计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(10):82−88 [doi: 10.15888/j.cnki.csa.005997]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Docker 的大规模日志采集与分析系统①

罗东锋[1,2], 李　芳[1], 郝汪洋[1,2], 吴仲城[1]

[1](中国科学院 强磁场科学中心, 合肥 230031)
[2](中国科学技术大学, 合肥 230026)

摘　要: 传统日志分析技术在处理大规模日志时存在效率低、功能简单、实际应用扩展性弱等问题. 为解决这些问题, 设计了基于 Docker 的大规模日志采集与分析系统. 系统分为数据采集、数据缓存、数据转发、数据存储、数据检索和展示五层, 支持从不同数据源采集各种类型的日志, 通过 Kafka 消息队列提供可靠数据传输, 利用 Elasticsearch 实现数据分布式存储和检索, 并以可视化方式分析日志. 同时采用 Docker 容器技术实现系统的快速部署和版本控制. 该系统具有实时性、可扩展性、易部署等特点. 实验结果表明了该系统可行有效, 具有良好的实用价值.

关键词: 日志采集与分析; Docker; elasticsearch; 开源; 可扩展性; 实时

引用格式: 罗东锋,李芳,郝汪洋,吴仲城.基于 Docker 的大规模日志采集与分析系统.计算机系统应用,2017,26(10):82–88. http://www.c-s-a.org.cn/1003-3254/5997.html

Large-Scale Log Collection and Analysis System Based on Docker

LUO Dong-Feng[1,2], LI Fang[1], HAO Wang-Yang[1,2], WU Zhong-Cheng[1]

[1](High Magnetic Field Laboratory, Chinese Academy of Sciences, Hefei 230031, China)
[2](University of Science and Technology of China, Hefei 230026, China)

Abstract: The traditional log analysis technology has low efficiency, simple function, poor scalability in practice in processing the large-scale log. To solve these problems, a large-scale log collection and analysis system based on Docker is designed. There are five layers including data collection, data cache, data forwarding, data storage, data retrieval and display in the system. And the system can take in any type of log files from different data sources, provide reliable data transmission through Kafka message queue, utilize Elasticsearch to realize distributed storage and retrieval of data, and analyze log by means of visualization. Meanwhile, the use of docker container technology can realize rapid deployment and version control of the system. The system has the characteristics of real-time, scalability, easy deployment and so on. The experimental results show that the system is feasible and effective with good practical value.

Key words: log collection and analysis; Docker; elasticsearch; open source; scalability; real time

1　引言

随着大数据时代的到来, 数据的重要性日益凸显, 大数据的研究和应用[1]带来了巨大的商业价值和社会价值. 日志是互联网企业日常运营生产和积累的海量数据中最有价值的数据之一. 根据日志的来源和类型的不同, 日志大致划分为系统日志、应用业务日志和安全日志这三类. 通过系统日志能够有效监控系统的运行状况, 及时发现及解决潜在的问题, 提高系统的服务质量. 应用业务日志包含业务信息和用户访问信息, 可用于分析用户行为[2]. 安全日志用于安全审计及取证, 发现攻击或者非法操作行为, 提高系统的安全性. 日志通常带有时间戳的属性, 是递增的事件序列. 随着

① 基金项目: 国家自然科学基金 (61273323)
收稿时间: 2017-01-13; 采用时间: 2017-02-23

用户量的增长和用户需求的不断变化, 互联网应用系统的规模不断扩大、复杂度不断加深, 系统产生的日志数据量也在急剧增长. 大规模日志的实时收集、存储和分析仍然面临诸多问题和挑战:

① 日志种类繁多, 比如 Tomcat 服务器日志、Nginx 服务器日志、MySQL 数据库日志、操作系统日志等, 由于格式不一致导致分析困难;

② 日志来源分散, 不仅存在于不同的服务器, 而且分布在同一服务器的不同文件中;

③ 日志产生的速度快, 占用服务器的存储空间不断增加;

④ 日志具有时效性, 日志产生后需要在短时间内进行分析;

⑤ 缺乏有效的日志可视化分析手段.

