서비스 메시 환경을 위한 설정 가능한 API 관측 가능성 및 원격 측정 시스템

차승빈[1], 남재현[2]
[1]단국대학교 컴퓨터공학과 학부생
[2]단국대학교 컴퓨터공학과 교수
[1]csb0710@dankook.ac.kr, [2]namjh@dankook.ac.kr

Configurable API Observability and Telemetry System for Service Mesh Environment

Seungbin Cha[1], Jaehyun Nam[2]
[1]Dept. of Computer Engineering, Dankook University (Undergraduate Student)
[2]Dept. of Computer Engineering, Dankook University (Professor)

요 약

최근 마이크로서비스 아키텍처가 널리 활용되면서, 분산 시스템의 규모가 점차 확장되고 시스템의 복잡성 역시 빠르게 증가하고 있다. 그리고, 이는 전체 시스템에 대한 가시성 저하시킬 뿐만 아니라 트러블 슈팅 역시 어렵게 만들었다. 결국, 이러한 시스템의 가시성을 확보하기 위한 관측 가능성의 중요성이 높아지고 있으며, OpenTelemetry와 Jaeger와 같은 도구들이 등장하게 되었다. 하지만, 이러한 도구들의 경우 수집 데이터의 형식이 고정 되어 있으며, 수집 범위 역시 제한적이다. 또한, 모니터링 과정에서 네트워크 트래픽과 디스크 I/O 등에서 추가적인 오버헤드를 발생시키는 문제점을 가지고 있다. 따라서, 본 연구에서는 대표적인 서비스 메시 환경인 Istio 환경을 기준으로 사용자가 원하는 형식과 범위로 마이크로서비스들과 관련된 로그 및 매트릭을 수집할 수 있는 경량화된 사이드카 기반 API 관측 가능성 및 원격 측정 시스템을 제안하고자 한다.

1. 서론

최근 컨테이너 환경이 널리 활용되면서, 기존의 시스템을 작은 단위로 분할하여 개발과 배포를 용이하게 하고, 이런 시스템을 대규모로 확장할 수 있는 마이크로서비스 아키텍처(Microservice Architecture)를 도입한 분산 시스템들이 등장하게 되었다. 하지만, 이러한 시스템들의 경우 규모 확장에는 유리하지만 시스템의 복잡성이 증가할 뿐만 아니라 시스템의 스케일이 증가할수록 전체 시스템에 대한 가시성이 줄어 들게 되며, 이는 트러블슈팅 등에 대한 오버헤드를 증가시키는 문제점을 야기하게 되었다[1].

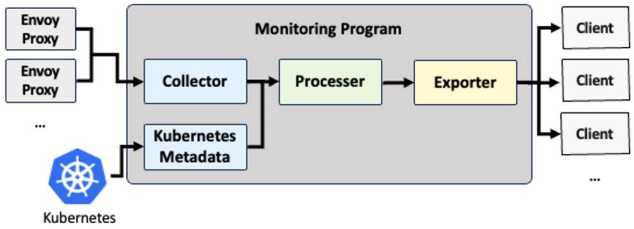
결국, 전체 시스템에 대한 관측 가능성 향상이 매우 중요한 과제로 대두되기 시작하였으며, 이를 해결하기 위해 현재 OpenTelemetry[2], Jaeger[3]와 같이 마이크로서비스 간 관계를 모니터링할 수 있는 도구들이 등장하게 되었다. 하지만, 기존의 도구들의 경우 각자 사전에 정해진 형식으로만 로그를 수집할 수 있고 추가적으로 수집을 원하는 데이터에 대한 커스터마이징이 어렵다는 제약이 있었다. 결국, 사용자가 원하지 않더라도 각 도구들이 자체적으로 정의한 모든 정보들을 수집하여야 하며, 네트워크 트래픽, 디스크 I/O 등의 관점에서 불필요한 오버헤드가 발생되는 문제점이 있다.

따라서, 본 연구에서는 이러한 문제점들을 보완하기 위해 대표적인 서비스 메시 환경인 Istio[4] 환경을 기준으로 기존의 도구들과 달리 사용자가 원하는 형식과 범위로 마이크로서비스 간 관계에 대한 로그 및 매트릭을 수집할 수 있는 경량화된 사이드카 기반 API 관측 가능성 및 원격 측정 시스템을 제안하고, 본 시스템에 대한 성능을 분석해보고자 한다.

2. API 관측 가능성 및 원격 측정 시스템

본 연구에서는 그림1과 같이 Istio 환경에서 사용자의 요구사항에 맞춰 설정 가능한 API 관측 가능성 및 원격 측정 시스템을 제안한다.

