Envoy 및 TC-BPF를 통한 마이크로서비스 환경에서의 API 관측 가능성 향상에 관한 연구

김지수[1], 남재현[2]
[1]단국대학교 인공지능융합학과 석사과정
[2]단국대학교 컴퓨터공학과 교수
[1]imjs0807@dankook.ac.kr, [2]namjh@dankook.ac.kr

Enhancing API Observability in Microservices Through Envoy and eBPF-based Traffic Control

Ji-Su Kim[1], Jaehyun Nam[2]
[1]Dept. of Artificial Intelligence Convergence, Dankook University (Graduate Student)
[2]Dept. of Computer Engineering, Dankook University (Professor)

요 약

최근 많은 기업들이 시스템의 유연성과 확장성을 고려하여 컨테이너 기술과 API 호출을 기반으로 하는 마이크로서비스 아키텍처를 도입하고 있다. 그러나 마이크로서비스의 수가 증가하고 시스템의 복잡성이 커짐에 따라 디버깅과 모니터링의 어려움이 함께 증대되고 있다. 특히, 분산된 서비스 구조에서는 API 요청이 어떤 경로로 처리되는지 추적하는 과정이 매우 복잡해져, 이는 전체 시스템의 운영과 유지 관리에 부정적인 영향을 미칠 수 있다. 이러한 문제를 해결하기 위해, 본 논문에서는 Envoy, WebAssembly(WASM), 그리고 Traffic Control BPF (TC-BPF)를 결합한 API 관측 가능성을 향상시키기 위한 시스템을 제안한다. 이 시스템은 세 가지 구성요소인 설정, 수집, 통합 엔진으로 이루어져 있으며, 패킷 처리 과정 중 각 엔진을 통해 API 호출에 대한 추적 가능한 데이터를 추출하고 이를 통합한다. 제안된 시스템은 기존 도구들이 겪는 성능 저하 및 인증 정보 수집의 한계를 극복하여, 분산된 마이크로서비스 환경에서 API 관측 가능성을 효과적으로 향상시킬 수 있다.

1. 서론

마이크로서비스 아키텍처(Microservices Architecture, MSA)는 시스템의 유연성과 확장성을 극대화할 수 있는 특징으로 인해 현대 소프트웨어 개발 분야에서 큰 주목을 받고 있다. 많은 기업들이 MSA를 구현하기 위해 컨테이너 기술과 REST API 같은 API 호출 방식을 활용하여 서비스 간 통신을 효율적으로 처리하고 있다. 그러나 서비스의 수가 증가하고 시스템의 복잡도가 높아지면서, 전체 시스템의 디버깅과 모니터링이 점차 어려워지는 문제에 직면하고 있다. 특히 분산된 서비스 구조에서는 하나의 요청이 어떤 경로를 통해 처리되는지 추적하는 것이 매우 복잡해지며, 이는 시스템 운영과 유지 관리에 상당한 어려움을 초래한다[1].

이러한 문제를 해결하기 위해 Istio와 같은 서비스 메시 솔루션이 등장했다. Istio는 사이드카 패턴을 통해 각 서비스 옆에 Envoy 프록시를 배치하여 트래픽을 제어하고 모니터링하는 역할을 한다. 이를 통해 서비스 간 통신이 효율적으로 처리되며, 시스템 전반의 가시성을 높일 수 있다. 이와 함께 다양한 연구 및 도구들이 Istio의 기능을 확장하고 최적화하기 위해 개발되었으며[2], 대표적으로 Kiali는 쿠버네티스 환경에서 서비스 간 통신을 추적하고 시각화하는 도구로, 서비스 관계와 트래픽 흐름을 직관적으로 보여줌으로써 디버깅과 모니터링 작업을 지원한다.

그러나 이러한 도구들도 몇 가지 한계를 지닌다. 예를 들어, Kiali는 사용자 공간에서 데이터를 수집하기 때문에 네트워크 패킷이나 인증에 사용되는 토큰과 같은 세부적인 정보를 완전히 식별하는 데는 한계가 있다. 이는 특히 서비스 간 통신에서 발생하는 세부적인 네트워크 패킷 정보나 인증 데이터를 충분히 수집하지 못하는 문제를 초래한다[3].

