一种改进的 ELK 日志采集与分析系统

鲜征征,叶嘉祥

(广东金融学院 互联网金融与信息工程学院,广东 广州 520521)

摘　要: ELK 架构是解决日志分布式采集与分析问题中具有代表性的解决方案。但原生 ELK 架构存在 Logstash 对 CPU 资源占用较大、无法动态更新日志相关配置和日志数据传输过程中数据易丢失等问题。为满足现实中大型分布式服务的日志采集与分析需求,对原生 ELK 架构作如下改进:在收集过程中增加限速器以减少 CPU 占用率;增加分布式注册中心实现相关配置动态更新;增加消息队列使得消息传输过程更健全。实验结果表明,改进后的 ELK 日志采集及分析系统与原生 ELK 相比,CPU 占用率减少了近 60%,日志速度提高了 3 倍,且消息丢失率为零。

关键词: 日志采集;分布式架构;ELK;消息队列

DOI: 10. 11907/rjdk. 191790　　开放科学(资源服务)标识码(OSID):

中图分类号: TP319　　**文献标识码:** A　　**文章编号:** 1672-7800(2019)008-0105-06

An Improved ELK Log Collection and Analysis System

XIAN Zheng-zheng, YE Jia-xiang

(*Department of Internet Finance and Information Engineering*, *Guang Dong University of Finance*, *Guangzhou* 510521, *China*)

Abstract: ELK architecture is a representative solution to the problem of distributed log collection and analysis. However, the native ELK architecture has problems such as Logstash consuming large CPU resources, unable to dynamically update log related configuration, and has data loss during log data transmission. In order to satisfy the needs of log collection and analysis of large distributed services in reality, the paper improves the native ELK architecture as follows: speed limiter is added to reduce CPU occupancy in the collection process; distributed registry is added to update the configuration dynamically; and message queue is added to make the message transmission process more sounder. The experimental results show that compared with the original ELK, the CPU occupancy of the improved ELK log acquisition and analysis system is reduced by nearly 60%, the log speed is increased by three times, and the excellent effect of zero message loss rate is achieved.

Key Words: log collection; distributed architecture; ELK; message queue

0　引言

如今,互联网在人们日常生活中扮演着不可替代的角色,当用户在访问、浏览或点击电子商务、社交娱乐、金融保险等网站时,都会产生相应的日志数据。这些日志数据是软件系统、硬件设备和用户行为的记录工具,旨在及时发现用户异常行为和软硬件故障,以确保网站稳定、安全运行,从而提升网站服务质量[1]。对于海量小文件存储随服务访问量上升的情况,大型网站通常在系统部署策略上采用分布式架构,但这会导致所产生的日志也分散在各台服务器中,而传统对多台服务器进行单机日志检索的方法效率低下。由此,基于分布式架构的日志采集与分析系统应运而生,各大互联网公司纷纷提出各种各样的日志监控平台方案。目前主流的监控平台有:基于传统 Web 框架的监控平台,基于数据索引的监控平台和基于大数据处理框架的监控平台。其中,集 Elasticsearch、Logstash 和 Kibana 为一体的基于数据索引的 ELK[2]开源框架,是实现企业通用轻量级大型日志监控平台的常用技术。ELK 的分布式日志采集与分析系统可根据不同场景,结合不同的工具或中间件,灵活简单地处理复杂日志问题,在分布式日志收集与分析领域得到了广泛研究和应用。

关于日志采集与分析技术,目前已取得一定研究成果。郝璇[3]基于 Apache Flume 设计并实现了一个适用于

收稿日期: 2019-06-13

基金项目: 广东省自然科学基金项目(2017A030313391);广东省科技厅国际合作项目(2017A050501042)

作者简介: 鲜征征(1977-),女,博士,广东金融学院互联网金融与信息工程学院讲师,研究方向为数据挖掘、隐私保护、机器学习、软件开发与测试;叶嘉祥(1996-),男,广东金融学院互联网金融与信息工程学院学生,研究方向为服务端开发。

