计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(8):121−126 [doi: 10.15888/j.cnki.csa.007611]

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

基于 Kafka 和 Kubernetes 的云平台监控告警系统①

郝鹏海, 徐成龙, 刘一田

(南京南瑞信息通信科技有限公司, 南京 210003)
通讯作者: 郝鹏海, E-mail: haopenghai@sgepri.sgcc.com.cn

摘　要: 为了实现对容器云、主机设备以及业务系统的实时监控, 设计了一种基于 Kafka 和 Kubernetes 的云平台监控告警系统. 通过 Kubernetes 对 Docker 容器进行管理, 通过 Kafka 接收不同地区不同主机的设备运行信息, 通过探针对业务系统进行监控, 并且通过告警的关联规则设置, 减少了冗余告警, 增强了告警的故障检测能力, 提高了告警的准确度.

关键词: Kubernetes; Kafka; 实时监控; 关联告警规则; 故障检测

引用格式: 郝鹏海,徐成龙,刘一田.基于 Kafka 和 Kubernetes 的云平台监控告警系统.计算机系统应用,2020,29(8):121–126. http://www.c-s-a.org.cn/1003-3254/7611.html

Monitoring and Alarm System for Power Grid Cloud Platform Based on Kafka and Kubernetes

HAO Peng-Hai, XU Cheng-Long, LIU Yi-Tian

(Nanjing NARI Information and Communication Technology Co. Ltd., Nanjing 210003, China)

Abstract: In order to achieve real-time monitoring of container clouds, host devices, and business systems, a cloud platform monitoring and alarm system based on Kafka and Kubernetes is designed. Docker containers are managed through Kubernetes, and Kafka receives device operation information from different hosts in different regions. The business system is monitored through probes. By setting the alarm association rules, redundant alarms are reduced, alarm fault detection capabilities are enhanced, and alarm accuracy is improved.

Key words: Kubernetes; Kafka; real time monitoring; associated alert rules; fault detect

随着云计算技术的高速发展, 新兴的虚拟化技术 Docker 容器凭借自身松耦合、分布式等优点,迅速被各大企业接受,越来越多的公司开始部署容器云并将之应用于实际生产中. 在容器云环境中,存在着大量的容器,因而需要一个高效便捷的集群管理方案[1]. 在众多容器编排系统中, Kubernetes 凭借其部署难可移动、可扩展、可修复等诸多优点, Kubernetes 也能让开发者斩断联系着实体机或虚拟机的“锁链”, 从以主机为中心的架构跃至以容器为中心的架构, 最终提供给开发者诸多内在的优势和便利[2]. 因此成为各大公司部署开发容器云的首选.

在现有的业务系统中, 主机上运行的各种业务程序承载了业务系统的采集、传输和展现, 因此主机的安全防护尤其重要, 一旦发生安全事故, 必然会大范围的影响业务系统的稳定运行. 基于以上两点, 本文设计了一种基于 Kafka 和 Kubernetes 的云平台实时监控系统, 通过对容器和主机进行全面监视, 可尽早识别主机的安全风险, 当出现数据异常时, 能够及时发现、及时

① 基金项目: 南京南瑞信息通信科技有限公司科技项目 (5246DR200014)
Foundation item: Science and Technology Project of Nanjing NARI Information and Communication Technology Co. Ltd. (5246DR200014)
收稿时间: 2020-02-13; 修改时间: 2020-03-17; 采用时间: 2020-04-03; csa 在线出版时间: 2020-07-29

处理, 从而达到保障操主机业务系统稳定的运行, 全面提升监控系统安全防护能力.

1 相关技术

1.1 Kubernetes 技术

Kubernetes (以下简称 K8s) 是 Google 开源的一款容器编排工具, 它是诞生在 Google 内部运行很多年的博格系统之上的产物, 因此其成熟度从其诞生初期就广泛受到业界的关注, 并且迅速成为编排工具市场的主流, 它的主要作用是对 Docker 容器做编排工作[3]. K8s 通过控制器实现容器状态监控、容器自动重建, 以及当容器处理能力不足时, 根据用户定义的扩展规则进行自动扩展等功能.

1.2 Kafka 技术

Kafka 是一个基于分布式的消息发布-订阅系统, Kafka 在主题中保存消息的信息. 生产者向主题写入数据, 消费者从主题读取数据, 从而实现数据传输. 同时主题也是可以在多个节点上被分区和覆盖的[4].