针对上述问题, 日志分析系统在处理速度、可扩展性、实时性、分析纬度等方面提出了更高的要求. 随着大数据技术的发展, 有不少新的日志采集分析方案被提出. Scribe[3]是 Facebook 开源的一个基于 Thrift 远程服务调用框架的日志收集系统, 它为日志分布式收集、统一处理提供一个可扩展的简单方案. 但 Scribe 存在单点故障问题, 比如中央服务器异常退出会造成内存中数据的丢失, Facebook 已经不再更新和维护 Scribe. Apache Chukwa[4]是一个用以监控大型分布式系统的开源数据收集系统, 它是基于 Hadoop 的 HDFS 和 Map/Reduce 框架之上构建的, 具备可伸缩性和鲁棒性. 由于依赖于 Map/Reduce 框架去处理数据, 导致数据流在数据处理间断时吞吐量急剧下降, 严重影响效率. 赵龙等人利用 Hadoop 分布式计算平台和 Hive 数据仓库进行日志分析[5]. 该方法采用批处理的方式处理数据, 实时性较差, 适合于数据离线挖掘, 且不支持全文检索. 结合现有各类日志采集分析系统的研究基础, 本文基于 Docker 容器技术, 设计了一种分层且可扩展的日志系统, 实现大规模互联网应用场景下的日志快速分析.

2 系统架构设计

系统整体结构分为数据采集层、数据缓存层、数据转发层、数据存储层、数据检索和展示层, 如图 1 所示. 数据采集层从不同的数据源中实时获取最新的日志信息, 并对其进行预处理后依次传输到数据缓存层. 数据转发层实时从缓存层拉取数据转发到数据存储层. 数据存储层把接收的数据写入磁盘, 并建立索引库. 数据检索和展示层连接到数据存储层, 提供统一的日志信息查询和分析入口.

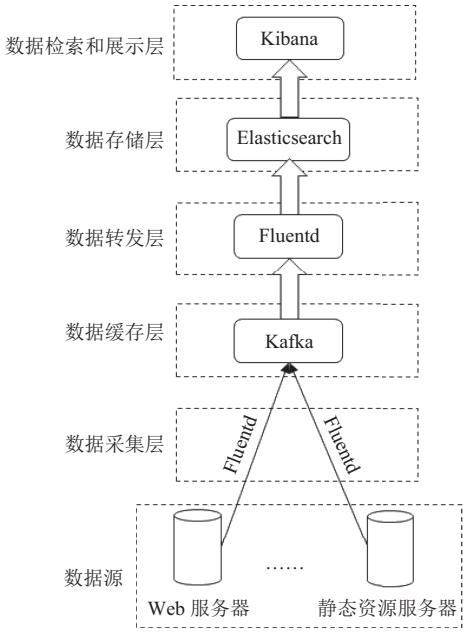

图 1 日志采集与分析系统架构

系统各层利用 Docker 容器虚拟化技术, 实现应用程序的快捷部署, 并且支持横向扩展, 以便应对更大规模的日志收集、存储和分析. 基于该系统架构, 系统各层之间通过网络传输方式进行交互, 实现系统层级的独立解耦.

2.1 Docker 容器虚拟化技术

Docker[6]是 Docker, Inc.公司开源的一个基于 Go 语言编写的轻量级应用容器引擎, 用于自动化部署应用. Docker 容器通过将应用和该应用运行所依赖的系统库、系统工具、源码等均打包到一个文件系统中, 从而确保了应用在任何环境下均能实现一致性的运行效果. 用户可在宿主机部署运行多个 Docker 容器, 每个容器拥有自己的资源并与其它容器相互隔离, 容器之间支持相互通信. Docker 采用类似于 Git 版本控制系统的思想, 实现对 Docker 镜像 (Image) 的版本管理, 比如通过 docker commit 命令从容器创建一个新的镜像. 此外, 在绝大多数情况下 Docker 容器的性能优于传统虚拟机 KVM, 接近原生的性能[7].