이에 본 연구에서는 기존 방법의 한계를 극복하기 위해 Envoy, WebAssembly (WASM), 그리고 Traffic Control BPF (TC-BPF)를 결합한 접근 방식을 제안한다. 또한, 플러그인 형태로 쉽게 확장 가능하면서 기존에 관측하기 어려웠던 요청 데이터를 효과적으로 식별하고 추적할 수 있는 시스템을 제안한다. 이를 통해 마이크로서비스 환경에서 발생하는 디버깅과 트레이싱 문제를 해결하고, 시스템의 전반적인 운영 가시성을 향상시키고자 한다.

2. 배경 지식

2.1 쿠버네티스 네트워킹과 서비스 매쉬

쿠버네티스(Kubernetes)는 컨테이너화된 애플리케이션의 배포 및 관리를 자동화하는 컨테이너 오케스트레이션 도구로, 대규모 분산 시스템의 운영을 효율적으로 지원한다. 쿠버네티스는 CNI(Container Network Interface)를 활용하여 컨테이너 간의 네트워크 통신을 설정하고 관리한다[4]. 그러나 CNI만으로는 마이크로서비스 아키텍처 내에서 발생하는 복잡한 서비스 간 통신을 효과적으로 처리하는 데 한계가 있다. 특히, 서비스 간 모니터링, 보안 정책 적용 등 고급 기능을 구현하기 위해서는 추가적인 도구나 작업이 필요하다. 이러한 한계를 해결하기 위해 Istio와 같은 서비스 메시 솔루션이 도입되었다.

Istio는 쿠버네티스 환경에서 동작하는 서비스 메시로, 애플리케이션 코드를 수정하지 않고도 서비스 간 통신을 제어하고 모니터링할 수 있는 기능을 제공한다[5]. Istio는 사이드카 프록시(sidecar proxy) 방식을 활용하여 쿠버네티스의 파드(pod)마다 프록시를 배치하고, 모든 트래픽을 이 사이드카 프록시를 통해 처리하도록 설정한다. 이를 통해 트래픽 라우팅, 모니터링, 보안 정책 적용 등의 기능을 중앙에서 관리할 수 있다. 이때 주로 사용되는 사이드카 프록시가 Envoy이다. Envoy는 클라우드 네이티브 애플리케이션을 위한 고성능 서비스 프록시로, 마이크로서비스 아키텍처에서 서비스 간 통신을 제어하고 모니터링하는 데 중요한 역할을 한다[6]. Envoy는 HTTP 트래픽 제어에 뛰어나며, 필터 체인 구조를 통해 모니터링, 인증 등의 기능을 확장할 수 있다. 또한, WebAssembly(WASM)를 통해 사용자 정의 로직을 삽입하여 HTTP 요청을 보다 세밀하게 분석할 수 있는 유연성을 제공한다[7].

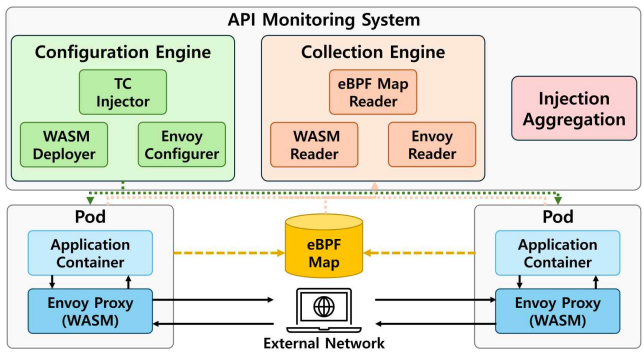
2.2 Traffic Control BPF

Traffic Control BPF (TC-BPF)는 리눅스 커널의 네트워크 트래픽 제어 기능을 제공하는 도구로, 네트워크 인터페이스를 통해 트래픽을 효율적으로 관리한다. TC-BPF는 eBPF를 활용하여 커널 레벨에서 네트워크 패킷을 실시간으로 필터링하고 분석할 수 있으며, 이는 유저 스페이스에서 작동하는 기존의 libpcap 방식보다 성능 면에서 더 우수하고 오버헤드가 적다[8].