异构网站平台间的分布式日志收集系统，该系统虽然可以将多个网站的日志集中存储，但缺少过滤和分析功能；廖湘科等[4]从日志特征分析、基于日志的故障诊断和日志增强 3 个方面分析了日志研究现状，然后通过对几种常用大规模开源软件的日志进行调研，发现一些日志相关的特征和规律以及现有工具难以解决的问题；李祥池[5]针对 ELK 在日志传输过程中可能丢失的情况，进行消息队列优化，利用 Redis、Kafka、Rockmq 等消息队列作为日志传输通道，达到了防止数据丢失、减缓服务器压力的效果；陈波[6]等针对 ELK 的 ElasticSearch 离线存储功能的性能瓶颈，进行存储功能拓展，利用大数据工具如 HBASE 等分布式存储系统提升日志的存储能力；张彩云等[7]通过对 ELK、Redis 进行整合，结合全国综合气象信息共享系统内蒙古气象局实际场景设计了 ELK 日志分析平台；姚攀等[8]针对 ELK 的 Logstash 对 CPU 有较大压力等问题，提出解析日志的规则和技巧，通过对 Logstash 作进一步优化，搭建能够对海量日志进行实时采集和检索的分析监控系统；汤网祥等[9]针对系统大规模部署下日志采集配置复杂、难以集中控制等问题，提出对各类源日志进行整合，并通过本地采集、网络汇集、集中管控、异步分级处理等方式加以实现；王参参等[10]根据银行业务发展需求，提出将全网范围日志分析系统由总行的一级中心平台和地级市的二级分行数据采集点组成，用以解决因日志规模过大造成的省级系统资源过载和网络堵塞等问题，但由于日志源众多，日志类型繁杂且未对收集的日志进行过滤，最终对业务系统运行带来一定影响；罗学贯[11]通过与互联网行业流行的配置管理数据库及监控系统进行关联，设计并实现了以 ELK 为基础的日志分析系统，利用 logstash 丰富的插件解决日志分析和处理难题，形成实际工作中丰富的日志字段，为重点日志增加了标识字段，为日志分析和故障诊断提供了便捷手段；袁华[12]提出 ELK 与大数据处理数据技术 Spark 相结合，用一种随机 key 后缀方式优化 Spark 集群性能，从而提高集群计算效率；严嘉维等[13]为提高可信计算平台日志数据分析效率，提出了一种基于 Hadoop 的可信计算平台日志分析模型，将实时日志与规则库匹配，实现对用户异常行为的检测，日志分析结果为主动防御提供了决策支持；王梦蕾[14]从网络安全角度设计与实现了一个基于 Spark Streaming 的实时日志分析与信息管理系统，在发生 DDoS 攻击时及时给予用户警告；孙鲁森[15]研究并使用 Flume 集群将 Web 应用集群所产生的日志进行汇总，搭建 Hadoop 集群并使用其内部组件 HDFS 来持久化 Flume 集群所汇总的日志数据，可以对网站的多维度 Page View、访客来源统计、用户关键路径转化进行多维度且详细的数据分析。但该研究与文献[13]都是基于 Flume 平台，Flume 自身比较繁琐，与 ELK 相比，它主要侧重于数据传输而非采集。随着容器虚拟化技术的进步与推广，主流的服务部署均挂载在宿主机的虚拟容器里，如 Docker[16]，但是容器会根据不同的策略分布与漂移，这样会导致日志过于分散。因此，基于如 Kubernetes[17]容器编排管理的大型日志收集系统相关研究受到关注。

针对现实中的复杂场景，以上方法都未从原生 ELK 架构自身缺点出发，原生 ELK 架构存在服务器处理效率低下、Logstash 对 CPU 资源占用过多、可扩展性差、无法动态更新日志相关配置和日志数据传输过程中数据易丢失等问题。本文通过使用 Etcd 搭建配置中心集群，使用 Gin 框架与 Nginx 实现日志产生器，模拟分布式系统下产生日志，使用 Vue.js 搭建前端，提出对收集模块和转发模块的改进方案，即实时收集产生的日志，同时能限制日志收集频率，从而替代 ELK 框架中的 Logstash，达到减少 CPU 压力的目的；还使用 Kafka 与 Zookeeper 搭建一个消息队列集群，将收集到的日志发送至消息队列中，达到削峰、保证消息健全的作用，并用 go 语言实现转发模块，从消息队列中拉取日志；最后使用 Elasticsearch 与 Kibana 构建日志存储与展示模块，向用户提供强大的日志搜索与分析功能。本文基于 ELK、Kafka 与 Etcd 等技术，通过对 ELK 的改进，弥补了原生 ELK 在大型分布式业务系统中不能直接使用的缺陷，设计并实现了一个日志健全、配置便捷、性能更高效的分布式日志采集与分析系统。