根据 Docker 的工作流程图 (图 2), Docker 的生命周期主要包括镜像 (Image)、Docker 仓库 (Repository)、

容器 (Container) 三个部分. Dockerfile 由一系列指令组成, 用于自动创建 Docker 镜像. Docker 镜像可以理解为一个只读模板, 而容器是由镜像创建的运行实例, 具有运行、暂停、停止三种状态. Docker 容器负责运行以及隔离应用程序. Docker 注册服务器 (Registry) 是用来存放各种仓库的中心, 每个仓库集中存放某一类镜像, 每个镜像文件拥有对应的标签 (tag). 仓库又分为公有仓库和私有仓库, 用户创建镜像后可以通过 push 上传到仓库, 需要时再用 pull 从仓库中下载. 用户既可在公有仓库发布镜像, 也可部署自己的私仓仓库发布镜像. 另外, 镜像制作遵循复用原则, 在其他镜像的基础上进行构建.

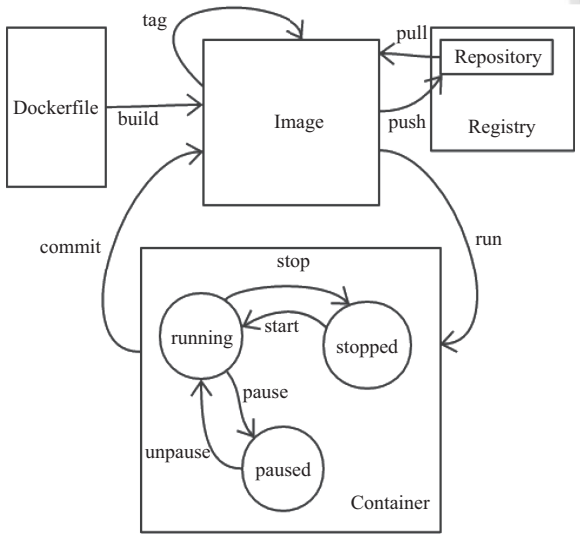

图 2 Docker 工作流程图

在容器数据管理方面, Docker 通过挂载宿主系统文件或文件目录作为数据卷 (Volume) 实现与宿主系统数据共享, 也可以通过数据卷容器实现容器之间的数据共享. 举个例子, 下面一条命令表示运行一个 Elasticsearch 容器, 并将宿主系统/home/elasticsearch/config 目录挂载到它的/usr/share/elasticsearch/config 目录:

docker run -d -v/home/elasticsearch/config:/usr/share/elasticsearch/config elasticsearch:2.4.1

2.2 数据采集层

数据采集是日志分析的基础. 日志分散在各个服务器上不同文件中, 其类型也不一定相同. 数据采集工具从不同的数据源获取日志信息, 进行预处理后传送到数据缓存层. 通常一条日志记录包含时间戳、来源和数据. 时间戳表示一条日志记录生成的时间. 来源标识产生该条日志的系统. 数据就是日志信息中最核心的内容. 但这种格式并不是一个标准, 不同系统产生的日志格式也各不相同. 为获得有价值且便于分析的日志信息, 需要对原始日志进行预处理. 预处理主要是对日志进行过滤和加工操作. 比如只传输符合规则的日志行, 给日志行添加用以区分来源的字段等. 日志实时采集传送到数据缓存层, 已采集的日志文件可定期删除, 而不必在服务器存储原始日志文件, 有效减轻了服务器的负载.

目前主流的开源数据收集工具有 Logstash、Fluentd 等. Logstash 是实时收集、加工、转发数据的工具, 支持处理各种常见类型的日志. 根据不同的功能, Logstash 分为日志收集者和日志转发者. 本文中日志收集者负责从日志文件中实时读取最新的行文本, 处理后输出到数据缓存层. 而日志转发者从数据缓存层中拉取数据, 加工处理后转发到数据存储层. 同样, Fluentd[8]也具备日志收集和日志转发的功能. Fluentd 有 Input、Parser、Filter、Output、Formatter 和 Buffer 六种类型的插件, 通过灵活的插件机制, 实现多种数据源的输入和输出. Fluentd 还支持基于内存和基于文件的缓存, 以防止内部节点数据的丢失, 也支持故障转移. 表 1 为这两种数据采集工具的比较.

