志保存，通过 BI 系统可以分析历史业务数据，但不具备实时性；而业务日志的分析则通过日志分析软件，分析缓慢且依赖于开发人员的经验，发现问题比较滞后。能否把业务指标数据统一收集、存储，并可以方便检索、分析、统计是解决系统监控问题的关键。

本文介绍了一种基于日志分析平台 ELK 的监控系统设计。日志分析平台 ELK 是近年来迅速发展的

收稿日期：2017-07-26。**王力群**，副教授，主研领域：软件设计与开发技术与面向对象设计模式。**黄必栋**，工程师。

开源日志分析平台,包含了 Elasticsearch 全文检索数据库、Logstash 日志收集器和 Kibana 可视化数据搜索、统计 Web 接口。监控系统以 Logstash 日志收集器为基础扩展了业务数据收集方法,使用 Elasticsearch 存储、索引业务运行数据,基于 Elasticsearch API 构建了监控模块。解决了业务数据的收集难,存储和分析困难的问题,提出了易扩展、可灵活配置和可快速搭建的监控系统框架。

1　架构设计

1.1　体系结构

监控系统的设计以 ELK 日志分析平台为基础,采用了 Kafka 消息队列作为数据传送缓冲器,实现了数据收集 API、监控服务、监控 Web 接口,系统架构如图 1 所示。

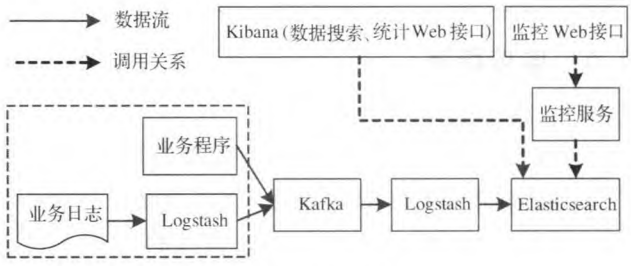

图 1　系统架构

在业务运行服务器中通过两种方法采集数据:(1) 通过 Logstash 收集文件更新数据;(2) 通过数据收集 API 收集业务程序中的数据。收集到的数据统一发送到数据传送缓冲器 Kafka,再通过 Logstash 传送到 Elasticsearch 数据库。Elasticsearch API 是一种基于 HTTP 协议的 RESTful API,其提供了数据搜索、统计等功能,这方便了基于 Elasticsearch 进行二次开发[4]。监控服务根据配置好的监控指标规则,使用 Elasticsearch API 周期性的查询、计算监控指标。用户通过监控 Web 接口对监控任务进行配置,并可以查询历史监控记录,方便了对监控事件的跟踪和查询。ELK 中的 Kibana 提供了灵活、易用的 Web 接口,使用户能搜索、统计收集到的监控数据,并可对监控规则进行验证。

1.2　数据收集

数据收集是指把运行中的程序数据通过某种方法收集起来,这些数据包括了用户访问设定的参数、用户访问接口数据、内部服务接口调用数据、外部接口调用数据、开发人员自定义的业务指标等。其中的接口调用数据还包含请求、返回、调用时间、调用异常信息等。数据收集点示例如图 2 所示,实心圆点代表的是数据收集点。收集点分布在前端(Web 页面和 App)与后台交互处、前端服务接口处(如 PHP 层接口)、后台业务服务调用点、第三方服务调用点等关键点中。这样用户访问行为数据、内部服务交互数据、数据访问交互行为、第三方服务访问数据都可以被收集到。数据收集方法及与业务程序的集成方法是数据收集的关键。

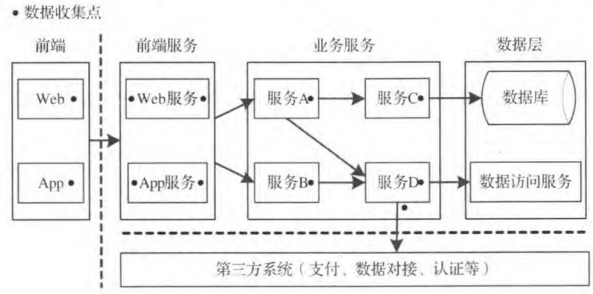

图 2　数据收集

根据收集的数据特性使用文件或 API 的收集方法。如大量用户访问情况下,前端服务接口会产生大量的数据,若使用文件收集的方法会影响到系统的磁盘 IO,则应使用 API 的收集方法。而与第三方服务交互处的数据收集则应使用文件的收集方法,因为第三方交互数据的收集可靠性要求较高,使用文件保存数据稳定、可靠。