1 系统设计

1.1 系统业务架构设计

系统分为四大模块，分别是系统入口模块、日志收集模块、日志转发模块和日志展示模块，如图 1 所示。

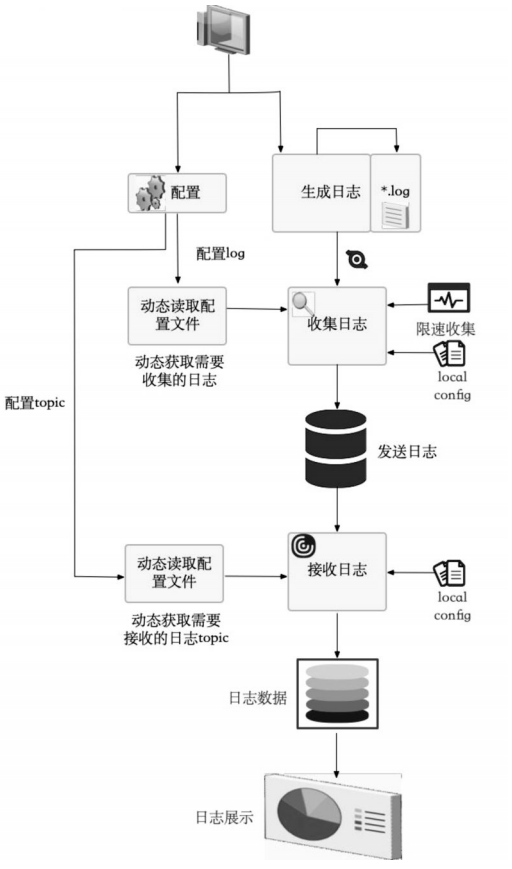

图 1 系统业务架构

图 1 中，其核心关注点在于日志收集模块和日志转发模块的消息健全、高性能以及高可用设计。系统入口模块直接对终端设备提供服务，同时入口模块通过调用 http 服务生成日志、配置文件等；日志收集模块主要是获取配置中心上的路径配置，对特定日志进行收集，并发送到消息队列中；日志转发模块主要是获取配置中心上的主题配置，对特定主题的日志进行过滤接收，并发送至 ElasticSearch；日志展示模块主要对日志进行搜索、分析和展示。

1.2　系统执行时序分析

系统业务在执行过程中是按照一定的时序进行，图 2 展示了本系统的业务执行时序。

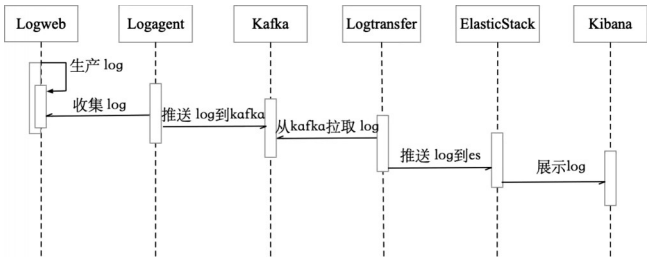

图 2　系统执行时序图

图 2 中，Logweb 服务是系统入口模块，产生日志；Logagent 是日志收集模块，对产生的日志进行收集，并将消息推送到 kafka 消息队列；Logtransfer 是日志转发模块，将从 Kafka 拉取的日志消息推送到 ElasticSearch，并且通过 Kibana 进行日志分析与展示。

1.3　系统物理架构设计

原生的 ELK 架构中没有嵌入消息中间件、配置中心等内容，因此，只适合日志量较少、业务简单的系统。随着业务逐渐复杂，ELK 原生架构已不能支撑庞大复杂的日志信息。因此，需要设计一个适合分布式系统的日志采集与分析系统，并考虑消息队列集群及应用集群。本系统针对此种场景，从分布式系统、高可用集群方面入手，对原生 ELK 架构进行改进。系统物理架构如图 3 所示。