表 1 Logstash 与 Fluentd 对比

	Logstash	Fluentd
开源	是	是
运行环境	JRuby语言实现, 依赖Java虚拟机环境	CRuby语言实现, 依赖Ruby环境
数据过滤	支持	支持
多行匹配	支持	插件支持
通用日志解析	Grok正则表达式解析	正则表达式解析
数据发送压缩	插件支持	插件支持
文件轮转	支持	支持
插件	丰富	丰富
性能	占用内存和CPU资源较多	占用内存和CPU资源较少

基于可靠性和性能的考虑, 本系统采用 Fluentd 实现数据采集层. Fluentd 配置文件示例如下:

```
<source>
  @type tail
  format /^(?<message>.*)$/
  path /home/dongfeng/access.log
  pos_file /data/logs/td-agent/nginx-access.log.pos
  tag access_log.tag
```

```
</source>
<match access_log.**>
  @type kafka
  zookeeper 192.168.19.104:2181
  default_topic access_log_raw
  compression_codec gzip
</match>
```

该配置文件包括 source 指令和 match 指令. source 指令中指定从文件中读取数据, match 指令则配置了数据输出目的地为 Kafka 服务器.

2.3 数据缓存层

为了解决数据采集的吞吐量超过数据处理容量而造成数据的丢失, 在数据采集层和数据转发层之间添加一层数据缓存层. 该缓存层采用消息队列的技术实现, 提高了系统的性能和可靠性. 目前提供消息队列服务的工具, 主流的有 Redis 和 Kafka. Redis[9]是一个开源、高性能、基于内存的键值对数据库. 它支持数据的持久化和订阅发布机制等高级特性. Kafka[10]是一种开源分布式发布订阅消息系统, 具有高性能、高吞吐率、水平扩展等特性. Kafka 支持数据离线处理和实时处理. 本文选用 Kafka 消息队列实现数据缓存层, 因为 Kafka 提供消息持久化能力和容错性保证, 在可靠性方面优于 Redis. Redis 不保存已发送出去的消息, 而 Kafka 可保存被消费的消息的副本. 此外, Kafka 在处理海量日志方面更具优势, Kafka 利用磁盘存储消息队列的数据, 而 Redis 则受物理内存的限制. 在消息压缩方面, Kafka 支持 snappy、gzip 等多种压缩方式. 尽管消息压缩需要消耗少量的 CPU 资源, 但是由于大规模的日志传输瓶颈主要在于网络 IO, 启用消息压缩机制能够有效减少网络传输的数据量, 提高日志传输效率.

在集群方面, Kafka 使用 ZooKeeper(分布式协调服务框架) 实现 Kafka 各组件的服务协调. Kafka 发布订阅的对象是话题 (Topic), 一个话题就是一类消息, Kafka 通过从物理上划分一个或多个分区 (Partition) 实现对话题的管理. Kafka 实现了一个分区内消息的有序性, 但不能保证不同分区之间的消息有序. Kafka 为同一数据源的所有日志创建一个话题. 本系统为保证一个话题中所有消息的有序性, 将该话题的分区数设置为一个. 发送消息到话题的一方称为生产者 (Producer). 数据采集层就是消息队列的消息生产者. 而消费者 (Consumer) 是订阅话题消费消息的一方. 若话题中只有一个分区, 消费者消费该话题中消息的顺序与该分区中消息先后顺序一致. 一个 Kafka 集群由一个或多个代理 (Broker) 服务器组成. Kafka 的容错性是通过副本 (Replica) 机制实现. Kafka 允许用户为每一个话题设置副本数量. 副本是以分区为单位创建的. 每个分区可以有一个或多个副本, 每个副本保存在不同的代理上. 若分区只有一个副本, 则该副本就是领导者 (leader), 负责处理该分区的读写请求. 若分区中有多个副本, 则其中一个副本会被选举为领导者, 其他的副本则充当跟随者 (follower) 的角色. 跟随者会被动去复制领导者上的数据. 当领导者发生故障时, 会从这些跟随者中选举新的领导者.

2.4 数据转发层

数据转发层作为数据缓存层中消息队列的消息消费者, 从消息队列中拉取消息并转发到数据存储层进行处理. Fluentd 作为数据转发层进行日志消息的拉取和转发. 为降低海量日志对数据存储带来的冲击, 数据转发层首先通过正则表达式对待分析的日志进行匹配, 过滤无用的日志信息, 仅对关注的日志进行提取和转发.

