文章编号:2096-1472(2023)09-0048-04
DOI:10.19644/j.cnki.issn2096-1472.2023.009.009

基于 OpenTelemetry+Jaeger 的分布式系统调用链路监控方案

张爱华[1]，白金峰[2]

[1.大连东软信息学院网络工程系，辽宁 大连 116023；
2.欧普杰科技(大连)有限公司，辽宁 大连 116023]
✉ zhangaihua@neusoft.edu.cn; baijinfeng1202@126.com

摘 要:在分布式系统中,由于各个功能模块的通信网络错综复杂,因此对于单一请求的调用链路监控与分析就显得尤为重要。文章在详细调研大部分分布式系统的通用结构和调用链路特征的基础上,设计了一套通用的分布式系统调用链路监控方案。该系统基于 OpenTelemetry(开源可观察性框架)对服务进行埋点采样及数据整体搜集,使用 Jaeger(分布式跟踪系统)对数据进行整理分析并进行可视化展示。使用该方案对分布式系统进行监控,可以快速发现系统中存在的链路问题并监控服务健康程度,使分布式系统的运行更加稳定,能给用户带来更好的体验。

关键词:分布式系统;链路监控;OpenTelemetry;Jaeger
中图分类号:TP391 **文献标志码**:A

Design of Distributed System Call Link Monitoring Based on OpenTelemetry and Jaeger

ZHANG Aihua[1], BAI Jinfeng[2]

[1.*Department of Network Engineering, Dalian Neusoft University of Information, Dalian* 116023, *China*;
2. *OpenJaw Technology (Dalian) Co., Ltd., Dalian* 116023, *China*]
✉ zhangaihua@neusoft.edu.cn; baijinfeng1202@126.com

Abstract: In distributed systems, monitoring and analyzing the call link of a single request is particularly important due to the complex communication networks of various functional modules. After a detailed investigation of the general structure and call link characteristics of most distributed systems, the paper proposes to design a universal distributed system call link monitoring scheme. OpenTelemetry (an open source observability framework) is used for buried point sampling and overall data collection of services, and Jaeger (a distributed tracing system) is used to organize, analyze, and visualize the data. By using this scheme to monitor distributed systems, it is possible to quickly identify link issues in the system and monitor service health, making the operation of the distributed system more stable and providing users with a better experience.

Key words: distributed system; link monitoring; OpenTelemetry; Jaeger

0 引言(Introduction)

随着互联网移动应用的兴起,一种新的计算机应用系统服务架构-分布式架构应运而生。但是,随着分布式架构方案的使用越来越多,该架构的缺点慢慢显现出来,首当其冲的就是调用链路的问题[1]。随着分布式系统中服务数量的增加,其相互调用的关系网络也变得错综复杂,无论是系统的开发人员,还是维护人员,都因为无法追踪一个请求的调用链路而犯难。

本文针对这一情况,提供了一种通用的调用链路监控方案,细化每一个服务接口的粒度。在本方案中,既可以通过可视化界面的方式查询应用接口的调用流程、错误率、负载强度,又可以通过图示的方式查看系统的调用结构。

通过使用本文中提供的监控方案,系统开发人员可以快速

收稿日期:2023-02-22

追踪系统中任意服务的调用步骤、执行的命令或者参数，方便快速排查问题；运维人员可以实时查看系统中每一个服务接口的负载压力与错误率，可以提前预判服务状态。

1　系统需求(System requirements)

分布式链路监控系统的主要用户为系统的开发人员和维护人员。图 1 为系统开发人员的需求。系统开发人员希望使用该系统查看应用的调用链路、接口耗时、调用步骤中的应用信息、参数、环境信息等，还需要查看系统的整体调用网络。