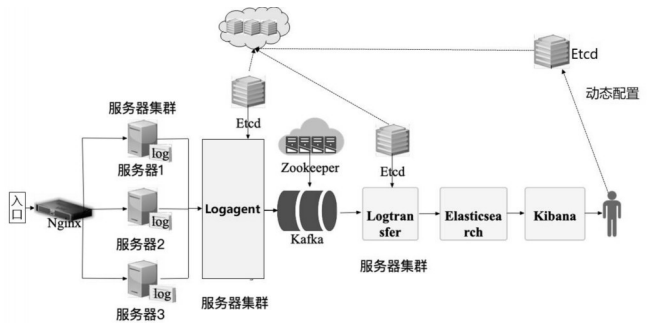

图 3　系统物理架构

图 3 中，Web 端的 http 请求首先会经过 Nginx 负载均衡，转发到服务器集群中 1~3 号服务器中的任意一台；服务器集群中的日志收集器，会通过 Etcd 集群中的配置路径，进行日志收集，然后将日志推送到 Kafka 集群；而 Kafka 集群之间的联系是靠 Zookeeper 集群进行服务注册与发现；日志转发器通过 Etcd 集群中的主题配置，从 Kafka 集群中选择相关主题的日志，拉取到服务器集群中；最后日志将发送至 ElasticSearch，通过 Kibana 连接 ElasticSearch 获取日志，进行日志展示与分析。

其中，日志消息队列 Kafka 的高可用与消息健全性尤为重要，其集群架构如图 4 所示。

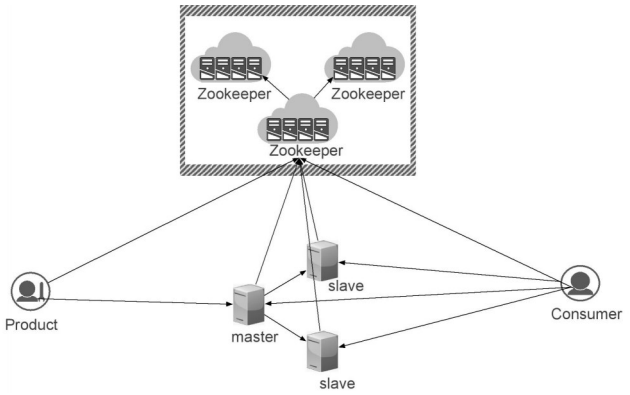

图 4　消息队列集群架构

图 4 中，product 是生产者，它从 Zookeeper 注册中心获取到 Kafka 集群的地址，然后将产生的消息发送至 Kafka 集群，而 Kafka 集群事先需要将自己本身的地址注册到 Zookeeper 集群，并且保持心跳检测，才能被生产者、消费者发现；consumer 是消费者，同理，它也从 Zookeeper 注册中心获取到 Kafka 集群的地址，然后去 Kafka 集群拉取消息。本系统的生产者指日志收集器，它将收集的日志发送至 Kafka 中，消费者是指日志转发器，将日志从 Kafka 中拉取下来。

2　系统功能模块

2.1　系统入口模块

系统入口模块主要作用是配置收集规则、产生日志，用 Nginx 与 Gin 框架搭建，对外开放 http 接口，网页端可以通过调用相关接口，完成该功能。Nginx 在本系统中的作用是作为业务应用服务器集群的负载均衡器，应用服务器集群架构如图 5 所示。图 5 中，所有服务器都处理相同业务，所有请求经 Nginx 转发到不同的服务器中。如果集群中的一个服务器发生故障，则 Nginx 能够迅速将该系统的任务转发到集群中其它正常工作的服务器上进行处理。业务具体处理则是用 Gin 框架支撑，大致分为两个步骤：①配置收集规则、收集路径等，主要完成对日志文件位置、日志主题过滤等功能模块配置；②模拟生产环境下的日志产生过程。

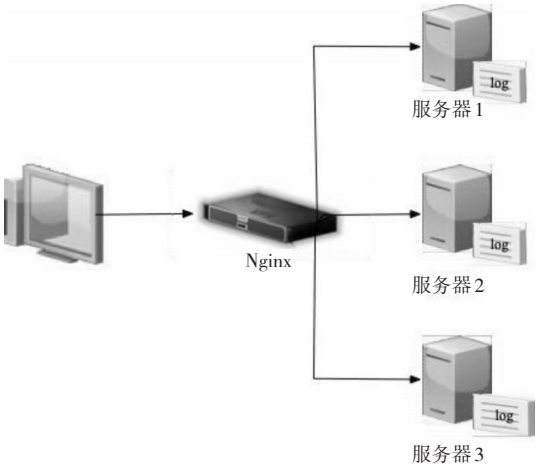

图 5 应用服务器集群架构

2.2 日志收集模块

该模块中日志收集器根据规则收集日志,并将日志发送到消息队列中,包括收集器、限速器和推送子模块。收集器和限速器主要根据配置规则收集相关日志,并配置特定的发送频率,达到限速目的,其 UML 设计如图 6 所示。