TC-BPF는 네트워크 스택의 여러 단계에서 트래픽을 가로채고 필터링하여 네트워크 성능을 최적화한다. 패킷이 소켓 버퍼(socket buffer) 형태로 존재할 때 이를 가로채어 패킷의 헤더 및 메타데이터를 분석할 수 있다. 이러한 기능을 통해 각 패킷의 출발지 및 목적지 주소, 메타데이터를 실시간으로 확인할 수 있으며, 이를 바탕으로 네트워크 흐름을 세밀하게 모니터링할 수 있다. 특히, 애플리케이션 레이어에서 발생하는 이벤트를 정확하게 식별해야 하는 상황에서 TC-BPF는 매우 효과적인 도구로 활용될 수 있다.

3. API 모니터링 시스템

쿠버네티스를 기반으로 한 마이크로서비스 아키텍처에서는 시스템의 안정성과 성능을 유지하기 위해 모니터링이 필수적이다. 마이크로서비스는 각 서비스가 독립적으로 상호작용하기 때문에, 통신 장애나 성능 저하와 같은 문제를 신속하게 감지하지 못하면 시스템 전체에 심각한 영향을 미칠 수 있다.



(그림 1) 실시간 API 모니터링 시스템

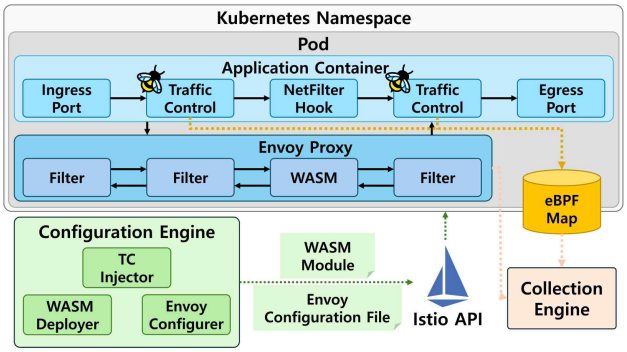
이러한 문제를 해결하기 위해 본 논문에서는 (그림 1)에 제시된 바와 같이 Configuration, Collection, Aggregation 엔진으로 구성된 실시간 API 관측 가능성 시스템을 제안한다. Configuration 엔진은 API 데이터 수집을 위해 Envoy 설정, WASM 모듈, eBPF 프로그램 등을 배포하는 역할을 한다. Collection 엔진은 Envoy와 eBPF를 사용하여 API 호출 및 네트워크 패킷에서 수집된 데이터를 처리하며, 이 데이터를 Aggregation 엔진으로 전송하여 통합한다.

3.1 Configuration Engine

Configuration Engine은 (그림 2)와 같이 API 관측을 위한 환경을 설정하고, 실시간으로 파드 및 컨테이너를 식별하며 다양한 모니터링 소스를 구성하는 역할을 담당한다. 먼저, Istio의 Envoy 사이드카 프록시에서 API 로그를 수집하기 위해 Envoy Configurer가 필요한 설정을 구성하고 이를 배포한

다. 이 과정에서 Istio의 ConfigMap을 수정하여, Envoy가 API 요청의 흐름을 모니터링할 수 있도록 설정하고, 수집된 데이터를 Collector로 전송할 수 있게 한다.

또한, API 로그에서 인증 토큰을 추출하기 위해 WASM Deployer는 Istio의 WasmPlugin을 활용하여 Envoy의 필터 체인에 WASM 모듈을 삽입한다. 이를 통해 WASM에서 추출된 정보를 Collector Engine으로 전달할 수 있도록 적절한 주소를 설정한다. 마지막으로, TC Injector는 컨테이너 내부의 네트워크 인터페이스에 TC-BPF 프로그램을 적용하여, 해당 컨테이너의 네트워크 트래픽을 실시간으로 추적할 수 있도록 한다.



