실시간 네트워크 중심 롤링 배포 신뢰성 향상 전략

오석원, 강상위, 권태경
서울대학교

swoh@mmlab.snu.ac.kr, swkang@mmlab.snu.ac.kr, tkkwon@snu.ac.kr

Real-Time Network Centric Rolling Deployment Reliability Enhancement Strategy

Seokwon Oh, Sangwi Kang, Taekyoung "Ted" Kwon
Seoul National Univ.

요 약

다수의 사용자를 보유한 대규모 소프트웨어 어플리케이션의 경우 크고 작은 다양한 소프트웨어 변경 사항이 발생한다. 버그 수정, 신규 기능, 내부 모듈 업그레이드 등의 사유로 주기적인 소프트웨어 업데이트가 요구된다. 대규모 소프트웨어 어플리케이션의 경우 롤링 배포를 수행하고 이상 상황이 탐지되는 경우 어플리케이션에 최소한의 영향을 주며 롤백을 수행할 수 있다. 본 논문은 네트워크 수준의 지표를 통해 롤링 배포 시 이상 상황을 파악하는 전략을 제시한다. 이상 상황이 어플리케이션에 영향을 주기 전 네트워크 수준의 지표를 활용하여 이상 상황을 사전에 감지하고 롤백을 신속하게 수행하며 배포 신뢰성을 향상시키는 방법을 제시한다.

Ⅰ. 서 론

대규모 소프트웨어 어플리케이션의 경우 크고 작은 다양한 종류의 소프트웨어 변경 사항이 발생한다. 오류 정정, 신규 기능, 내부 모듈 업그레이드 등의 변경 사항이 존재하고 이는 주기적인 소프트웨어 업데이트를 요구한다.

이러한 소프트웨어 업데이트는 어플리케이션의 동작에 영향을 주지 않으면서 수행되는 것이 요구된다. 소프트웨어 업데이트로 인해 어플리케이션이 비정상 동작하는 것은 더욱 방지되어야 한다. 이에 따라 점진적으로 인스턴스에 소프트웨어 업데이트를 수행하는 롤링 배포[1] 방식이 존재한다. 모든 인스턴스를 한 번에 업데이트를 수행하는 것이 아닌 하나씩 점진적으로 업데이트를 수행하는 방식이다. 어플리케이션의 비정상 동작을 방지하기 위한 대표적인 배포 방법 중 하나이다.

롤링 배포 수행 시 어플리케이션의 비정상 동작을 탐지하기 위해서는 어플리케이션 수준의 지표를 활용하는 것이 직관적이다. 예를 들어 API 에러 발생 비율, HTTP Request/Response 완료 시간, CPU 사용량을 기반으로 비정상 동작을 탐지하는 것이 가능하다. 다만 어플리케이션 수준의 지표를 활용하여 비정상 동작을 판단하는 것은 이미 어플리케이션에서 비정상 동작이 존재한다는 것을 의미하고 사용자 불편함과 이상 현상이 이미 발생되었다는 것을 의미한다. 또한 롤링 배포 중 인스턴스 간 호환성 문제와 같은 롤링 배포 자체가 초래하는 비정상 동작도 존재한다.

본 논문에서는 네트워크 수준에서 이상 상황을 파악하고 어플리케이션에 영향을 주기 전에 롤링 배포를 중단하고 롤백을 수행하는 전략을 제시한다. 커널 레벨에서 수집 가능한 네트워크 수준 지표를 통해 이상 상황을 조기 탐지하고 롤백을 수행하는 롤링 배포 신뢰성 향상 전략을 제시한다.

Ⅱ. 본론

롤링 배포란 소프트웨어 업데이트를 점진적으로 인스턴스에 적용하는 배포 방법이다. 롤링 배포 수행 중에 이상 현상이 발생하면 어플리케이션에 적은 영향을 주며 롤백을 수행할 수 있다. 그림 1 과 같이 신규 버전인 Version 2 가 인스턴스에 점진적으로 적용되는 배포 방식이다.