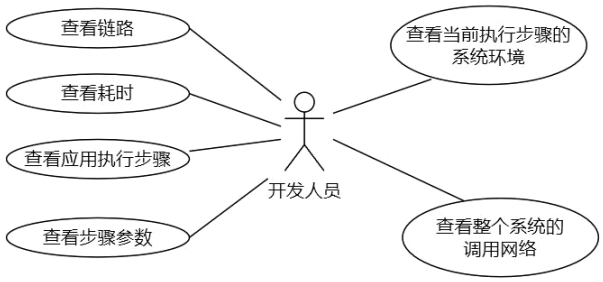

图 1　开发人员需求

Fig. 1 Developer requirements

如图 2 所示，系统维护人员希望能够查询系统的调用链路，每个接口的负载状况、错误率等信息。

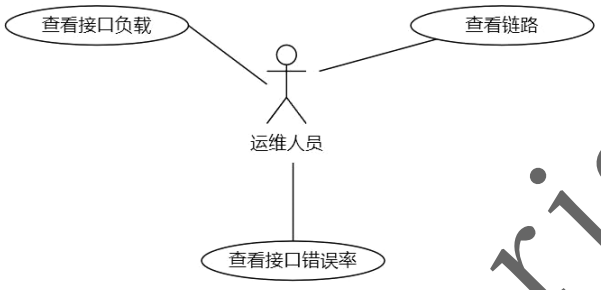

图 2　运维人员需求

Fig. 2 Operator requirements

2　系统设计(System design)

2.1　技术路线

系统服务端采用 B/S 架构，前端和后端基于 HTTP 协议进行通信，数据格式使用 JSON。前端使用的是 NodeJS＋React 技术，通过 NodeJS 查询 Server 端数据，然后结合 React 构造的模块化页面展示实现动态响应。

系统后端使用的是 Golang 语言，该语言自身的协程设计使其非常适合应用在处理高并发请求环境中，并且构建后的可执行产物体积小、启动方便。

服务埋点以 Java 应用为例，使用的是 Java 语言本身支持的 JavaAgent 方案，通过 JavaAgent 挂载到应用本身的 Java 虚拟机(Java Virtual Machine，JVM)中。该方案对应用本身没有任何的代码侵入，业务应用不需要任何改造即可轻松集成 Agent。

ElasticSearch 是一款分布式的搜索和分析引擎，该引擎可以使链路监控的数据搜集部分与展示部分解耦[1]。Agent 在应用层面搜集到的应用运行数据和日志等信息上报给 ElasticSearch 引擎，引擎对数据进行存储和索引，页面展示部分从 ElasticSearch 中读取并分析数据，然后展示到页面上。

OpenTelemetry Collector 是一款设备无关性的新兴采集器。在以往的分布式系统度量数据采集方案中使用的采集器一般与技术栈绑定，一旦选择了某个技术栈中的一个工具，则后续拓展必须使用该技术栈下的其他工具。例如，在搜集应用日志时，一旦选择了使用 filebeat，则后续的日志存储只能选择 ElasticSearch，日志展示与分析也只能选择 Kibana。OpenTelemetry Collector 的出现打破了这一局限性，它支持从多个输入端采集数据，然后通过自定义的 Processor 处理，处理后又可以从多个输出端输出数据，方便随时替换度量数据的搜集器和分析器。

Jaeger 是一款新型的链路分析工具，它包含 Agent、Analyzer、QueryUI 等组件。本文使用 Jaeger 的 Analyzer 和 QueryUI 组件，其中 Analyzer 用来分析链路数据和服务的网络结构，QueryUI 用来查询和展示分析后的链路和调用网络。

Prometheus 是一款存储应用度量数据的工具，它提供了一种基于时间轴的新型数据存储和管理方案，可以很好地记录某一指标随着时间变化而变化的值[2]；同时，它还提供一套类似于结构化查询语言(Structured Query Language，SQL)，通过使用其提供的查询语言可以方便地从各个维度查询度量数据。

2.2　系统架构

2.2.1　整体架构

系统的整体架构如图 3 所示，系统中存在多个业务应用(Business Application)，并且每一个业务应用都挂载了一个 Agent 用来收集应用的度量数据和运行时的状态数据[3]。Agent 收集数据后会发送给 OpenTelemetry Collector，Collector 对数据进行处理后，根据不同的数据类型分别发送给 Jaeger Collector 搜集器和 Prometheus 数据库。Jaeger Collector 对数据做筛选，加标签处理后存入 ElasticSearch 中等待分析和展示，ElasticSearch 中的数据会定时由 Spark Job 获取并计算调用网络，计算后的结果会再次存储至 ElasticSearch 中。Jaeger Query 服务负责根据用户的需求将数据从 ElasticSearch 和 Prometheus 中取出并通过用户界面(User Interface，UI)进行相应的展示。

