计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(10):139−143 [doi: 10.15888/j.cnki.csa.006006]

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

# 柔性微服务监控框架①

刘一田, 刘士进, 郭　伟, 何　翔

(南瑞集团公司 (国网电力科学研究院), 南京 210003)

**摘　要**: 微服务细化了服务的粒度, 针对微服务的监控是微服务管理需要面对的核心问题, 以可扩展全链路应用服务监控为目标, 在已有服务管理层之上设计实现了一种柔性微服务监控框架, 用于监控微服务状态和不断变化的服务负载, 与已有的分布式监控架构相比, 这种监控框架灵活度更高, 对服务变化的感知能力更强, 使用 Raft 算法增强了数据一致性, 避免了单点故障的情况, 经试验分析, 代价更小, 在实际项目中验证了框架及方法的有效性.

**关键词**: 柔性; 微服务监控; 智能分析告警

引用格式: 刘一田,刘士进,郭伟,何翔.柔性微服务监控框架.计算机系统应用,2017,26(10):139–143. http://www.c-s-a.org.cn/1003-3254/6006.html

## Flexible Microservice Monitoring Framework

LIU Yi-Tian, LIU Shi-Jin, GUO Wei, HE  Xiang

(NARI Group Corporation(State Grid Electric Power Research Institute), Nanjing 210003, China)

**Abstract**: Microservices refine the granularity of services, and the monitoring of flexible microservices is the core of the microservices management. With the aim of extending the full-link application service monitoring, a flexible microservices monitoring framework is designed and implemented on the existing service management layer to monitor the microservices status and the changing service load. Compared with the existing distributed monitoring architecture, the framework is more flexible, more aware of service changes. The Raft algorithm is used to enhance data consistency and avoid single point failure. The experimental analysis shows that the cost is smaller, and the validity of the framework and the method is verified in the actual project.

**Key words**: flexible; microservices monitoring; intelligent analysis and alarm

## 1　引言

微服务的特点决定了应用功能模块的部署是分布式的, 以往在单体架构应用环境下, 所有的业务都在同一个服务器上, 服务器出现错误和异常时可以快速定位和处理问题, 但是在微服务的架构下, 大部分功能模块都是单独部署运行的, 彼此通过预先编排的无状态服务交互, 前后台业务流会经过多个微服务的处理和传递, 因此, 如何实时采集系统及应用服务的访问及异常信息, 根据业务流的错误和异常信息统计分析以辅助快速定位问题并告警, 灵活可扩展的配置告警, 以及能及时跟踪业务流的处理顺序和结果等技术是微服务监控的关键所在.

以 Zabbix 为代表的传统分布式监控技术更倾向于提供系统级的监控点和指标[1], 不能快速定位到系统问题的根本原因. Google 公司研发的 Dapper[2]分布式跟踪系统为服务器上每一次发送和接收动作来收集跟踪标识符和时间戳事件, 每次 RPC 服务调用被称为一个 Span, 用户的 URL 请求是根 Span A, 如果根 Span 调用了 RPC 服务 B 和 C, 则 B、C 为 A 的子 Span, 这样整个请求就被建模为由 A 为根的 Span 树, 被称为一个 Trace. 每个 Span 由 trace_id, span_id, parent_id 组成, 通过 parentId 和 spanId 就可以有序地把所有的关

① 基金项目: 国网电力科学研究院科技项目 (524606160150)
收稿时间: 2017-01-19; 采用时间: 2017-02-20

系串联起来, 达到记录业务流的作用, 从而实现端到端的追踪能力, 通过对 Span 计时, 可以统计性能数据. 但是 Dapper 只提供调用时间的追踪, 而不提供如错误追踪, 业务监控等功能, 无法较好满足业务系统日常业务应用监控的需求. 淘宝鹰眼是基于网络调用日志的分布式跟踪系统, 它可以分析网络请求在各个分布式系统之间的调用情况, 从而得到处理请求的调用链上的入口 URL、应用服务的调用关系, 从而找到请求处理瓶颈, 定位错误异常的根源位置. 同时, 业务方可以在调用链上设置业务埋点日志, 使各个系统的网络调用与实际业务内容得到关联. 需要一定的代码植入, 且扩展性略差.