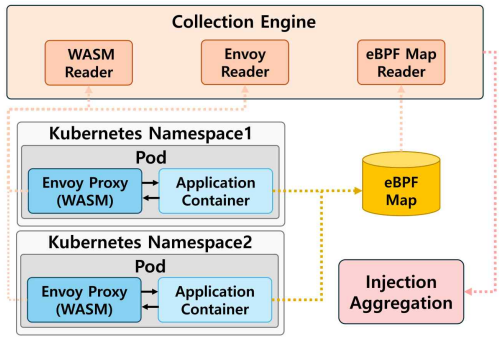
(그림 2) Configuration Engine 동작도

3.2 Collection Engine

Collection Engine은 (그림 3)에서 볼 수 있듯이, Envoy, WASM, 그리고 TC-BPF를 통해 수집된 다양한 데이터를 처리하는 핵심 역할을 수행한다. 먼저, Envoy 사이드카 프록시에서 수집된 API 로그는 Collection Engine으로 전송되어 API 요청 경로 및 관련 세부 정보가 기록된다. Envoy는 Istio 서비스 메시 내에서 각 파드에 배포되어 서비스 간 통신을 가로채고, 수집된 API 로그는 gRPC 프로토콜을 통해 Collection Engine에 전달된다.

또한, 인증 토큰과 같은 중요한 데이터는 기본적인 API 로그에서는 추출하기 어렵기 때문에, WASM 모듈을 통해 수집된다. WASM 모듈은 HTTP/1.1 프로토콜을 사용하여 데이터를 수집하며, 이를 Collection Engine으로 전달하기 위해 별도의 데이터 처리 구성이 필요하다.

마지막으로, eBPF 기반의 TC-BPF프로그램은 각 컨테이너의 네트워크 인터페이스에서 발생하는 트래픽을 실시간으로 추적한다. Envoy와 WASM은 주로 애플리케이션 레이어의 데이터를 식별하는 데 중점을 두기 때문에, 패킷 식별자나 TCP 시퀀스 번호와 같은 네트워크 레벨의 세부 정보는 수집하는 데 한계가 있다. TC-BPF는 이러한 정보를 수집하여 eBPF 맵을 통해 Collection Engine에 전달하며, 이를 통해 패킷 식별자와 TCP 시퀀스 번호 등 네트워크 레벨에서의 통신 흐름을 더욱 정밀하게 분석할 수 있다. 또한, 컨테이너는 호스트 시스템의 프로세스로 동작하므로, 호스트 커널에서 하나의 eBPF 맵을 통해 여러 파드의 트래픽을 동시에 수집하는 것이 가능하다.



(그림 3) Collection Engine 동작도

3.3 Aggregation Engine

Aggregation Engine은 Collection Engine으로부터 수집된 다양한 데이터를 통합하여 API 요청 흐름을 분석하고, 관련 트래픽을 연결하는 역할을 수행한다. Envoy와 WASM에서 수집된 데이터는 공통적으로 x-request-id를 포함하고 있어, 이를 기반으로 API 로그와 인증 토큰 정보를 연계할 수 있다. 또한, TC-BPF에서 수집된 패킷의 출발지 및 목적지 정보를 활용하여 네트워크 레벨의 트래픽과 애플리케이션 레벨의 API 요청을 연결할 수 있다.

이러한 데이터 통합을 통해, WASM 모듈에서 추출한 인증 토큰, Envoy를 통한 기본적인 API 로그, 그리고 TC-BPF를 통해 수집된 패킷 식별자와 TCP 시퀀스 번호를 하나의 흐름으로 결합할 수 있게 되어, 보다 정밀하고 심층적인 모니터링과 분석이 가능해진다. 이를 통해 API 호출 경로 및 트래픽 흐름을 전반적으로 파악할 수 있으며, 네트워크와 애플리케이션 레이어 간의 연계성을 향상시킨다.

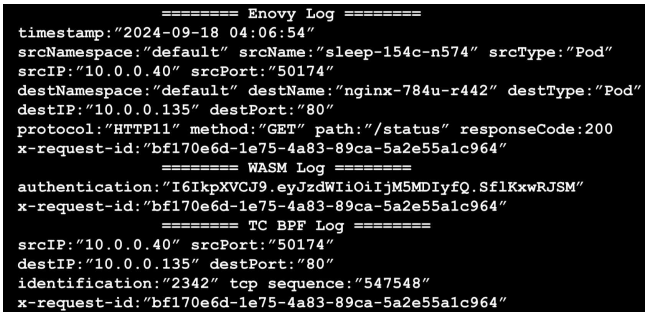
4. 실험 환경 및 로그 수집

본 장에서는 제안된 API 모니터링 시스템의 기능을 검증하기 위한 실험 결과를 제시한다. 실험 환경은 <표 1>에 명시된 대로 구성하였다.