以 Nginx 访问日志为例, 日志各字段含义如图 3 中标注所示. 若使用 Fluentd 对此类型的日志提取数据, 可采用以下正则表达式进行匹配:

```
^(?<client_ip>\S+)\s-\s-\s\[(?<timestamp>[^\]]*)\]\s"
(?<request_url>[^\"]+)"\s(?<status>\d+)\s(?<size>\d+)?$
```

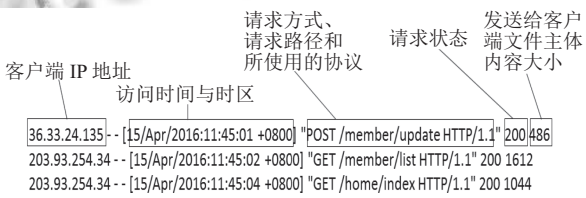

图 3 Nginx 访问日志

在数据提取时, 通过正则表达式匹配后的各字段被提取出来并命名. 例如, 日志中的客户端 IP 地址被命名为 client_ip, 访问时间与时区被命名为 timestamp. 通过对字段进行命名, 有利于为后续日志分析提供关键信息.

2.5 数据存储层

数据存储层是本系统最核心的一层. 数据转发层将转发的日志逐条插入到数据存储层的 Elasticsearch

集群. Elasticsearch[11]是一个高可用、高伸缩、基于 Apache Lucene 的开源分布式搜索和分析引擎. 虽然 Elasticsearch 与传统的关系型数据库 (Rational Database) 存在很大的差异性, 但是两者在一些核心概念上有相似之处, 见表 2.

表 2 Elasticsearch 和 Rational Database 对比

Elasticsearch	Rational Database
索引(Index)	数据库
类型(Type)	表
文档(Document)	行
字段(Field)	列

Elasticsearch 既支持全文检索, 也可以存储数据. 它存储不依赖模式 (Schema) 的 JSON(JavaScript Object Notation) 文档, 并对每一个文档进行索引. 一条日志对应一个文档. Elasticsearch 运行在 Java 虚拟机之上, 通过 RESTful API 接口的方式对数据进行增删改查操作. 操作时, 若有参数输入, 参数的格式以 JSON 形式表示, 操作后输出的结果也是以 JSON 形式显示. 图 4 表示使用 curl 命令删除一条文档. 可以看得出 RESTful 风格的 API 结构清晰、易于理解.

图 4 curl 命令删除一条文档

Elasticsearch 是分布式的, 可通过管理节点 (Node) 实现扩展. 节点是运行 Elasticsearch 的实例. 一个集群 (Cluster) 是一组具有相同集群名称的节点集合. 集群中节点可以配置为主节点、数据节点和客户端节点这三种不同的角色. 主节点控制集群, 负责集群的管理工作, 比如创建或删除数据索引. 数据节点负责存储数据分片和数据操作, 比如数据分片增删改查操作. 客户端节点将到来的请求转发到集群的其他节点, 起到负载均衡的作用.

当集群需要存储超出单个机器容量的数据时, Elasticsearch 会自动将数据分别发送至多个存储 Lucene 索引的机器上, 这些 Lucene 索引称为分片 (Shard) 索引. 同时 Elasticsearch 通过副本 (Replica) 机制对分片进行数据冗余, 保证了集群的高可用. 另外, Elasticsearch 支持插件机制, 包含丰富且功能强大的插件. 在分析含有中文内容的日志时, 需要对中文检索, 而 Elasticsearch内置的标准分析器对中文分词支持相对较弱. 它在处理分词时, 将中文的每一个汉字作为一个词 (token) 分开, 这种方式搜索的结果查全率非常高, 而查准率很低, 无法满足复杂或者特定的中文搜索需求. 本系统中 Elasticsearch 采用 IK analyzer 中文分词器[12]进行中文搜索, 效果较为理想.