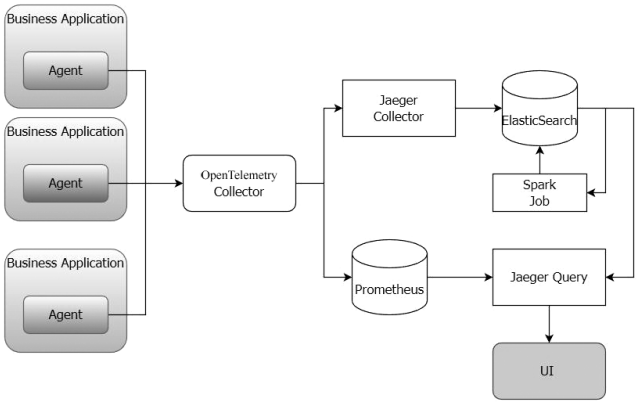

图 3　系统整体架构

Fig. 3 Overall structure of the system

2.2.2 高可用节点集群方案

为保证系统处于高可用状态(如图 4 所示),系统中的 OpenTelemetry Collector、Jaeger Collector 和 Jaeger Query 等组件都可以使用集群化部署,根据系统压力进行水平扩容[4]。

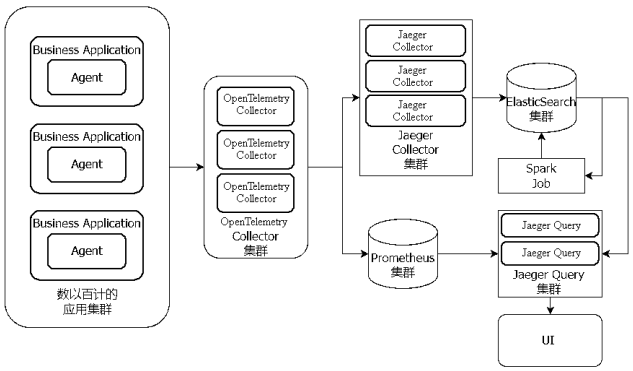

图 4 高可用节点集群

Fig. 4 High availability node cluster

同时,开源组件 ElasticSearch 和 Prometheus 也支持高可用的集群化部署,多节点通过访问单一存储系统实现数据一致性,开源组件高可用方案如图 5 所示[5]。

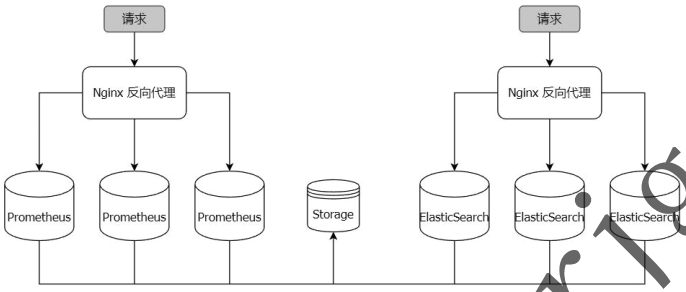

图 5 开源组件高可用方案

Fig. 5 High availability solution for open source components

3 系统实现(System implementation)

3.1 节点分配

如表 1 所示,搭建一个最小集群共需要 6 台机器,可以选择在虚拟机中创建这些机器并分配为固定 IP。其中,用来部署 ElasticSearch 的机器需要分配足够的内存(建议不小于 8 GB)和存储(不小于 50 GB);用来部署 Prometheus 的机器需要分配足够的存储(不小于 50 GB);用来部署 Collector 的两台机器需要分配足够的 CPU(不小于双核),以满足运算需要。

表 1 系统机器分配列表

Tab. 1 System machine allocation list

机器 IP	机器用途	备注
192.168.10.1	OpenTelemetry Collector	CPU 需要多分配
192.168.10.2	Jaeger Collector	CPU 需要多分配
192.168.10.3	Jaeger Query	无特殊要求
192.168.10.4	ElasticSearch	存储、内存多分配
192.168.10.5	Prometheus	存储多分配
192.168.10.6	UI & Spark Job	无特殊要求

3.2 应用的配置与部署

由于开源组件 ElasticSearch 和 Prometheus 的部署方案在官网或者网络上有很多相关资料,故在此不做赘述。OpenTelemetry Collector 的部署重点是配置文件的编写,需要配置 Collector 的输入监听、数据处理器和输出监听。输入监听使用的是 Jaeger 格式数据的监听方案,数据处理器使用 SpanMetrics 方案将度量数据与链路数据进行拆分,输出监听分别需要 Prometheus 和 Jaeger Collector。配置文件的具体内容如图 6 所示。

```
receivers:
  jaeger:
    protocols:
      grpc:
        endpoint: 192.168.10.1:14250 #OpenTelentry Collector 监听地址
exporters:
  jaeger:
    endpoint: 192.168.10.2:14250 #Jaeger Collector 服务监听地址
  prometheus:
    endpoint: 192.168.10.5: 9090 #Prometheus 服务监听地址
processors:
  batch:
  spanmetrics:  # 数据处理器
    metrics_exporter: prometheus # 处理器分离出来的度量数据输出地址
service:
  pipelines:
    traces: # 链路数据的输入输出与数据处理
      receivers: [jaeger]
      processors: [spanmetrics,batch]
      exporters: [jaeger]
  metrics: # 度量数据的输入与输出处理
    receivers: [otlp]
    exporters: [prometheus]
```