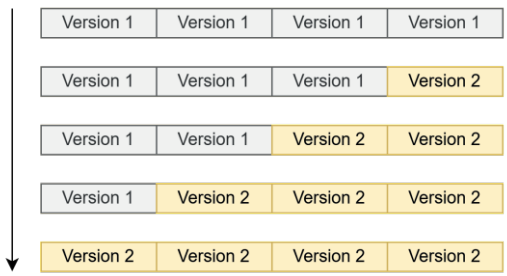

그림 1 롤링 배포

롤링 배포 시 이상 현상 탐지하는 방법에 대해서는 프로토콜, 가이드라인이 존재하지 않는다. 직관적으로 어플리케이션 수준의 지표를 사용할 수 있다. API 성공률, 에러 비율, RPS (Request Per Second), CPU/Memory 사용량, 비즈니스 로직 수준 통계 등의 지표를 사용하여 이상 현상을 탐지 가능하다.

본 논문에서는 네트워크 수준의 지표를 사용하는 이상 현상 탐지 전략을 제시한다. TCP-Level 지표, UDP-Level 지표, TLS-Level 지표, DNS-Level 지표, 공통 Packet-Level 지표, Socket-Level 지표를 활용하여 이상 현상 탐지 가능하다. 본 논문에서는 대표적으로 TCP 재전송 지표를 통해 예를 들어 설명하겠다.

이상 현상 탐지하는 방법은 2 가지 방식이 존재한다. 비교 방식과 임계 값 방식이다. 비교 방식은 그림 2 와 같이 신규 버전이 설치된 인스턴스와 기존 인스턴스를 비교하는 방식이다. 임계 값 방식은 그림 4 와 같이 설정된 임계 값 기준 이상 현상을 탐지하는 방식이다.

비교 방식은 소프트웨어 업데이트가 마이너 수정 혹은 로직에 변경이 없는 경우에 적용 가능한 방식이다. 비교 방식은 신규 인스턴스와 기존 인스턴스의 지표를 비교하는 방식이다. 동일한 작업이 두 인스턴스에서 실행이 되었을 때 동일한 TCP 재전송이 일어날 것이다. 이에 따라 신규 인스턴스와 기존 인스턴스를 실시간 비교하며 동일한 지표를 보일 때 다음 단계로 넘어가는 전략이다. 또한 그림 3 의 공식 적용 가능하다.

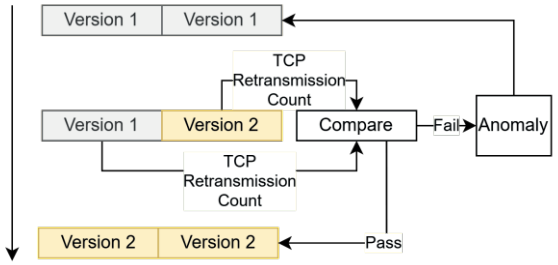

그림 2 비교 방식

$$RelativeChange = \frac{Metric(Version2) - Metric(Version1)}{Metric(Version1)}$$

$|RelativeChange| > \epsilon$ then Fail otherwise Pass

그림 3 비교 공식

임계 값 방식은 소프트웨어 업데이트가 메이저 수정 혹은 신규 로직이 도입되는 경우에 적용 가능한 방식이다. 임계 값 방식은 기존 인스턴스에서 추출한 지표 기반 임계 값을 설정하는 것이다. 신규 인스턴스에서 실시간으로 지표를 추출하고 해당 지표가 임계 값을 초과하는 경우 이상 현상으로 판단하는 것이다. 또한 그림 5 의 공식 적용 가능하다.