文件或 API 的收集方法对开发人员而言是透明的,使用统一的 API 形式。这样方便了开发人员集成数据收集功能到业务模块中,不需要了解数据收集机制、数据格式、数据文件管理等。

1.3　数据格式

开发人员通过 API 的方式集成数据收集模块,不需要关心采集的数据格式,只需要了解 API 参数的含义,而 API 底层则需要处理数据格式。对于文件收集方法,写入文件的是多行 JSON 字符串;对于 API 收集方法,发送至 Kafka 的是 JSON 格式数据。从 Kafka 到 Logstash 再到 Elasticsearch 传送也都是 JSON 格式数据。因此整个数据传送过程中都使用 JSON 格式数据,数据格式清晰、不容易出错。

1.4　数据定义

数据定义决定了数据收集的内容,是能否完成项监控任务的前提。一般使用倒推的方法定义数据收集字段,即根据监控需求分析出需要收集的数据内容。基本收集字段如表 1 所示,使用这些基本字段可以对接口调用情况进行统计以获得监控指标,如根据“event_time”和“time_consumed”字段可以统计出在某个时间段执行某项任务的平均时间。根据“event_time”,“task”,“result”字段可以统计出在某个时间段执行某

项任务的异常次数、异常率。

表 1 数据收集的基本字段

字段	含义
event_time	收集数据的时间点
system	收集数据所在的服务
module	收集数据所在的模块
task	执行的任务
time_consumed	执行某项任务使用的时间
result	执行某项任务的结果
msg	自定义消息
error	异常信息

在实际的应用中基本字段往往还不能完全满足监控需求,这时需要对这些字段进行扩充以达到监控要求。如需要对页面的 PV(访问量)、UV(独立访客)进行统计则需加入表 2 所示字段。

表 2 页面访问相关字段

字段	含义
page	页面
user	用户

收集字段的定义需要统一规划,可以预先定义部分未来可能会使用到的字段,这样能有效地减少收集字段的调整次数。同时字段的调整也不可避免,为了方便后期维护字段名称应尽量确保无二义性,且清晰易懂。

1.5 Elasticsearch 数据库 schema 的设计

收集数据定义好之后便可以设计 Elasticsearch 数据库的 schema,schema 中的字段可以直接使用收集数据定义的字段,需要指定字段的类型。Elasticsearch 是一个基于 Lucene 构建的开源、分布式、RESTful 风格的搜索引擎数据库,已应用到 Github、Stack OverFlow 等大型网站的全文搜索中[2]。搭建的 Elasticsearch 集群稳定可靠、可以容纳较大的数据量,符合存放业务数据的要求。

Elasticsearch schema 中必须指定一个字段为时间字段,以此字段为线索构成时间相关的事件序列,这里选择 event_time 字段作为时间字段。在 Elasticsearch 中字符串字段有两种类型:analyzed 和非 analyzed。analyzed 字段为全文检索字段,索引时会对文本进行分词处理;非 analyzed 字段为非全文检索字段。Analyzed 字段也不可以做聚合操作,若对此类型字段进行聚合即是对文本分词的结果进行聚合,执行效率很低。表 3 给出了各基本字段类型的配置,其中“time_consumed”字段需要进行数值计算,如计算平均值,故设置为 number 类型;“result”、“system”、“module”、“task”和“result”字段为枚举类型字段,故设置为非 analyzed 字段;“msg”和“error”字段中的值变化较大,包含的内容多,故设置为 analyzed 字段。