图 6 OpenTelemetry collector 配置文件

Fig. 6 Configuration file of OpenTelemetry collector

Jaeger Collector 可以使用官网上的二进制文件,然后通过命令行参数的方式直接启动该应用即可,主要的命令行参数包括指定存储类型、索引前缀名称等。将下载好的二进制文件放在/opt/app/jaeger/jaeger-collector 中,则可以使用图 7 中的命令启动 Jaeger Collector。

```
nohup sh /opt/app/jaeger/jaeger-collector \
--listen-address=192.168.10.2:14250 \
--collector.storage.type=elasticsearch \
--collector.otlp.enabled=true \
--es-arcive.index-prefix=demo \
--es-archive.server-urls=http://192.168.10.4:9200 \
--es.index-prefix=demo \
--es.server-urls=http://192.168.10.4:9200 \
 > ./jaeger-collector.log 2>&1 &
```

图 7 Jaeger 收集器命令行启动参数

Fig. 7 Startup parameters of Jaeger collector command line

Jaeger Query 和 Jaeger UI 的启动方式与 Jaeger Collector 的启动方案类似,这里不再赘述。

3.3 业务服务 Agent 配置

以 Java 应用为例,使用 Java Agent 的方案进行配置[6]。该方案需要准备收集数据的 Agent.jar 和应用相关的参数配置文件,应用配置文件内容详解如图 8 所示。

更新参数配置文件后,需要对业务服务的启动命令进行改造,将 agent 和 agent 使用的应用参数配置文件加入启动命令中,应用启动参数的修改如图 9 所示。

```
otel.sdk.disabled=false #是否禁用agent
otel.service.name=demo-service # 服务名称
otel.traces.exporter=jaeger # 链路数据格式
otel.exporter.jaeger.endpoint=http://192.168.10.1:14250/api/traces # 上报地址
otel.metrics.exporter=otlp # 元数据导出方式
otel.logs.exporter=otlp # 日志数据导出方式
otel.exporter.otlp.endpoint=http://192.168.10.1:4317
#OpenTelentry Collector 的地址
otel.exporter.otlp.metrics.endpoint=http://192.168.10.1:4317
#OpenTelentry Collector 的地址
otel.exporter.otlp.logs.endpoint=http://192.168.10.1:4317
#OpenTelentry Collector 的地址
```

图 8　Agent 应用相关参数配置

Fig. 8 Agent application related parameter configuration

```
# 旧的启动命令
java -Xms2048m -XmX8192M -Xss4m \
-XX:+PrintGC -XX:+PrintGCTimeStamps \
-Xloggc:/opt/applog/gc.log \
-XX:+HeapDumpOnOutOfMemoryError \
-XX:HeapDumpPath=/opt/applog/heapdump.hprof \
-Dfile.encoding=utf-8 \
-Duser.timezone=Asia/Shanghai \
-jar /opt/app/application.jar

# 新的启动命令
java -javaagent:/opt/app/agent/agent.jar \ #指定agent
-Dotel.javaagent.config=/agent-config.properties # agent 参数
-Xms2048m -XmX8192M -Xss4m \
-XX:+PrintGC -XX:+PrintGCTimeStamps \
-Xloggc:/opt/applog/gc.log \
-XX:+HeapDumpOnOutOfMemoryError \
-XX:HeapDumpPath=/opt/applog/heapdump.hprof \
-Dfile.encoding=utf-8 \
-Duser.timezone=Asia/Shanghai \
-jar /opt/app/application.jar
```

图 9　应用启动参数的修改

Fig. 9 Modification of application startup parameters

4　系统测试(System testing)

4.1　服务应用 Agent 测试

系统采用的是 Agent 挂载到业务应用上的启动方案，所以对 Agent 的测试可通过应用打印的启动日志进行查看[7]。在应用启动的过程中，可以通过观察应用的输出日志判断 Agent 是否成功挂载到应用上。经测试，Agent 已经成功挂载到应用上。