本文设计的柔性微服务监控框架结合上述框架各自的典型优势技术, 分别从各方面进行了改进和提升. 在数据采集上报组件技术方面, 通过应用服务器中间件代理组件, 通过业务无侵入的方式提升了采集的定制化能力和可靠性; 在采集数据转换和合并方面, 采用 Raft[3]一致性算法, 将数据处理后分发到数据存储和告警服务, 确保数据的一致性和准确性; 在告警组件方面, 基于配置下发的告警处理策略及可复用的告警策略模板, 提高告警的定制化处理能力; 在统计分析组件方面, 基于聚类分析算法, 对实时和历史数据进行统计分析, 将分析后数据发往消息总线对应的订阅主题, 进行智能告警判定裁决, 提高告警准确度; 在监控展示方面, 通过配置可视化定制告警参数, 支持用户自定义监控所需的服务指标、JVM 等指标, 提高监控的自主性.

## 2 监控框架设计

本文设计的柔性微服务监控框架结构如图 1 所示. 其主要具备如下特征.

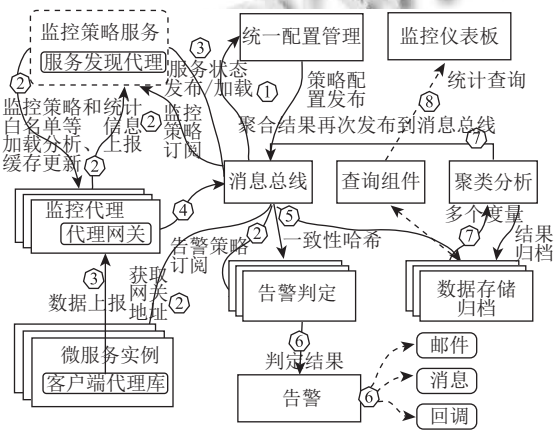


图 1 柔性微服务监控框架结构

(1) 架构设计高可用. 整个系统无核心单点, 易运维, 易部署.

(2) 低侵入性. 尽可能少侵入或者无侵入其他业务系统, 对于使用方透明, 减少开发人员的负担.

(3) 告警策略灵活和自发现. 支持策略模板、模板继承和覆盖、多种告警方式、支持回调动作, 可以随时决定所收集数据的范围和粒度.

(4) 支持最大告警次数、告警级别设置、告警恢复通知、告警暂停、不同时段不同阈值、支持维护周期, 支持告警合并.

(5) 时效性高、扩展性好. 从数据的收集和产生, 到数据计算处理和最终展现都尽可能快; 生产环境每秒 30 万次数据收集、告警、存储、绘图, 可持续水平扩展.

(6) 决策支持. 提炼分析数据从 DevOps 角度并提供决策支持服务.

(7) 自助分析仪表板. 多维度的数据展示, 用户自定义仪表板等功能, 提升更细粒度的用户自助分析应用体验.

柔性微服务监控框架的执行流程分为八个步骤, 分别介绍如下.

(1) 统一配置管理中心统一维护监控和告警策略配置, 配置的变更自动发布到消息总线的约定主题上, 配置管理支持集群、负载均衡及高可用.

(2) 监控策略服务作为监控代理的辅助工具, 从消息总线订阅监控策略配置, 监控代理根据监控配置收集微服务状态信息; 微服务实例从消息总线指定主题订阅监控代理地址, 作为数据上报的地址; 告警判定组件从消息总线订阅告警策略配置, 作为告警的判定依据; 监控策略包括监控告警阈值、特殊定制的采集项, IP 白名单等, 监控代理将特殊采集项信息上报给监控策略服务.