表 3 基本字段类型配置

字段	类型	是否可聚合	Analyzed
event_time	Date	是	否
system	String	是	否
module	String	是	否
task	String	是	否
time_consumed	Number	否	否
result	String	是	否
msg	String	否	是
error	string	否	是

1.6 监控服务

1.6.1 监控指标

监控数据进入 Elasticsearch 数据库后,应对其进行定期的监控指标计算才能完成监控功能。监控指标的计算是通过对收集数据的统计完成的,如要计算最近 5 分钟执行任务 A 的异常率,则需要统计出最近 5 分钟执行任务 A 的异常次数和总次数,再计算得出异常率;要计算最近 5 分钟执行任务 B 的平均时间,则需计算出“time_consumed”字段在最近 5 分钟的均值。

监控指标的计算可以通过 Elasticsearch 统计 API 完成。统计 API 中的统计功能在 Elasticsearch 中完成,不需要先查询数据再进行统计,即方便了统计又提高了效率。监控指标主要分为三类:

1) 符合条件的次数 如监控指标为最近 5 分钟执行任务 A 的异常次数,这里条件为:(a)最近 5 分钟;(b)执行任务 A;(c)执行异常,则监控指标:

次数 = 符合条件 a 且条件 b 且条件 c 的记录数

2) 符合条件的比例 如监控指标为最近 5 分钟执行任务 B 的异常率,这里条件为:(a)最近 5 分钟;(b)执行任务 A;(c)执行异常,则监控指标:

异常率 = 符合条件 a 且条件 b 且条件 c 的记录数/符合条件 a 且条件 b 的次数

3) 符合条件的字段的平均值 如监控指标为最近 5 分钟执行任务 C 的平均时间,这里条件为:(a)最近 5 分钟;(b)执行任务 C,则监控指标:

平均值 = 符合条件 a 且条件 b 的“time_consumed”字段的均值

这三类的统计都可以通过 Elasticsearch 的聚合计算完成,具体对应关系举例如表 4 所示。监控指标条件的设定较为灵活,结合不同字段的取值可以构成复杂的统计,如:在最近 5 分钟内服务 A 中的模块 B 在执行任务 C 发生异常 D 的异常率。这些复杂的监控指标条件复杂但都能归为上述定义的三类监控指标,可以通过 Elasticsearch API 统计得出,这得益于统计 API 的灵活和功能强大。

表 4　监控指标与 Elasticsearch 查询的对应关系

监控指标	Elasticsearch 查询参数			
	Metric	查询条件	时间范围	聚合方法
最近 5 分钟执行任务 A 的异常次数	Count	task:"任务 A"	最近 5 分钟	过滤器聚合 result:"ERROR"
最近 5 分钟执行任务 B 的异常率	Count	task:"任务 B"	最近 5 分钟	过滤器聚合:1. result:"ERROR" 2. *
最近 5 分钟执行任务 C 的平均时间	Average Field:time_consumed	task:"任务 C"	最近 5 分钟	无

1.6.2　监控规则

监控规则是由监控指标构成的条件、执行时间、行动组成的。监控指标一般为数值类型如次数、比例、时间等,监控条件则为指标与阈值的比较,如异常次数大于 100、异常比例大于 10%、平均时间超过 5 秒等。执行时间指定了什么时间执行监控规则,可以是周期性的,如每隔 2 分钟。监控行动是指满足监控条件后所采取的动作,通常有邮件通知、短信通知[5]等。当行动为通知时还包括了通知的内容模板,内容模板定义了如何向用户传送通知信息,如标题、内容及格式等。通知内容可以包含 Kibana 链接,方便用户对通知信息进行进一步的分析。一条典型的监控规则如表 5 所示,使用的是邮件通知,并包含了邮件通知模板。

表 5　监控规则示例

条件	执行时间	行动
最近 5 分钟内执行任务 A 的异常率超过 5%	每隔 3 分钟	邮件通知 alert@company.com 邮件标题:任务 A 执行异常 邮件正文: 在最近 5 分钟内任务 A 的异常率为[rate],点击链接查看异常信息[Kibana 链接]