4.2　链路监控系统测试

对链路监控系统的测试，可以通过访问应用接口和提升应用接口压力的方案实现。通过访问业务应用的接口，然后在本系统中查看追踪到的链路即可确认链路上报与监控功能是否可用[8]。经测试，系统的链路监控功能可用，链路监控页面测试结果如图 10 所示。

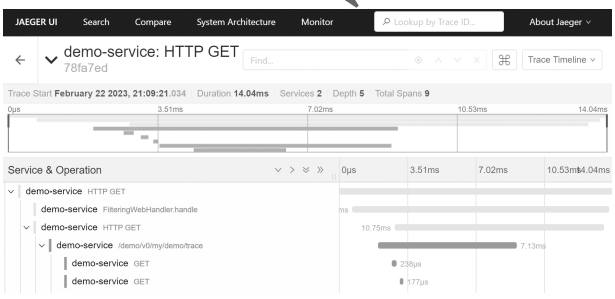

图 10　链路监控页面测试结果

Fig. 10 Test results link monitoring page

4.3　指标展示测试

对系统的指标展示的测试，可以通过给指定应用增加访问压力迫使指标数据出现波动的方案实现。经测试，系统指标展示功能可用。系统指标数据页面测试结果如图 11 所示。

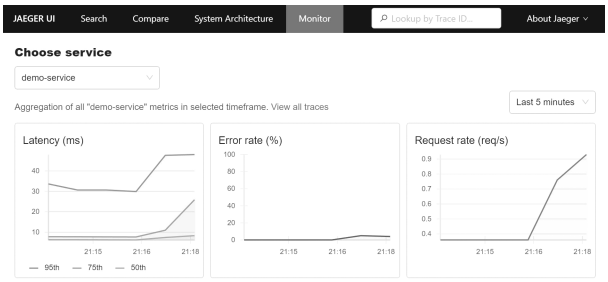

图 11　指标数据页面测试结果

Fig. 11 Testing results of indicator data page

4.4　其他测试

除了上述几项重要功能测试，本文还对系统的其他分支功能如链路步骤查询、链路标签查询、链路中某一步骤参数与结果分析、链路对比、服务网络分析等进行测试，测试结果均正常，各项功能均可用。

4.5　测试结果总结

系统共设计了功能性测试用例共 35 个，执行成功 35 个；设计了非功能性测试 12 个，成功执行了 12 个；设计了性能测试共 3 项，在单节点状态下，服务接口成功率达 99.97%，系统资源占用稳定，没有内存泄漏和 CPU 占用率过高的情况。

5　结论(Conclusion)

本文提出的链路监控方案与传统的链路监控方案相比，对业务应用的侵入几乎可以忽略不计，对业务应用的性能影响非常小。同时，提供了更全面、详细的各项链路指标，方便系统的开发和维护人员快速排查、定位问题。同时，系统的可拓展性非常强，既可以兼容现有比较主流的链路监控插件，也可以支持个性化的链路监控指标配置。

参考文献(References)

[1] 李传根. Elasticsearch 数据存储策略研究[D]. 重庆：重庆邮电大学，2019.

[2] 刘小磊，程伟华，章路进. 基于 Prometheus 的云计算资源全链路监控系统[J]. 电子设计工程，2023，31(2)：170-174.

[3] 张华兵，周英耀，徐磊，等. 基于微服务架构的电力信息系统全链路监控技术[J]. 沈阳工业大学学报，2022，44(4)：409-414.

[4] 张智强，郭龙，赵雷，等. 分布式软件开发与系统集成平台的研究[J]. 物联网技术，2020，10(6)：67-70，75.

[5] 夏刚. 数据中心基础设施高可用提升研究与实践[J]. 中国金融电脑，2022(5)：83-85.

[6] 李庆民. 基于 java 的软件 agent 开发环境的分析[J]. 数字技术与应用，2017(1)：189.

[7] KADA B，KHALID M，SHAIKH M S. Distributed cooperative control of autonomous multi-agent UAV systems using smooth control[J]. Journal of Systems Engineering and Electronics，2020，31(6)：1297-1307.

[8] POSHTKOHI A，GHAZNAVI-GHOUSHCHI M B. Implementing parallel and distributed systems[M]. Boca Raton：CRC Press，2022：16-22.

作者简介：

张爱华(1981-)，女，硕士，讲师。研究领域：DevOps，网络工程，软件开发。

白金峰(1997-)，男，本科，工程师。研究领域：Linux，分布式系统，DevOps。