(3) 监控策略服务根据统计信息发布不可用的微服务实例状态到消息总线指定主题, 统一配置管理中心订阅相应主题后及时更新监控状态. 客户端代理库作为微服务中间件代理, 应用通过代理将服务埋点、调试、异常等信息采集到监控代理服务实例.

(4) 监控代理网关依据监控策略对监控信息进行筛选, 将筛选后的数据发布到消息总线指定告警主题.

(5) 告警判断组件和数据存储和归档组件通过一致性哈希数据转换合并工具将发布的消息通知订阅到本地进行存储.

(6) 告警判定组件从消息总线订阅告警策略和微服务统计信息, 并依据告警策略和服务统计信息等参数, 初步告警加权判定, 根据判断结果告警并以邮件等方式处理.

(7) 监控聚类分析组件定时从数据存储中获取历史数据, 进行聚类学习和训练, 分析出服务调用的调用链等信息, 并将分析结果提交给消息总线指定监控主题分发. 重复告警判定流程.

(8) 监控展示组件支持指定的环境、应用和服务指标配置定制, 根据定制参数实时展示监控图表, 实现良好的自助分析效果.

## 3 监控框架实现

柔性微服务监控框架采用 JSON 格式的策略模板支持快速配置定义监控和告警策略; 微服务实例的客户端代理动态获取代理网关地址, 将日志及埋点信息上报给监控代理组件; 监控代理组件根据监控和告警策略, 筛选过滤所需的监控统计信息并发布到消息总线指定主题; 告警判定组件从消息总线订阅告警策略及数据统计信息, 形成初步判断结果并发布到告警通知事件; 数据存储归档组件订阅的上报数据进行聚类统计分析, 聚合结果存储归档并再次发布到消息总线作为告警判断组件的输入, 同时用户可通过监控仪表板调用查询组件监控及自助分析统计指标. 根据上述框架的设计, 需要实现五个主要关键技术组件: (1) 高可用的统一配置管理中心. (2) 无侵入的微服务实例客户端代理. (3) 高可用及高性能的消息总线. (4) 灵活可复用的监控及告警策略模板. (5) 聚类分析的统计算法.

(1) 高可用统一配置中心

框架采用动态路由的高可用代理及分布式一致性 Raft[3]算法作为统一配置中心集群环境下负载均衡、统一配置的一致性及高可用, 实现配置管理集群实例在节点故障重启时对请求透明化. 如图 2 所示.

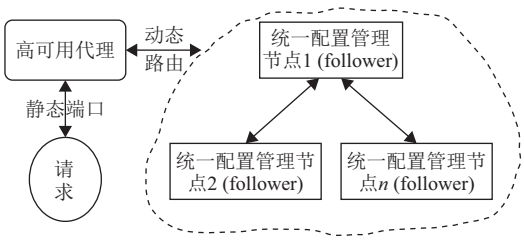


图 2 高可用统一配置中心

统一配置管理集群由高可用代理节点对外统一暴露静态固定端口, 而高可用代理本身监控配置管理节点的状态变更信息, 当配置管理节点集群中出现节点不可用、动态扩充配置管理节点等状况时, 高可用代理及时更新路由表.

配置信息的高可用技术基于 Raft 算法, 集群中的节点分为 leader、follower、candidate 三种角色, 由 leader 响应客户端请求并确保响应结果的一致性. 统一配置管理节点集群首先选举一个 leader 节点, 由 leader 节点对外提供服务, 当配置更新时, leader 节点发出命令, 并在确保集群中多数节点都已完成命令操作后, 返回正确响应给请求对象. 如果一个 follower 在选举超时的时间周期里没有收到 leader 的信息, 就进入新的选举周期, 自身转成候选人 candidate 角色, 给自己投票, 发起选举, 并重写产生 leader 以继续提供高可用一致性服务.

(2) 无侵入的微服务实例客户端代理

系统实施监控的方式分为两类: 一类是内嵌式, 一类是伴随式. 对大型分布式系统来讲, 伴随式是一种更加合适的结构[4].