2　关键技术

2.1　前端数据收集

前端数据收集是系统服务入口监控的关键,能够直接反应用户的使用情况、客户端与网站的交互情况等[6]。由于 Web、App 都在用户端,收集方法和后台服务收集方法有所不同,需要建立专用通道收集数据,系统架构如图 3 所示。认证与 Session 管理提供了认证机制,防止服务被不可信方调用。并且为了防止数据被窃取和伪造,使用了 HTTPS 协议加密传输数据。前端收集服务处理前端数据的收集,把收集到的数据传入 Kafka,后续流程与后台数据收集流程相同。由于传入的流量大,前端收集服务需考虑负载均衡功能。部署多个前端收集服务,并配置加入负载均衡器。

图 3　前端数据收集

2.2　Elasticsearch Index 的管理

Elasticsearch 中的 Index 和传统数据库的表概念类似,采用单个或者多个 Index 取决于收集到的数据特征,如数据量、数据更新速度等。Index 通常按照时间划分存储,如按天、月等。而查询时则视多个按时间存储的 Index 为同一 Index,存储的划分对应用是透明的[7]。按时间存储的策略取决于对应的数据特征,由于前端收集到的数据量较大,可以单独配置 Index,并按天存储。而服务调用数据具有一定关联性,数据特征相似,可以把所有服务调用数据配置到同一个 Index,并按天存储。调用第三方服务的数据特征和服务调用不同,可靠性要求高,且要求能够进行后期跟踪、查询,也需要单独配置 Index,由于数据量不大可以按月存储。Elasticsearch 提供了 Index 保留功能,超过保留时间的 Index 会自动被删除,这方便了 Index 的维护。Index 具体配置如表 6 所示。

表 6　Elasticsearch Index 管理

监控数据源	Index 名	Index 存储策略	Index 保留策略
前端	frontend - *	按天	一个月
后端服务调用	backend_service - *	按天	一个月
第三方服务调用	thridparty_service - *	按月	一年

2.3　监控的高可用性要求

监控服务需要加载所有的监控规则,并在监控规则指定的时间执行监控任务。监控规则的数量和复杂

度取决于用户的配置和监控需求,若需要执行大量的监控任务,则对运行监控任务的平台提出了要求,如高可用性、性能等。监控服务的高可用性要求是一项重要的基本要求,若监控服务的不可用则会导致整个监控系统的不可用,所以在设计时应当避免监控服务的单点故障问题。Quartz 是一个开源的作业调度框架,支持集群搭建,并提供负载均衡、可伸缩、高可用性等功能特性[8]。使用 Quartz 集群是解决监控服务高可用要求的较好方法。Quartz 分布式框架通过 MySQL 关系数据库管理执行的作业,确保单个任务同一时刻在一台机器上运行。在系统中 Elasticsearch 数据库、MySQL 数据库都搭成集群模式,以保证系统的高可用性要求,部署结构如图 4 所示。

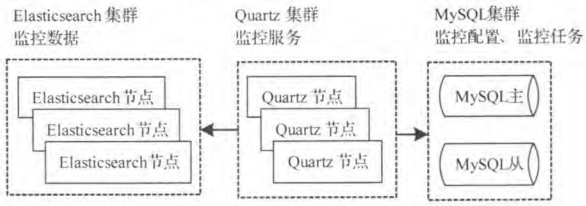

图 4 监控服务的集群部署

3 实现与应用示例

3.1 数据收集点的布置

本文介绍的监控系统实际应用到互联网网站中,对网站入口、后台服务的运行、第三方服务的交互进行了实时监控。网站的系统架构图如图 5 所示,其中虚线方框的模块布置了数据收集点,包括了前端、前端服务、后端服务、部分中间件。前端数据收集服务、PHP 层和数据访问服务数据量大,使用 API 方式收集数据;业务服务、服务治理框架、调用第三方服务则使用文件方式收集数据。客户端的监控数据通过前端收集服务收集;PHP 层收集点包括了网站入口和后端服务入口监控数据的收集。业务服务间的调用监控不需要在每个服务布置,而是把收集模块集成到服务治理模块中,这样业务模块则不需要增加额外处理逻辑来实现服务调用监控。同时业务服务还布置了对中间件调用以及第三方服务调用监控数据的收集。