Kafka 可以以分布式集群的方式存在, 也可以单台模式存在其内部数据传输, 通过 TCP 协议实现 Kafka 传输的消息内容保存在操作系统内核的页面缓存中.具体的传输流程分为以下 5 个步骤:

(1) 生产者创建通信的“主题”并通知 Kafka;

(2) 消费者从 Kafka 上订阅其待接收的主题;

(3) 生产者将其待发送的消息发布到 Kafka 上;

(4) Kafka 将该主题的消息发送到订阅该主题的消费者;

(5) 消费者从 Kafka 接收其订阅主题的消息.

通过客户端访问数据库总线的方式, 如图 1 所示.

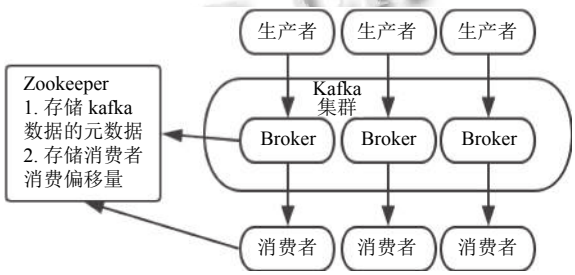

图 1 通过客户端访问 Kafka 集群示意图

图 2 描述了通过客户端访问 Kafka 的程序代码流程.

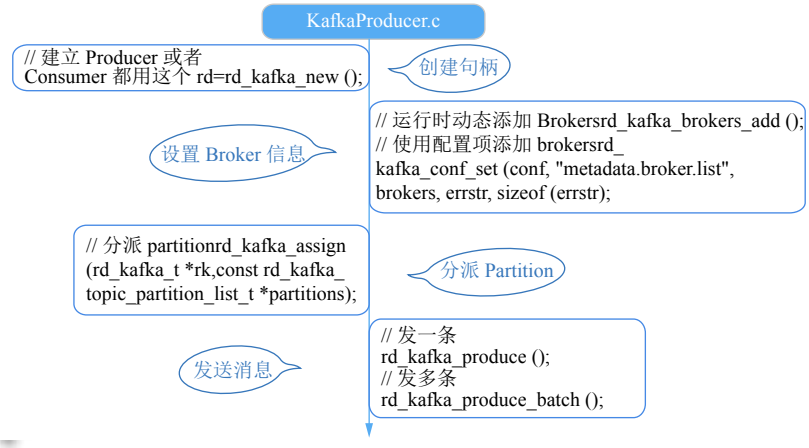

图 2 客户端访问消息总线程序代码流程示意图

2 系统架构设计

云平台监控系统采用分层架构设计实现,运行时从业务系统和各类基础设施中采集数据, 历经数据存储, 分析处理等多个功能层次的交互, 依次进行监控和告警数据采集、分析、上报、展示给终端用户等多个阶段, 形成如图 3 所示的分层结构.

云平台监控系统系统自下而上可分成数据采集层、数据存储层、数据分析层、展现层共 4 层. 第 1 层数据采集层, 负责接收业务系统监控的数据, 节点和容器的监控数据, 告警数据, 这些数据传输均采用 JSON 格式, 监控系统接收并进行相应的解析; 第 2 层为数据存储层, 负责将基础设施监控数据, 系统监控数据通过 Redis 缓存[5] 并转发存储到非关系型数据库 Elasticsearch 上, 告警数据写入本地的达梦数据库中;

第 3 层为数据分析层, 负责分析监控数据, 统计分析业务系统运行情况, 对监控数据进行分析, 生成告警信息; 第 4 层为数据展现层, 包括监控概览, 基础设施运行状况展示, 业务系统运行状况展示, 告警信息的展示与处理.