框架采用伴随式的中间件劫持技术实现微服务实例的客户端代理, 业务系统应用开发时只需关注业务功能实现. 中间件劫持就是将我们自己的代码行为植入到中间件的各种行为中. 实现劫持和监控主要依靠四种关键行为: 应用启动, 停止, 接收请求, 响应回复. 对 JavaEE 应用服务器的劫持核心是根据 classloader 的加载时机, 在 classloader 的树型层次上获得优先加载权, 从而可以改变这些行为. 如图 3 所示.

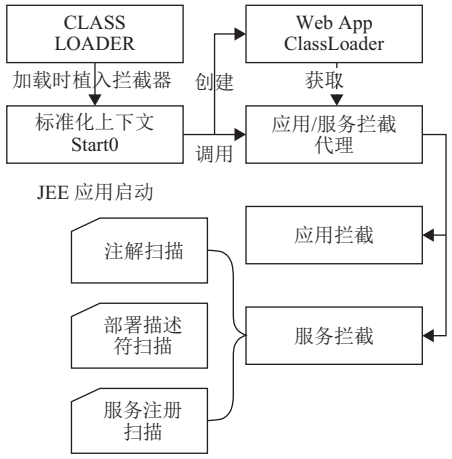


图 3 无侵入的微服务实例客户端代理

通过中间件劫持技术, 全自动的收集微应用实例、微服务实例及服务接口的信息. 这些信息包括应用唯一标识、服务名、服务实例的 URI、服务接口的 URI、服务接口的元数据 (类, 方法, 入参出参, 注解, 部署描述符) 等统计信息. 其中, 服务类型和方法通过 Java 的反射方式提取信息; Annotation 通过注解扫描工具提取具有相关注解的类, 然后通过注解 API 提取注解信息; 部署描述符通过 WebAppClassLoader 获取 web.xml, spring-config.xml, log4j.xml 等部署描述符文件路径, 然后使用文档对象模型解析提取关注的标签信息. 最后调用特定服务接口获取服务实例的内存、CPU 等信息, 并动态获取监控代理网关的地址, 将统计数据上报给代理网关.

(3) 灵活可复用的监控及告警策略模板

框架采用灵活可复用的监控及告警策略模板, 以 JSON 格式描述, 如表 1 所示.

表 1 监控策略模板描述

```
{ metric: load.1min,
endpoint: pi6000-host-n1,
tags: srv=abc, idc=aws-sgp, group=az1,
value: 1.5,
timestamp: ‘date +%s’,
counterType: GAUGE,
step: 60 }
```

主机域是一组采用相同监控策略机器的集群, 监控策略模板支持继承和策略覆盖, 模板和主机域绑定后, 主机下的机器会自动应用该模板的所有策略, 节点进出主机域时, 相关的模板会自动关联或者解除. 服务上下线不需要手动来变更监控, 从而在提高效率的同时降低了遗漏和误报警. 策略模板支持以表达式方式动态添加监控及告警策略, 如图 4 所示.


图 4 监控及告警策略模板定义

(4) 高可用及高性能的消息总线

为了实现在分布式环境下的即时消息通知, 并保障消息的高性能传输, 框架采用高性能分布式消息总线 Apache Kafka[5]组件实现消息的发布/订阅功能, 并使用多个节点组成集群以保障消息总线的高可用. 通信模式采用发布/订阅模式, 使消息按照特定的主题甚至内容进行分发, 应用程序可以根据主题接收到所需要的消息. 发布/订阅功能使得发送者和接收者之间的耦合关系变得更为松散, 发送者和接收者不必关心消息的发送和接收地址, 而只是根据消息的主题进行消息的收发.

(5) 聚类分析的统计算法

统计分析采集日志的主要目标包括分析服务调用全过程链、用户在导航或浏览网站时生成的点击流或单击路径、发生系统宕机之前的事件的日志、用户在系统关键操作顺序的事务记录、根据一段时间内用户的互动来预测服务取消或其他不良结果的记录.