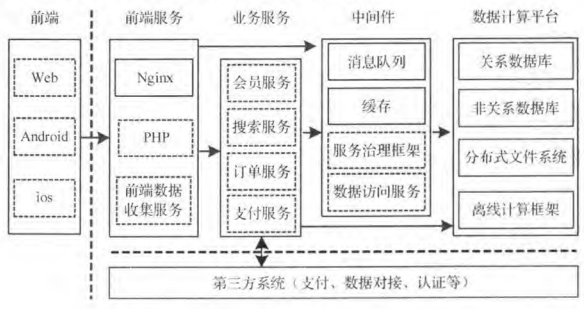

图 5 数据收集点部署

数据收集点的布置应当避免数据重复收集,如若服务调用方和被调用方都布置了收集点,则会造成服务调用监控数据重复,增加了数据库存储和监控数据的复杂度。在服务、中间件调用场景应尽量把收集点布置在调用方处,这样被调用方不可用、性能下降等异常可以被监控到。监控 API 中的消息字段用户后续的分析、追踪问题,在集成时不应存入无用、冗余、过长的信息。而 Elasticsearch 中的枚举字段则应由 API 内部填入,避免程序员直接填写,可以通过编程语言的枚举类型实现。

3.2 监控系统部署和配置

监控系统采用了高可用部署,收集点处的 Logstash 进程配置了服务管理器,可以在 Logstash 进程异常退出时重启进程;Kafka 搭建为集群模式,配置为两个节点;Elasticsearch 数据库搭建为集群模式,配置为三个节点,每个节点的存储为 2 TB。在 Elasticsearch 集群的一个节点上部署了 Kibana 服务,同时也配置了服务管理器;监控服务部署在 Quartz 集群上,Quartz 集群配置为两个节点。存储监控配置和任务的 MySQL 数据库则使用了主从模式配置。

3.3 监控规则的配置

根据布置好的监控点,可以配置的监控规则分为四类:

(1) *网站入口的监控* 网站入口的运行指标直接关系到用户体验,配置的优先级较高。对接口异常次数、异常率、接口平均调用时间配置了监控。配置了短信、邮件报警方式,可以第一时间通知相关人员。

(2) *服务调用的监控* 通过配置了服务接口的异常率、平均调用时间可以监控内部各服务的运行情况,包括了服务不可访问、服务异常、服务响应时间长等。

(3) *第三方服务调用的监控* 由于第三方服务为外部系统,易发生服务不可访问、网络不稳定等异常情况,故需要通过监控及时了解异常情况并采取相应措施。

(4) *服务内部关键点的异常监控* 服务内部的关键点监控需要开发人员在关键处加入数据收集逻辑,对某些指标进行预先计算、处理,再传入收集 API。收集的数据内容形式多样,取决于业务监控需求。

(5) *页面访问指标(PV,UV)的监控* 页面访问指标的监控可以反馈业务功能的访问、使用情况,对运营及产品设计有着重要的作用[9]。根据收集点的布置情况可以实现具体某个页面在不同条件下的 PV,UV 监控,同时在 Kibana 中可以对历史数据进行统计、分析。

(下转第 201 页)

试盗取用户凭证。一旦攻击者获得用户凭证,就可以获得资源所有者存储的敏感资源。

改进 OAuth2.0 授权机制的对策如下:

(1) 对于 AS 或任何请求响应的真实性问题,使用传输层的安全特性来解决。

(2) 用户凭证依靠 VGateway 中的 OTP 动态生成,并发送到 AS。该过程发生在服务提供程序中,因此,攻击者很难钓取到用户凭据。

4 结　语

本文通过防止攻击者伪装成授权服务器来改进 OAuth2.0 授权机制以解决钓鱼攻击问题。改进后的授权机制引入 OTP 和三方协商的方式,使得攻击者很难通过网络浏览器或移动设备进行钓鱼攻击。而且,为避免原始密码被盗,改进的 OAuth2.0 授权机制还引入了 VGateway 来安全生成用户凭证。本文还对改进 OAuth2.0 授权机制的有效性进行了理论方面的评估。

在以后的工作中,我们可以尝试借助模型检测工具来对改进的 OAuth2.0 授权进行建模分析。另外,也可以尝试针对 OAuth2.0 协议在实现过程中存在的其他类型攻击研究改善方案,从而达到进一步完善 OAuth2.0 认证授权系统的目的。

参 考 文 献

[1] 刘大红,刘明. 第三方应用与开放平台 OAuth 认证互连技术研究[J]. 电脑知识与技术,2012,8(22):5367 - 5369.