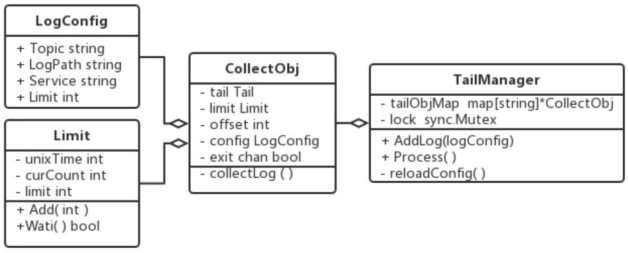

图 6 收集器与限速 UMI 设计

图 6 中,日志收集器模块分别由 TailManager 类(用作管理收集日志)、CollectObj 类(实现日志收集器)、LogConfig 类(实现日志配置)、Limit 类(解决日志限速)4 个类构成。

推送子模块负责将收集的日志推送到 kafka 消息队列中,其 UML 图设计如图 7 所示。该模块功能实现包括 Message 和 Sender 两个类。前者定义日志消息,后者解决日志消息推送。

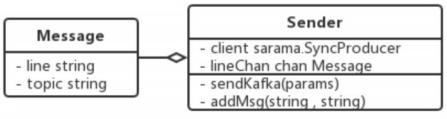

图 7 推送子模块 UMI 图

2.3 日志转发模块

该模块主要是转发器设计,转发器从消息队列中拉取日志,将日志转发到 ElasticSearch 中,消息拉取子模块,其 UML 图设计如图 8 所示。

日志转发器主要由 KafkaManager 类(用作管理转发 kafka 消息实例)、KafkaObj(kafka 消息实例)和 TopicMeta(描述日志消息)3 个类构成。

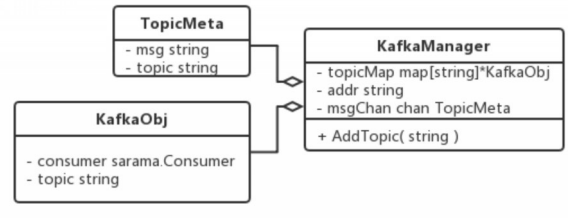

图 8 日志转发器 UML 图

2.4 日志展示模块

该模块主要用来展示日志,设计思想是首先配置 Kibana 索引,然后设置可视化图标,再过滤相关字段,最后显示相关日志信息。

3 关键技术

3.1 配置与搭建

实现过程中,需要对相关环境进行配置和搭建,包括 Ngnix、ElasticSearch 和 Kibana 配置、ElasticSearch 索引设计等。

3.1.1 Nginx 配置与搭建

一般而言,一个 Nginx 服务器反向代理多个应用服务器,通过配置 nginx.conf 创建应用服务器集群。nginx.conf 配置如图 9 所示。

```
upstream logweb_pool{
    server 127.0.0.1:8080 weight=4;
    server 127.0.0.1:8081 weight=4;
    server 127.0.0.1:8082 weight=4;

}
server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        proxy_pass http://logweb_pool;
        root   html;
        index  index.html index.htm;
    }
}
```

图 9 Nginx 配置

图 9 中,logweb_pool 配置了 3 台不同端口的应用服务器,其中 weight 表示权重;proxy_pass 将访问 localhost 的请求代理至 logweb_pool 的任意一个地址。

3.1.2 ElasticSearch 和 Kibana 配置与搭建

对 ElasticSearch 和 Kibana 进行配置与搭建,具体步骤为:①下载 ELK 套件中的 ElasticSearch 与 Kibana,安装并且配置 ElasticSearch 和 kibana 的端口,例如:本文分别配置的端口为 9200 和 5601;②用命令行开启运行 ElasticSearch 并访问 9200 端口,获得 ElasticSearch 相关信息;③用命令行开启 Kibana,访问 5061 端口,并且产生一条 yjxxx 的日志,在 Kibana 对其进行模糊搜索,搜索成功即搭建成功。