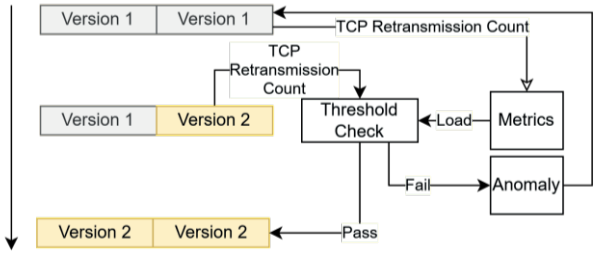

그림 4 임계 값 방식

$$Threshold = Metric(Version1) \times T$$

$Threshold < Metric(Version2)$ then Fail otherwise Pass

그림 5 임계 값 방식 공식

어플리케이션 수준 지표만 활용하면 네트워크 오류가 발생해도 네트워크 오류는 은폐될 가능성이 존재한다. 또한 비정상적인 네트워크 패턴 감지도 되지 않을 것이다.

네트워크 중심 롤링 배포 신뢰성 향상 기법은 False Positive 가 발생할 수 있다는 한계점이 존재한다. 비교 방식에서 실패한 경우에도 로직이 의도한 방식으로 동작할 수 있고, 임계 값 방식에서 실패한 경우에도 로직이 의도한 방식으로 동작할 수 있다. 다만 어플리케이션 비정상 동작이 초래하는 부정적 영향을 방지하기 위해 False Positive 롤백은 감수 가능하다고 판단된다.

Unmanned Aerial Mobility 의 경우 안전성, 신뢰성 보장을 위해 롤링 배포 사용 가능하다. 모든 플라잉카에 소프트웨어 업데이트를 수행할 때 롤링 배포를 사용하면 문제 발생 시 신속한 롤백이 가능하다. 어플리케이션 수준의 지표만 활용하면 문제가 이미 발생한 후에 롤백이 수행되는 것을 의미한다. 또한 플라잉카에 문제가 발생하면 큰 사고로 이어질 수도 있다. 하지만 네트워크 수준의 지표를 활용해서 문제를 탐지하면 실제 문제가 발생하기 전에 롤백을 시작하는 것이 가능할 것이다. 안전성, 신뢰성이 중요한 어플리케이션에서 유용하게 사용 가능한 전략으로 판단된다.

구현은 eBPF(extended Berkeley Packet Filter)[2] 통해서 가능할 것으로 예상된다. Hook 형식으로 커널의 이벤트를 모니터링 할 수 있다. DeepFlow[3]의 경우 Ingress System Call, Outgress System Call 구분하는 방식을 수행했다. 본 전략도 Ingress System Call, Outgress System Call 구분하여 지표 수집하고 이상 현상 탐지에 활용 가능할 것으로 판단된다.

향후 네트워크 수준, 어플리케이션 수준 지표를 통합하여 이상 현상을 판단하는 전략도 가능할 것으로 예상된다.

Ⅲ. 결론

본 논문에서는 롤링 배포 상황에서 네트워크 수준 지표를 활용해서 이상 현상을 탐지하고 롤백을 수행하는 전략을 제시한다. 어플리케이션 로직의 변경 여부에 따라 비교 방식, 임계 값 방식으로 이상 현상 탐지 가능하다. 또한 eBPF 통해서 구현 가능하고 향후 네트워크 수준 지표와 어플리케이션 수준 지표를 통합하여 이상 현상을 판단하는 전략도 가능할 것으로 예상된다.

ACKNOWLEDGMENT

이 연구는 정부 (과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. RS-2023-00220985)

참 고 문 헌

[1] Rolling Deployment. https://www.techtarget.com/searchitoperations/definition/rolling-deployment

[2] eBPF. https://ebpf.io/

[3] Junxian Shen, Han Zhang, Yang Xiang, Xingang Shi, Xinrui Li, Yunxi Shen, Zijian Zhang, Yongxiang Wu, Xia Yin, Jilong Wang, Mingwei Xu, Yahui Li, Jiping Yin, Jianchang Song, Zhuofeng Li, and Runjie Nie. 2023. Network-Centric Distributed Tracing with DeepFlow: Troubleshooting Your Microservices in Zero Code. In Proceedings of the ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23). Association for Computing Machinery, New York, NY, USA, 420- 437.