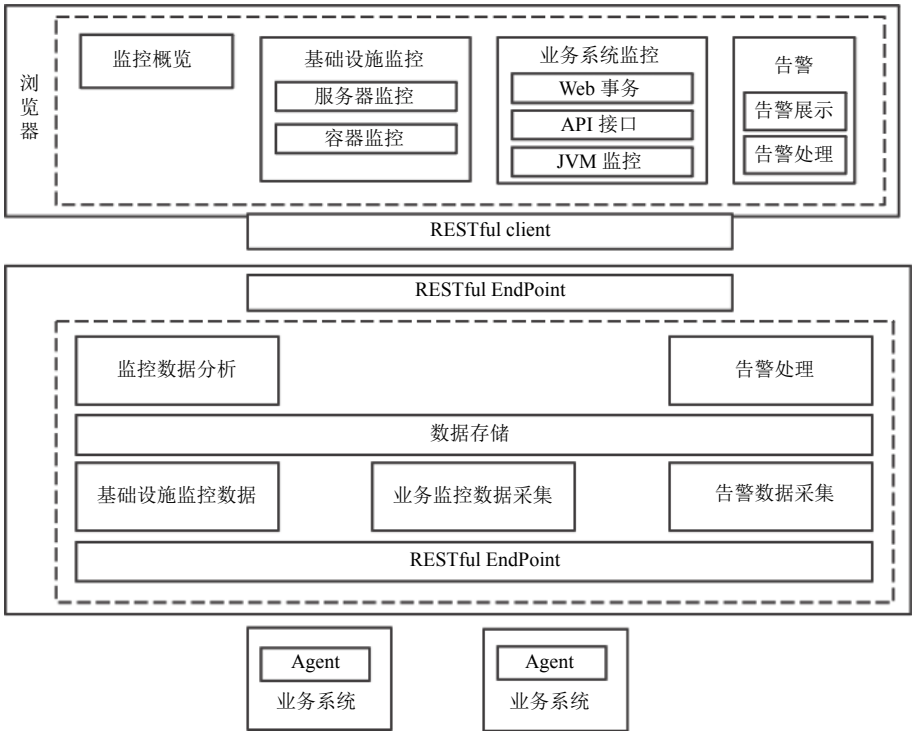

图 3 云平台业务系统架构图

3 系统设计与实现

云平台监控告警系统主要分为以下 3 个主要的模块: 基础设施监控模块, 业务系统监控模块, 告警模块.

基础设施监控模块: 对主机设备、服务器和 Docker 容器的 CPU, 内存, 文件使用率等数据进行监控, 并且对监控数据进行存储、统计和展示.

业务系统监控模块: 对业务系统的 Web 请求, SQL 执行, JVM 等数据进行监控、存储、分析和展示.

告警模块: 包括自定义告警规则, 主动分析监控数据生成告警信息, 接收由业务系统上报的告警信息, 告警通知, 告警处理和告警展示等功能.

3.1 基础设施监控

基础设施监控包括容器监控和主机设备监控, 本文采用 Kubernetes 来管理容器, 采用 cAdvisor 来采集服务器和容器的监控运行数据, cAdvisor 是谷歌开发的容器监控工具, cAdvisor 用于采集一台机器上运行的所有容器的信息, 对节点机器上的资源及容器运行状态信息进行采集, 包括 CPU 使用情况、内存使用情况、网络吞吐量和文件系统使用情况.

主机信息采集方式为, 由操作系统 (凝思、麒麟) 主动进行信息采集, 通过 Kafka 上报到监管平台. 信息采集分为周期上报 (运行信息) 和触发上报 2 种上报方式, 以达到对主机运行状态进行全面监视的目的.

表 1 为主机的静态配置信息, 操作系统主动采集如下信息 (发送一次即可), 通过 Kafka 发送至采集服务器, 最后在系统界面进行汇总展现.

表 1 主机部分配置信息采集表

监视项	具体内容
CPU配置信息	CPU数量(单位: 核) CPU主频(单位: GHz) CPU缓存(单位: MB/核)
内存配置信息	物理内存数(单位: GB) 虚拟内存数(单位: GB)
网卡信息	网卡数量名称 网卡速率类型
操作系统信息	操作系统名称 版本号 内核版本号

表 1 中内容的发送时机为操作系统采集模块启动时发送, 以后周期性发送 (每 24 小时发送一次), 通信

方式为 Kafka, 消息格式为 syslog.

除了表 1 中的配置信息、系统还会对主机的动态运行信息进行采集, 包括僵尸进程数量、未释放的 TCP 连接数、内存和硬盘的使用状态、网络端口监情况、网卡状态和主板温度状态等信息.

操作系统对监视内容使用 Kafka 消息总线通过订阅、发布不同的主题, 实现操作系统与平台的双向交互通信. 表 2 为传输内容说明表.

表 2 传输内容说明表

信息类型	报文格式	操作系统发布主题	操作系统订阅主题
登录操作信息	自定义格式	“pf_oper”	“sys_oper”
配置信息	syslog	“pf_monitor”	
状态信息	syslog	“pf_monitor”	
告警信息	syslog	“pf_warn”	

表 2 中, syslog 格式信息定义如下:

<告警级别><空格>告警时间<空格> IP#主机名<空格>设备类型<空格>内容描述. 例如: <0> 2006-03-12 20:12:23 192.168.20.22#scada SVR 0 System EXCEPTION

告警级别依照采集信息对监控系统安全的影响, 分为紧急 (0)、重要 (1)、普通 (2)、监视 (3) 4 个级别. 告警发生日期和时间格式 YYYY-MM-DD HH:MM:SS, YYYY 表示年份, MM 为月份, DD 是日期. 24 小时制, 有效值为 (00–23), MM 和 SS 的值的范围为 (00–59). 月、日、时、分、秒各 2 个字符, 小于 10 时十位应补 0.