3.1.3　索引设计

为了方便日志的搜索、统计和分析，该系统为 ElasticSearch 配置了一个索引模板。系统可以根据模板，将日志数据进行分类。配置过程中有如下需求：①不同 IP 产生的日志索引和数据只会进入相应 IP 下的索引；②在搜索时，借助 alias 别名可以实现全局索引；③建立时间索引，便于系统根据时间戳可视化日志。

3.2　集群设计与搭建

在本系统中，主要的集群有消息队列集群与 Etcd 集群，其中构建消息队列集群，除消息队列本身外，还需要构建一个注册中心集群，对消息队列集群进行服务的注册与发现。需要注意的是：根据分布式理论，搭建一个集群最少为 3 台机器。以下集群搭建都是 3 台机器搭建而成。

3.2.1　消息队列集群搭建

系统采用 3 个不同的端口，配置模拟 3 台机器开启 Kafka，并连接 Zookeeper，搭建一个消息队列集群。首先需要为 3 台主机配置相应的端口，然后填写配置文件注册至 Zookeeper 集群，形成一个消息队列集群。具体而言，3 台 Kafka 通过连接 Zookeeper 集群，将各自的地址注册上 Zookeeper，Kafka 集群间能够互相发现，且选端口号为 9092 的为 Master，其它两个端口的 Kafka 为 Slave。其中，Master 为主机，Slave 为备机，构成主从结构的消息队列集群。

3.2.2　Etcd 搭建

该系统中，Etcd 是作为一个配置管理中心，对日志文件的位置、Kafka 消息的 topic 进行过滤管理。搭建方法是采用命令行运行 Etcd 服务端创建脚本，并且创建 Etcd 客户端连接服务端。Etcd 开启成功，通过 Raft 算法[18]选取某端口号的 Etcd 为 Leader，其它两个端口的 Etcd 为 Follower。其中，Leader 为领导者，负责监听网络，处理任务，Follower 为追随者，负责在 Leader 处理任务时，争取成为新 Leader。

4　实验及结果分析

4.1　实验环境

硬件环境：实验均在笔记本电脑上进行，Windows 10 64 位操作系统，Core i5-4200H 处理器，8G 内存，1T 硬盘。

软件环境：实验选用 Go 语言实现，系统的日志配置模块、日志生产器模块、日志收集器模块和日志转发器模块在 Goland 开发环境中实现，Kafka、Zookerper、Etcd、ElasticSearch、Kibana 等在 Power Shell 上开启。

4.2　实验结果及分析

4.2.1　模拟产生日志

实验模拟 5 000 个用户同时访问购物网站，即产生 5 000 个用户在购物网站上浏览、购买商品、支付和收货等日志，且设置每个用户产生 5~10 条不等的日志。

4.2.2　日志收集

对日志收集器进行初始化后监听 Etcd 集群配置变化以及日志内容变化。特别地，在日志收集时会根据限速条件对日志收集器进行限速处理。图 10 中，limit is running 代表限速模块对一秒内生产 5 000 次日志的文件进行限速，limit is exited 表示限速结束。

```
2019/03/05 17:09:38.995    limit is running, limit:5000 s.curCount:5000
2019/03/05 17:09:38.997    limit is running, limit:5000 s.curCount:5000
2019/03/05 17:09:38.999    limit is running, limit:5000 s.curCount:5000
2019/03/05 17:09:38.004    limit is running, limit:5000 s.curCount:5000
2019/03/05 17:09:38.005    limit is exited
2019/03/05 17:09:38.005    limit is exited
```