框架算法综合了顺序分析、聚类分析方法和 Markov 链分析, 以识别数据分类及其顺序并进行动态信任决策[6]. 分析算法的特点之一是使用顺序数据, 此数据通常表示数据集状态之间的一系列事件或转换, 如图 5 所示.

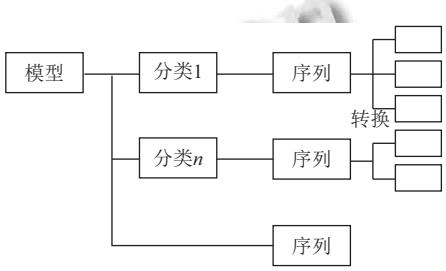


图 5 聚类分析的统计算法

## 4 案例分析与评估

为了对本文提出的柔性微服务监控框架进行验证, 在一个真实的环境中进行了案例的部署和研究. 案例环境由 16 台 8 核 8 G 内存的 PC 服务器 (HPDL 380 G4 378735-AA1) 组成, 其中数据库、微服务实例、监控代理节点、统一配置管理节点、消息总线节点、日志归档节点、告警节点比例为 1:8:1:2:2:1:1, 数据库和应用服务采用国家电网公司智能运检系统的测试数据库和经过拆分的微应用服务, 微服务实例以 Docker 容器形式运行. 通过 loadrunner 模拟并发 600 用户的微服

务访问, 通过监控策略和告警模板定制统计了服务响应时间超过 5 秒的 URL 及异常信息, 观察柔性微服务监控框架的效率和稳定性, 评估客户端代理库对应用的性能影响, 试验结果如表 2 所示.

表 2 客户端代理对应用性能的影响

| 处理器数 | 处理请求数 | CPU占用(%) |
|---|---|---|
| 4 | 5000/sec | 0.015 |
| 8 | 15000/sec | 0.079 |
| 16 | 50000/sec | 0.035 |

验证结果表明, 柔性微服务监控框架在大并发吞吐量时保证了高可用和高效, 但也发现了一些问题, 针对小规模的应用现有框架的部署架构略显复杂, 客户端代理库会在生产模式下会产生微小的性能损耗, 尚有提升空间. 目前, 柔性微服务监控框架已在国网智能运检系统中实现并应用, 取得了较好的应用效果.

## 5 结束语

本文研究了传统分布式系统监控技术及基于服务调用链的分布式系统追踪技术, 在此基础上, 设计了柔性微服务监控框架, 给出了高可用配置中心、无侵入的微服务实例客户端代理、灵活可复用的监控告警模板等创新点, 阐述了该框架的架构设计及关键实现技术. 最后, 以国网智能运检系统的微服务应用案例为背景, 给出了柔性微服务监控框架的应用验证评估, 验证结果、效率评估及生产运行实践表明, 该框架提升了分布式系统中微服务监控管理的灵活度、效率和问题定位准确度, 提高了电网应用信息系统的服务水平. 后续将针对遗留问题持续改进优化该框架.

## 参考文献

1 Zabbix. Enterprise-class monitoring system. http://www.zabbix.com/functionality.

2 Sigelman BH, Barroso LA, Burrows M, *et al*. Dapper, a large-scale distributed systems tracing infrastructure. Google Technical Report dapper-2010-l, 2010.

3 Ongaro D, Ousterhout J. In search of an understandable consensus algorithm. Proc. of the 2014 USENIX Annual Technical Conference. Philadelphia, PA, USA. 2014. 305–319.

4 刘东红, 郭长国, 王怀民, 等. 监控使能的分布式软件系统构造方法. 软件学报, 2011, 22(11): 2610–2624.

5 Kafka Apache. A distributed streaming platform. http://kafka.apache.org/intro.

6 李小勇, 桂小林, 毛倩, 等. 基于行为监控的自适应动态信任度测模型. 计算机学报, 2009, 32(4): 664–674.