<표 1> 실험 환경

종류	항목
CPU/RAM	Intel(R) Xeon(R) Gold 5220R CPU @ 2.20GHz / 8GB
Linux	Ubuntu 22.04(5.15.0-78-generic)
SW	Kubernetes 1.29.8, Istio 1.23.0

(그림 4)는 제안된 시스템을 통해 수집된 데이터를 시각화한 결과로, nginx 이미지를 이용하여 간단한 웹 서비스 파드를 생성하고 해당 파드에 API 요청을 발생시켜 로그를 수집한 사례이다.



(그림 4) Aggregation Engine에서 식별한 정보

(그림 4)에서 볼 수 있듯이, Envoy, WASM, TC-BPF를 활용하여 서로 다른 레이어에서 수집된 정보를 종합적으로 수집하는 것이 가능했다. 특히, 각 레이어에서 수집된 로그가 동일한 x-request-id 필드를 공유함으로써 요청 경로의 전반적인 흐름을 명확하게 추적할 수 있었다. 또한, TC-BPF를 통해 수집된 네트워크 레벨의 정보는 Envoy와 WASM 로그만으로는 확인할 수 없는 세부적인 통신 데이터를 포함하고 있음을 확인하였다. 이를 통해 네트워크 및 애플리케이션 레벨에서의 종합적인 모니터링이 가능함을 입증할 수 있었다.

5. 결론 및 향후 연구

본 연구에서는 마이크로서비스 아키텍처 환경에서 API 요청을 효율적으로 모니터링하고 분석하기 위한 새로운 접근 방식을 제안하였다. 마이크로서비스 환경에서 발생하는 복잡한 서비스 간 통신은 추적이 어렵고, 특히 기존의 사용자 공간에서 동작하는 모니터링 도구는 성능 저하와 세부 정보 수집의 한계가 있었다. 이러한 문제를 해결하기 위해, 본 연구에서는 Envoy, WASM, 그리고 TC-BPF를 결합한 시스템을 설계하여, 애플리케이션 레벨과 네트워크 레벨에서 통신 흐름을 통합적으로 추적하는 방안을 제시하였다. 그 결과, 다양한 레이어에서 수집된 데이터를 x-request-id를 통해 연계함으로써 단일 API 요청에 대한 종합적이고 심층적인 분석이 가능해졌으며, 네트워크와 애플리케이션 레벨의 세부 정보를 동시에 확보할 수 있었다. 이를 통해 마이크로서비스 환경에서 디버깅, 성능 최적화, 보안 강화와 같은 다양한 측면에서 실질적인 개선 효과를 기대할 수 있다.

향후 연구에서는 TC-BPF를 통해 Envoy와 WASM에서 획득하기 어려운 정보를 추가로 식별하고, 이를 관련 로그와 연계하여 더욱 정밀한 분석을 수행하는 방법을 탐구할 계획이다. 또한, 단일 노드가 아닌 클러스터 전체에서 API 요청의 흐름을 추적하는 방안을 제시함으로써, 보다 복잡한 분산 시스템에서도 통합적인 모니터링이 가능하게 하는 연구도 진행할 계획이다.

Acknowledgement

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

참고문헌

[1] M. Cinque, R. Della Corte, and A. Pecchia, “Microservices monitoring with event logs and black box execution tracing,” IEEE Trans. Serv. Comput., vol. 15, no. 1, pp. 294 - 307, Jan. 2019.

[2] Bento, Andre, et al. “Automated analysis of distributed tracing: Challenges and research directions.” Journal of Grid Computing 19.1 (2021): 9.

[3] S. Y. Nikouei, R. Xu, Y. Chen, A. Aved, and E. Blasch, “Decentralized smart surveillance through microservices platform,” in Sensors and Systems for Space Applications XII, vol. 11017. Bellingham, WA, USA: SSPIE, 2019, Art. no. 110170K.

[4] Y. Park, H. Yang, and Y. Kim, “Performance analysis of CNI (Container networking Interface) based container network,” in Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC), Oct. 2018, pp. 248 - 250.

[5] “Istio” [Internet], https://istio.io/

[6] “envoy” [Internet[, https://www.envoyproxy.io/

[7] “WASM” [Internet], https://webassembly.org/

[8] “eBPF” [Internet], https://ebpf.io/