告警发生日期和时间格式 YYYY-MM-DD HH:MM:SS, YYYY 表示年份, MM 为月份, DD 是日期. 24 小时制, 有效值为 (00–23), MM 和 SS 的值的范围为 (00–59). 月、日、时、分、秒各 2 个字符, 小于 10 时十位应补 0.

设备名称为标识产生告警事件的主机 IP 地址#主机名. 设备类型为一个用来描述告警源的不超过 32 个字符的字符集.

基础设施监控的整体流程为:

第 1 步. 基础设施监控模块连接 Kubenetes 管理机, 获取当前服务器节点信息, 应用信息, 服务信息, 接收主机设备通过 Kafka 传来的报文信息, 解析成需要的格式并将其记录入库.

第 2 步. 各个服务器中安装 cAdvisor 插件, 该插件启动会提供相关 CPU, 内存数据获取接口.

第 3 步. 基础设施监控模块逐个连接服务器, 从 cAdvisor 插件中获取 CPU, 内存等监控数据, 并存储入库.

第 4 步. 基础设施模块逐个连接所知的 Docker 容器, 从 cAdvisor 插件中获取 Docker 容器的 CPU, 内存等监控数据.

第 5 步. 基础设施模块获取主机设备通过 Kafka 上报的报文信息, 进行消息解析并存储入库.

第 6 步. 基础模块展示服务器和 Docker 容器的实时 CPU, 内存等运行数据, 支持查看 Docker 的日志, 部署等情况.

3.2 业务系统监控

业务系统监控依赖于监控 Agent 监控业务系统并上报监控信息. 因为业务系统和云平台监控系统是两个相互独立的系统, 因此在设计中应当尽量减少业务系统对监控系统的感知, 保证监控系统不会对业务系统的稳定、性能、安全造成影响, 监控系统在保障系统安全稳定运行的同时能够及时发现业务系统中的问题. 业务系统监控的整体流程如下:

第 1 步. 业务系统集成监控 Agent 包, 配置监控系统访问地址, 监控项等信息.

第 2 步. 监控 Agent 会根据配置, 对 HTTP, Servle, RESTful, Hessian 等 Web 请求进行监控, 并统计其请求次数, 响应时间, 异常等信息, 定期向云平台监控系统发送监控信息.

第 3 步. 云平台监控系统会接收各个节点发送的监控信息, 进行持久化存储.

第 4 步. 云平台监控系统统计和分析某业务系统各个节点的监控数据, 得出当前系统的总体运行数据.

第 5 步. 云平台监控系统支持展示系统总体的 Web 请求次数, 最大响应时间, 平均响应时间,异常次数等数据, 支持查询某时间段的运行情况, 支持查询 SQL 执行的次数, 平均执行时间等.

第 6 步. 云平台监控系统支持单节点的 Web 请求, SQL 执行, JVM 运行情况等展示, 支持单次请求服务链的跟踪, 该请求的 SQL 调用展示及异常堆栈信息展示.

3.3 告警模块

首先定义几个名词的概念:

(1) 事件告警. 本文中包括容器、节点、设备等 3 种告警. 包含发生时间、告警级别、告警对象、告警描述、告警类型、告警采样值及处理状态等属性.

(2) 告警采样值. 采样值触发告警时的容器状态信息数据, 包括 CPU 使用率、内存使用率、磁盘使用率等.

(3) 告警关联规则. 由多个告警指标抽象出的告警

规则, 由用户自主设定针对容器、节点或者设备的告警规则, 如对节点 node1 设定规则“CPU 使用率 6 小时内只要有一次大于 95%”,这项规则中包含 4 种判别条件, CPU 使用率、6 小时内、只要有一次、大于 95%, 通过监控节点 node1 的状态信息数据、只要有满足这 4 个条件的数据, 即会触发告警.

云平台告警通过@Scheduled(cron = "40 0/1 * * *?") 注解启动一个定时任务定时扫描告警规则, @Scheduled 由 Spring 定义, 用于将方法设置为调度任务. cron 参数接收一个 cron 表达式, cron 表达式是一个字符串, “0 0/1 * * *?”即代表从每小时的第 0 分 0 秒开始, 每分钟触发一次.

通过对告警规则的定期扫描, 查询 Elasticsearch 中的数据判断是否有满足告警规则的数据, 如果有的话,查询当前是否已经告警, 如果没有告警, 则把告警信息写入数据库中, 并修改状态为告警. 图 4 为告警分析流程图.