图 10　日志收集器限速处理情况

4.2.3　日志转发

对日志转发器进行初始化后监听 Etcd 集群配置变化。当日志转发器拉取到相关主题的日志消息后，会发送到 ElasticSearch。如图 11 所示，打印出日志信息表示已经成功收集，send to es success 表示成功将日志发送到 ElasticSearch。

```
2019/03/06 12:19:23.005    Partition:0, Offset:2199, Topic:path2, Key:,
Value:2019/03/06 12:19:11.430  users [45349041423] add goods [3458327]

2019/03/06 12:19:23.005    Partition:0, Offset:2200, Topic:path2, Key:,
Value:2019/03/06 12:19:11.430  users [48572239997] add goods [3242341]

2019/03/06 12:19:23.006    Partition:0, Offset:2202, Topic:path2, Key:,
 Value:2019/03/06 12:19:11.430  users [48572239997] ordering goods [3242341]

2019/03/06 12:19:23.009    Partition:0, Offset:2201, Topic:path2, Key:,
 Value:2019/03/06 12:19:11.430  users [48572239997] pay goods [3242341]

2019/03/06 12:19:23.011    Partition:1, Offset:2198, Topic:path2, Key:,
 Value:2019/03/06 12:19:11.430  users [34123524123] add goods [7845733]

2019/03/06 12:19:23.013    Partition:1, Offset:2200, Topic:path2, Key:,
 Value:2019/03/06 12:19:11.430  users [19937419341] add goods [4857344]

2019/03/06 12:19:23.014    Partition:1, Offset:2201 Topic:path2, Key:,
 Value:2019/03/06 12:19:11.430  users [69834572303] pay goods [3393221]

2019/03/06 12:19:23.022    send to es success
```

图 11　日志转发结果

4.2.4　日志搜索和分析

当日志成功收集后，需对这些日志数据进行搜索和分析，其结果如图 12 所示。柱状图表示每秒钟收集到的日志，其下是按时间先后顺序排列的日志数据，默认情况下是全量显示，若需要搜索特定规则的日志，可以通过搜索栏进行搜索匹配。

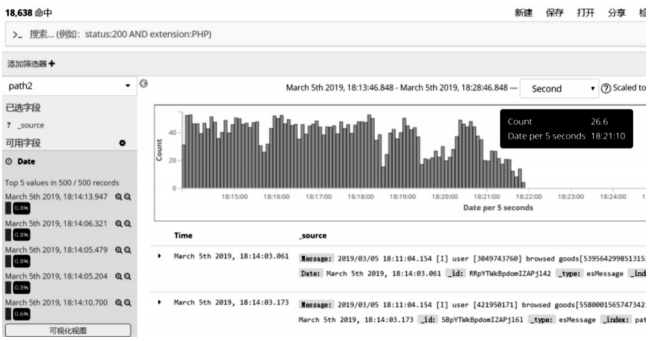

图 12　日志展示效果

通过上述实验，系统可以完成收集、转发和展示分析等功能，体现了其强大的分布式日志采集与分析能力。

4.3 对比实验

在同样的实验环境下，针对同一实验对象，将改进后的系统与原生 ELK 在性能方面作对比实验。

较原生 ELK 而言，改进后的系统优势主要体现在日志收集时多项性能的提升。具体压测方案设计如下：

（1）压测环境：Core i5-4200H 处理器，8G 内存，1T 硬盘。

（2）Logstash / Logagent 读取 500 000 条日志，单行数据 500B。

（3）压测实验一：正常收集日志数据，统计 CPU 平均占用率、总耗时和收集速度，结果如表 1 所示。

（4）压测实验二：在收集日志数据时，人工对网络进行实验性阻断，产生网络抖动，统计消息丢失数和丢失率，结果如表 2 所示。

表 1 压测实验一结果

收集器	CPU 占用率(%)	总耗时(s)	收集速度(line/s)
Logstash(原生)	75.6	30	16 667
Logagent(改进)	44.8	11	45 455

表 2 压测实验二结果

收集器	总消息数	消息丢失数	消息丢失率(%)
Logstash(原生)	500 000	698	0.14
Logagent(改进)	500 000	0	0

从表 1 可看出，原生 ELK 系统中 Logstash CPU 占用率高达 75.6%，而改进后的收集器 CPU 占用率降低到 44.8%，且收集速度为原生 ELK 的 3 倍。从实验结果来看，改进后的系统解决了原生 ELK 中 Logstash 对 CPU 资源消耗问题，且提升了日志收集速度与系统的稳定性。

从表 2 可以看出，在产生网络抖动时，原生 ELK 系统会丢失一些数据，500 000 条消息丢失了 698 条，消息丢失率达 0.14%，改进后的日志收集系统，500 000 条消息丢失 0 条，消息丢失率为 0。从实验结果来看，改进后的系统解决了原生系统日志消息容易丢失的问题，显著提高了日志收集过程中消息的健全性。