[2] 王焕孝,顾纯祥,郑永辉. 开放授权协议 OAuth2.0 的安全性形式化分析[J]. 信息工程大学学报,2014,15(2):141 - 147.

[3] Pai S, Sharma Y, Kumar S, et al. Formal Verification of OAuth 2.0 Using Alloy Framework[C]//International Conference on Communication Systems and Network Technologies. IEEE,2011:655 - 659.

[4] 魏成坤,刘向东,石兆军. OAuth2.0 协议的安全性形式化分析[J]. 计算机工程与设计,2016,37(7):1746 - 1751.

[5] Hardt D. The OAuth 2.0 authorization framework[EB/OL]. [2013-02]. http://tools. ietf. org/html/draft - ietf - oauth - v2 - 31.

[6] Al-Sinani H S. Browser Extension-based Interoperation Between OAuth and Information Card-based Systems[D]. Royal Holloway, University of London,2011.

[7] Bansal C, Bhargavan K, Maffeis S. Discovering Concrete Attacks on Website Authorization by Formal Analysis[C]// IEEE, Computer Security Foundations Symposium. IEEE Computer Society,2012:247 - 262.

[8] 时子庆,刘金兰,谭晓华. 基于 OAuth2.0 的认证授权技术[J]. 计算机系统应用,2012,21(3):260 - 264.

[9] Lodderstedt T. OAuth 2.0 Threat Model and Security Considerations[J/OL]. 2013. http://tools. ietf. org/html/rfc6819.

(上接第 162 页)

3.4 运行效果

监控系统在实际运行中取得了较好的效果,能及时报警网站入口异常、服务运行异常、第三方服务不稳定、与第三方服务间网络异常等异常情况。对页面访问指标的监控数据的统计分析,实现了 BI 系统无法完成的复杂统计功能。

4 结　语

本文介绍的监控系统在搜索引擎数据库 Elasticsearch、数据收集器 Logstash、可视化搜索与分析平台 Kibana 的基础上,扩充并完善了数据收集方法,提出了实时监控的具体方法,并在运用中不断完善,逐渐形成了稳定、可靠的监控系统。系统的实际运行效果较好,配置灵活方便、易于使用,对复杂软件系统的监控系统设计与实现具有参考意义。

参 考 文 献

[1] 方晓晴. 校园网站监控系统的设计与实现[D]. 大连:大连理工大学,2013.

[2] 张一. 网站综合监控系统的设计与实现[D]. 西安:西安电子科技大学,2013.

[3] 王余应. Zabbix 监控系统[M]. 北京:电子工业出版社,2015.

[4] 库赛, 罗格辛斯基. Elasticsearch 服务器开发[M]. 蔡建斌,译. 2 版. 北京:人民邮电出版社,2015.

[5] 王雪冰,胡宏伟,王炯炜,等. 基于短信报警的文件监控系统研究与实现[J]. 微计算机信息,2009,25(30):66 - 67,15.

[6] 王雅坤. 电商网站用户行为实时监控系统的设计与实现[D]. 北京:北京交通大学,2016.

[7] Lei X, Wang Z, He Y Z. The Data Management and Real-time Search Based on Elasticsearch[C]// International Conference on Computer, Mechatronics, Control and Electronic Engineering. 2015.

[8] 朱哲明. 基于 Quartz 的消息沟通平台的研究[D]. 北京:北京邮电大学,2015.

[9] 董冠军. 基于用户浏览行为分析的 Web 前端页面优化方法研究[D]. 天津:南开大学,2015.