2.6 数据检索和展示层

仅从每一条日志信息中难以发现海量日志中隐含的规律. 数据检索和展示层为数据存储层中所有日志信息提供统一的查询入口, 并可对日志进行统计分析以可视化形式呈现结果. 日志可视化将复杂和抽象的日志信息转成图表形式, 使得用户更容易理解日志之间的联系和变化情况, 便于用户更好地做出决策. Kibana 和 Grafana 都支持 Elasticsearch 数据可视化展示. Grafana 是开源的可视化测量数据的工具, 但对 Elasticsearch 数据检索支持较弱. 而 Kibana 是基于 Apache License2.0 开源协议, 使用 HTML 语言和 JavaScript 编写, 可提供数据分析和可视化的 Web 前端应用. 它可以对 Elasticsearch 中的数据进行搜索、分析和以统计图表的方式展示结果. Kibana 的查询语法是基于 Lucene 的查询语法, 通过布尔运算符、通配符和字段筛选进行模式匹配搜索. 用户可选择对查询的结果进行保存, 方便以后再次查看. Kibana 可以实时查看数据存储层中最新存储的日志信息.

3 实验测试及结果分析

本文重点验证系统整体的有效性以及 Fluentd 的数据采集效率. 通过部署在同一局域网的 9 台 PC 机搭建集群进行测试. 每台 PC 机的硬件环境为主频 3.1 GHz 的四核 CPU、内存 4 GB、磁盘 200 GB, 软件环境为 CentOS7.2(64 bit)、Docker1.10.3、JDK1.8.0_65、Kafka2.11-0.10.0.1、Elasticsearch2.4.1、Kibana4.6.2、ZooKeeper3.4.9、Fluentd0.12.30. 图 5 为本次实验系统部署架构, 集群各个节点的配置及其功能描述如表 3 所示. 实验数据集为模拟 Nginx 日志和 Tomcat 日志, 实验时不断地往被收集的日志文件中写日志.

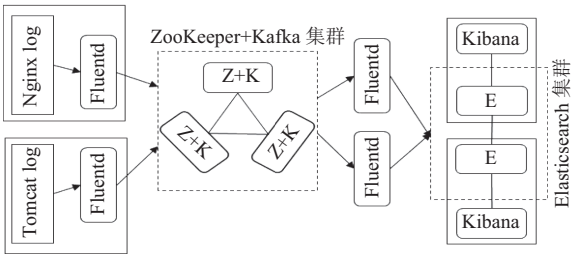

图 5 系统部署架构图

表 3 系统集群配置情况

IP地址	部署的软件	功能描述
192.168.19.102	Fluentd	采集Nginx日志
192.168.19.103	Fluentd	采集Tomcat日志
192.168.19.104	Kafka, ZooKeeper	缓存
192.168.19.105	Kafka, ZooKeeper	缓存
192.168.19.106	Kafka, ZooKeeper	缓存
192.168.19.107	Fluentd	转发Nginx日志
192.168.19.108	Fluentd	转发Tomcat日志
192.168.19.109	Elasticsearch, Kibana	存储, 检索和展示
192.168.19.110	Elasticsearch, Kibana	存储, 检索和展示

经过测试, Fluentd 的内存使用率变化不大, 大概占 2% 左右. Fluentd 数据采集速率与 CPU 使用率基本上呈线性关系, 峰值数据采集速率可达 2.7 万条日志/秒. 数据产生到数据展示耗时 30 秒以内, 基本满足日志分析实时性的需求. 在 Kibana 中可以看到最新收集的日志信息, 并能以图表形式进行统计分析, 图 6 为统计某一时间段客户端 IP 访问次数.

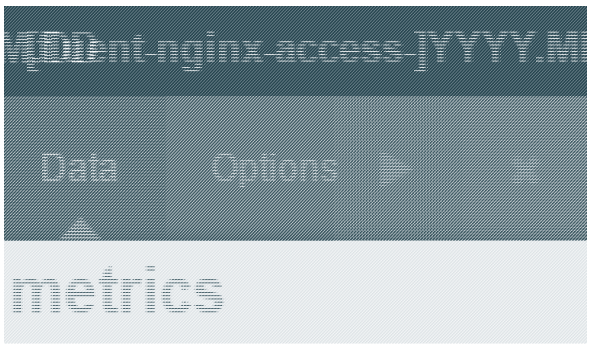

图 6 统计某一时间段客户端 IP 访问次数

Elasticsearch 中添加 IK Analyzer 分词器后, 在 Kibana 中检索“中国科学技术大学”, 得到较好的分词结果, 图 7 高亮显示了有实际意义的词组.