Istio와 같은 서비스 매시 환경에서는 개별 마이크로서비스(컨테이너)별로 Envoy Proxy를 사이드카 형태로 붙여 마이크로서비스 간 통신을 제어/관리한다. 본 시스템에서는 이러한 서비스 매시 환경의 특성을 이용하여 개별 마이크로서비스가 다른 마이크로서비스로 API 호출할 때 Envoy Proxy가 이를 제어/관리하기 위해 수집된 정보들을 gRPC를 이용하여 API 접근 로그와 매트릭을 수집한다.



(그림 1) API 관측 가능성 및 원격 측정 시스템 구조

하지만, Envoy Proxy로부터 수집된 API 접근 로그의 경우 IP주소를 기반으로 생성된 로그이기 때문에 마이크로서비스에 대한 정보가 부족하다. 따라서, 본 시스템에서는 수집된 API 접근 로그를 쿠버네티스 메타데이터와 병합하여 마이크로서비스 관점에서의 API 접근 로그를 다시 생성한다.

또한, 수집된 매트릭 역시 주어진 매트릭을 그대로 사용하는 것이 아닌, 사용자가 원하는 형식과 범위에 따라 특정 단위 시간별로, 주제별로 매트릭을 재구성하며, API 접근 로그와 같이 쿠버네티스 메타데이터와 병합하여 매트릭을 다시 생성한다.

마지막으로, 가공된 API 접근 로그와 매트릭은 gRPC를 통해 외부로 추출할 수 있도록 하여, 사용자가 원하는 형태로 다양하게 활용할 수 있도록 한다.

3. 실험 환경 및 성능 평가

<표1> 실험 환경

종류	항목
CPU/RAM	AMD Ryzen 9 5950X / 8GB
Linux	Ubuntu 22.04(5.15.0-67-generic)
SW	Kubernetes 1.24.17, Istio 1.21.0

주요 동작 및 성능 평가 환경은 표1과 같으며, 우선 사용자가 원하는 정보의 수집가능 여부를 평가하였다. 그리고, 그림 2와 같이 원하는 결과를 얻을 수 있음을 확인하였다.

(그림 2) 로그 및 매트릭 수집 결과

다음으로, 본 시스템의 성능을 평가하기 위해 로그 후처리와 출력 등으로 발생하게 되는 성능저하를 측정하였다. 그리고, 이를 위해 API 벤치마킹 도구를 컨테이너 내부에서 실행하여 요청을 생성하고 서버의 초당 API 요청 수를 측정하였다. 비교를 위해 컬렉터가 없는 환경, OepnTelemtry 컬렉터가 적용된 환경, 본 시스템을 적용한 환경에서 초당 API 요청 수를 측정하였다. 표 2는 8개 스레드로 8개의 연결을 사용하여 5초간 요청을 보낸 결과를 나타낸다.

<표 2> 실험 결과

컬렉터 유무	요청 수 (Requests/sec)
컬렉터가 없는 환경	3557
OpenTelemetry	3150
제안 시스템	2915

본 실험을 통해, 제안한 시스템을 사용하는 경우 컬렉터가 없는 환경 대비 약 18%, OpenTelemetry 컬렉터가 적용된 환경 대비 약 7.5%의 성능 저하를 발생함을 확인하였다.

4. 결론

본 연구에서는 Istio 서비스메시 환경에서 동작하는 사이드카 기반 API 관측 가능성 및 원격 측정 시스템을 구현하여 기능과 이로 인해 발생하는 성능 저하를 평가하였다. 그 결과, Istio 서비스메시 환경에서 효과적으로 작동하며, 사용자가 원하는 정보를 수집할 수 있음을 확인하였다. 하지만, 기존의 원격 측정 도구인 OpenTelemetry에 비해 약간의 성능 저하가 발생한다는 점을 알 수 있었으며, 추가적인 최적화가 필요하다는 점을 확인하였다. 그리고, 향후 연구로 로그 및 매트릭 후처리 로직 최적화 및 LSTM과 SQLite를 활용한 로그 집계 및 분류와 관련된 추가 기능에 대한 연구를 진행할 계획이다.

Acknowledgement

이 성과는 2024년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임. (RS-2023-00212738)

참고문헌

[1] Lei Zhang, "The Benefit ofHindsight: Tracing Edge-Cases in Distributed Systems," Proceedings of the20th USENIX Symposium on Networked Systems Design and Implementation, 2023.

[2] OpenTelemetry, https://opentelemetry.io

[3] Jaeger: open source, distributed tracing system, https://jaegertracing.io

[4] Istio Service Mesh, https://istio.io