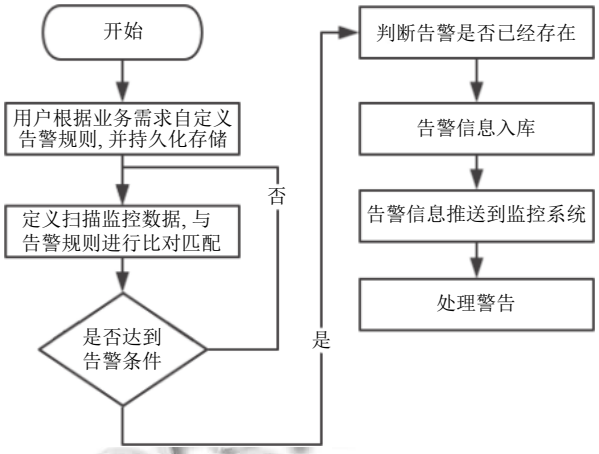

图 4 告警分析流程图

4 实验测试

4.1 系统平台部署

系统平台主要分为数据采集层, 数据存储层和 Web 展示层. 系统的开发环境部署如下, 数据采集层由部署在同一个局域网内的 Kubernetes 集群, 一台主机设备, 一台部署了业务系统的服务器和一台 Kafka 服务器组成, 数据存储层由 Elasticsearch 和 Oracle 数据库组成, Web 展示层由一台部署了云平台监控系统的服务器组成.

4.2 实验结果分析

图 5 左上为资源监控概况, 分别为 K8s 管理的节点、应用、容器已经设备的个数, 点开后有相应的详情, 如果有告警, 会在右下方亮红灯并显示告警个数, 可点开查看. 图 5 上中部分为实时 K8s 事件信息, 图 5 右上为实时告警信息, 点开后可看到详情及处理流程. 图 5 下半部分依次为 15 分钟内监控的 Web 事务访问最慢的 top5 请求、最慢的应用以及最慢的 SQL 统计, 点开后可看到详情. 图 6 为容器、设备、服务的运行状态信息.

图 7 和图 8 分别为告警规则管理界面和告警处理界面, 实现对告警规则的增删查改, 支持节点告警、容器告警和设备告警. 告警处理界面可以查看触发告警的详细数据, 以及对告警进行处理.

图 5 监控主界面

(a) 容器监控

(b) 业务系统监控

(c) 设备监控

图 6 监控系统监控详情

图 7 告警规则管理界面

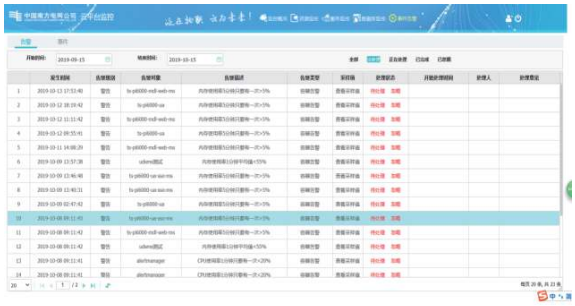

图 8 告警处理界面

5 结论与展望

本文设计了一套基于 Kafka 和 Kubernetes 的云平台监控告警系统, 为用户管理容器、主机设备和业务系统提供了支持, 可以实时监控容器、主机设备和业务系统的运行状态, 一旦出现故障, 能够迅速准确定位到故障根源, 保障了主机设备的平稳运行. 为了能够进一步保障系统的安全状态, 下一步仍需要对告警触发流程进行优化, 并且通过多种安全机制和技术手段保证系统安全稳定运行.

参考文献

1 杜军. 基于 Kubernetes 的云端资源调度器改进 [硕士学位论文]. 杭州: 浙江大学, 2016.

2 陈金光. 基于阿里云的 Kubernetes 容器云平台的设计与实现 [硕士学位论文]. 杭州: 浙江大学, 2018.

3 杨茂, 陈莉君. 基于 Kubernetes 的容器自动伸缩技术的研究. 计算机与数字工程, 2019, 47(9): 2217–2220, 2232. [doi: 10.3969/j.issn.1672-9722.2019.09.022]

4 王岩, 王纯. 一种基于 Kafka 的可靠的 Consumer 的设计方案. 软件, 2016, 37(1): 61–66. [doi: 10.3969/j.issn.1003-6970.2016.01.015]

5 李轲. 原生 redis 集群的优化与实现 [硕士学位论文]. 武汉: 华中科技大学, 2017.