传统日志分析方案通常采用单机处理, 它的资源利用率不高且可靠性较弱. 传统日志分析方案与本文设计方案的各项指标对比见表 4.

图 7 日志关键词检索结果

表 4 传统日志分析方案与本文设计方案对比

	传统日志分析方案	本文设计方案
部署环境	高性能计算机	廉价的PC机集群环境
功能	局限于某个特定场景	有检索, 统计分析等功能
性能	低效	高效
耦合度	较高	分层, 耦合低
扩展性	难以扩展	易扩展
时效性	离线处理, 非实时	实时
部署效率	效率低	Docker容器部署, 效率高

本系统还具备快速部署的优点. 传统方式需要对新部署的服务器重新配置应用的运行环境, 而 Docker 容器部署方式只需要拷贝 Dockerfile 文件、构建镜像、运行容器, 即可完成部署, 效率远高于传统部署方式. 随着部署节点不断增多, Docker 容器部署方式的优势更加明显.

4 结语

本文设计了基于 Docker 的大规模日志采集和分析系统, 该系统结合了 Fluentd 高效数据采集工具、Kafka 消息队列、Elasticsearch 分布式搜索引擎、Kibana 数据分析和可视化平台、Docker 容器等开源技术. 本系统主要分为数据采集、数据缓存、数据转发、数据存储、数据检索和展示五层. 该系统实现日志分析的分层解耦, 能够实时高效对大规模日志进行处理分析, 并以可视化方式展示. 同时利用 Docker 虚拟化容器技术实现系统快速部署和版本控制, 提高了运维效率. 实验结果分析表明, 本文采用的技术路线和设计方法可行有效. 下一步将结合 Hadoop、Spark 等其他大数据分析技术对日志进行更深层次的挖掘, 从而获得更多有价值的信息.

参考文献

1 涂新莉, 刘波, 林伟伟. 大数据研究综述. 计算机应用研究, 2014, 31(6): 1612–1616, 1623.

2 Suneetha KR, Krishnamoorthi R. Identifying user behavior

by analyzing web server access log file. IJCSNS International Journal of Computer Science and Network Security, 2009, 9(4): 327–332.

3 Scribe logfile aggregation system described by facebook's Jeff hammerbacher. https://github.com/facebook-archive/scribe. 2008.

4 Rabkin A, Katz RH. Chukwa: a system for reliable large-scale log collection. Proc. the 24th International Conference on Large Installation System Administration. Berkeley, CA, USA. 2010. 1–15.

5 赵龙, 江荣安. 基于 Hive 的海量搜索日志分析系统研究. 计算机应用研究, 2013, 30(11): 3343–3345.

6 Boettiger C. An introduction to Docker for reproducible research. ACM SIGOPS Operating Systems Review, 2015, 49(1): 71–79. [doi: 10.1145/2723872]

7 Felter W, Ferreira A, Rajamony R, *et al*. An updated performance comparison of virtual machines and Linux containers. Proc. of 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). Philadelphia, PA, USA. 2015. 171–172.

8 Fluentd. http://docs.fluentd.org/articles/quickstart.

9 Carlson JL. Redis in action. Shelter Island, NY: Manning Publications Co., 2013.

10 Kreps J, Narkhede N, Rao J. Kafka: a distributed messaging system for log processing. Proc. of the NetDB'11. Athens, Greece. 2011. 1–7.

11 Gormley C, Tong Z. Elasticsearch: the definitive guide. Sebastopol, CA: O'Reilly Media, Inc., 2015.

12 朱潜, 吴辰铌, 朱志良, 等. Hadoop 云平台下 Nutch 中文分词的研究与实现. 小型微型计算机系统, 2013, 34(12): 2772–2776. [doi: 10.3969/j.issn.1000-1220.2013.12.022]