5 结语

随着云计算、物联网技术的迅速发展，数据已呈爆炸式增长，人们进入大数据时代[19]。而对于 Web 日志数据量的增加，文件的分布式存储和数据挖掘技术也成为 Web 日志挖掘研究领域的重点分支[20]。因此，日志的采集和分析将是其研究的重要支撑。

本文系统主要着眼于日志系统的简易实用性、高可用性与日志的健全性。简易实用性的实现途径主要是：可配置的 Web 管理后台、简易操作的搜索与分析 Web 页面等；高可用性的实现途径主要是：分布式、集群、负载均衡、限流等；日志的健全性实现途径主要是：消息队列、注册中心等。

下一步工作将关注两个问题：一是将日志收集与分析系统整合到 Docker 容器，并且用 Kubernetes 进行管理编排，这样能极大方便开发者和运维者，后续迭代开发也具有敏捷性；二是改进现有系统，增加分布式链路追踪系统，以获得服务之间调用的耗时、吞吐量、是否故障等信息，从而为运维和监控提供更好的参考。

参考文献：

[1] 应毅，任凯，刘亚军. 基于大数据的网络日志分析技术[J]. 计算机科学，2018，45(11A)：353-355.

[2] 高凯. 大数据搜索与日志挖掘及可视化方案—ELK Stack：Elasticsearch、Logstash、Kibana[M]. 第 2 版. 北京：清华大学出版社，2016.

[3] 郝璇. 基于 Apache Flume 的分布式日志收集系统设计与实现[J]. 软件导刊，2014，13(7)：110-111.

[4] 廖湘科，李姗姗，董威，等. 大规模软件系统日志研究综述[J]. 软件学报，2016，27(8)：1934-1947.

[5] 李祥池. 基于 ELK 和 Spark Streaming 的日志分析系统设计与实现[J]. 电子科学技术，2015，6(2) 674-678.

[6] 陈波. 基于 HBASE 分布式存储的通用海量日志系统设计方法研究[J]. 信息通信，2017(6)：7-9.

[7] 张彩云，牛永红，赵迦琪. ELK 日志分析平台在系统运维中的应用[J]. 电子技术与软件工程，2017(6)：182-183.

[8] 姚攀，马玉鹏，徐春香，等. 基于 ELK 的日志分析系统研究及应用[J]. 计算机工程与设计，2018，39(7)：2090-2095.

[9] 汤网祥，王金华，赫凌俊，等. 大规模软件系统日志汇集服务平台设计与实现[J]. 计算机工程与设计，2018，35(11)：173-178.

[10] 王参参，姜青云，李彤. 基于大数据的日志分析平台在银行中的研究与实现[J]. 网络安全技术与应用，2018(5)：68-70.

[11] 罗学贯. 基于 ELK 的 Web 日志分析系统的设计与实现[D]. 广州：华南理工大学，2018.

[12] 袁华. 基于 ELK 和 Spark 的日志分析系统的研究与实现[D]. 南昌：南昌大学，2018.

[13] 严嘉维，张琛. 基于 Hadoop 的可信计算平台日志分析模型[J]. 软件导刊，2018，7(8)：71-75.

[14] 王梦蕾. 基于 Spark Streaming 的实时日志分析与信息管理系统的设计与实现[D]. 哈尔滨：哈尔滨工业大学，2018.

[15] 孙鲁森. 基于分布式 Web 应用的大数据日志分析方法研究[J]. 电脑知识与技术，2019，15(3)：16-19.

[16] BOETTIGER C. An introduction to docker for reproducible research[J]. ACM SIGOPS Operating Systems Review-Special Issue on Repeatability and Sharing of Experimental Artifacts，2015，49(1)：71-79.

[17] BERNSTEIN D.Containers and cloud：from LXC to docker to kubernetes[J]. IEEE Cloud Computing，2014，1(3)：81-84.

[18] DIEGO ONGARO，JOHN OUSTERHOUT. In search of an understandable consensus algorithm[C]. In Proc ATC'14，USENIX Annual Technical Conference，2014.

[19] 孟小峰，慈祥. 大数据管理：概念·技术与挑战[J]. 计算机研究与发展，2013，50(1)：146-169.

[20] 班秋成. 基于 Hadoop 的 Web 日志存储和分析系统的研究与实现[D]. 北京：北京邮电大学，2018.

（责任编辑：孙 娟）