서비스 메시 환경에서 강화학습을 이용한 트래픽 부하 분산 메커니즘에 관한 연구

김채호[1], 남재현[2]
[1]단국대학교 인공지능융합학과 석사과정
[2]단국대학교 컴퓨터공학과 교수

boanchkim@dankook.ac.kr, namjh@dankook.ac.kr

Reinforcement Learning-based Traffic Load Balancing in Service Mesh Environments

Chae-Ho 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

서비스 메시 환경에서의 트래픽 분산은 시스템의 성능과 안정성, 그리고 보안에 필수적인 기능을 담당한다. 현재의 트래픽 분산 방식들은 대부분 정적 설정을 기반으로 하기 때문에 시스템 환경의 변화에 신속하게 대응하기 어렵고 최적화된 성능을 보장하기 힘들다. 본 논문에서는 강화학습을 활용해 서비스 메시 환경 내에서의 트래픽 분산을 자동화하고 최적화할 수 있는 새로운 시스템을 제안한다. 특히, 텔레메트리 기술을 활용해 트래픽의 분산을 실시간으로 추적하며, 강화학습 알고리즘을 이용해 트래픽 가중치를 조정함으로써 기존의 로드 밸런싱 방법들에 비해 더 빠른 처리 시간과 보다 효율적인 로드 밸런싱을 달성할 수 있을 것으로 기대한다.

1. 서론

모놀리식 아키텍처는 소프트웨어 애플리케이션을 통합된 하나의 독립 시스템으로 구축하는 구조를 말한다. 이는 애플리케이션의 모든 비즈니스 로직을 한 곳에 모으는 방식으로, 애플리케이션 규모가 클 경우 소소한 수정에도 전체를 다시 빌드하고 배포해야 하며, 많은 코드가 집중되어 있어 유지보수가 어렵다는 단점이 있다.

MSA (Microservice Architecture)는 이러한 모놀리식 아키텍처의 제약을 극복하고자 나온 개념으로, 애플리케이션을 여러 작고 독립적인 서비스로 나누어 구축하는 방식이다. MSA 를 통해 대규모 프로젝트를 진행할 때 각각의 서비스를 독립적으로 배포할 수 있으며, 서비스는 자신의 접근점을 API 형태로 외부에 노출하고 내부적인 세부사항은 추상화 하여 관리한다.

서비스 메시는 MSA 의 서비스들이 통신하는데 중심이 되는 인프라 계층으로, 서비스 간의 통신을 추상화하고 서비스 발견, 로드 밸런싱, 인증, 모니터링 등을 담당한다. 이는 서비스 간의 효율적이고 안정적인 통신에 핵심적인 역할을 한다. 서비스 메시에서의 트래픽 로드 밸런싱은 서비스 요청을 다수의 서비스 인스턴스에 고르게 분산하여 처리하는 방법이다.

특히 Istio 와 같은 서비스 메시 솔루션은 쿠버네티스와 잘 통합되어 있으며, 다양한 로드 밸런싱 방법을 제공하여 효과적인 트래픽 분산을 도와준다. 하지만 Istio 의 로드 밸런싱은 정적으로 설정되기 때문에, 트래픽 패턴 변화에 따라 자동으로 로드 밸런싱 정책을 조정하지 못하는 단점이 있다.

본 연구에서는 MSA 환경에서 서비스 메시의 Ingress 트래픽 변화에 맞춰 최적의 로드 밸런싱을 추천하는 강화학습 기반의 시스템을 제안한다. 특히, 텔레메트리를 이용하여 서비스 간 트래픽을 실시간으로 추적하고, 강화학습으로 서비스 자원의 상태를 파악하여 최적화된 로드 밸런싱을 수행하게 된다.

2. 관련 연구

2-1. 분산 추적

분산 추적은 구글이 2010 년에 발표한 Dapper 백서를 기반으로 한 개념이다[1]. 이 개념은 각 서비스가

수행한 작업에 관한 정보를 방출하도록 하여 서비스들의 상호작용 방식을 간접적으로 관찰하고, 공통된 식별자를 통해 이 정보들을 서로 연결하는 것이다. 고유한 식별자인 추적 ID 를 사용해 특정 요청을 모니터링하면, 그 요청이 수백 개의 다양한 서비스와 상호작용하는 과정에서 어떻게 처리되는지를 파악할 수 있다.

분산 추적은 클러스터 전반에 걸친 통합된 가시성을 확보하는 전략의 중요한 부분이다. 마이크로서비스 기반 애플리케이션 내부에서 발생하는 일들을 이해하기 위해서는, 해당 애플리케이션 내의 컨테이너 작동에 대한 맥락을 먼저 파악할 필요가 있다. 또한 서비스 요청 기간 동안 수집된 다양한 원격 측정 데이터들 사이의 상관관계를 분석하여, 사고가 왜 발생했는지를 명확하게 하는 감사 추적 기능도 제공한다.

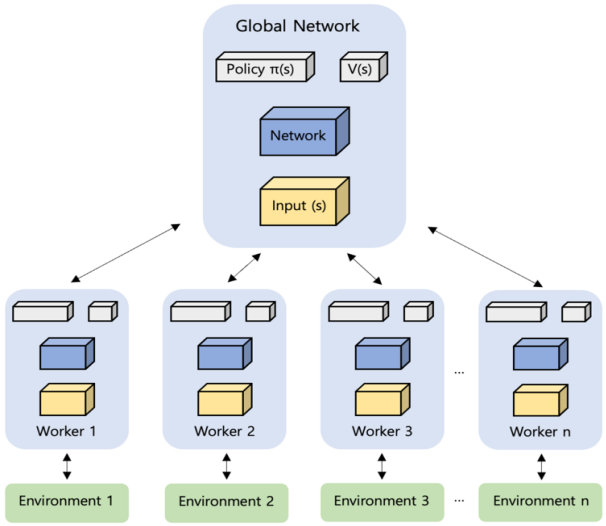
2-2. 로드 밸런서

로드 밸런서는 서버나 서비스에 가해지는 부하를 분산하는 장치나 기술을 말한다. 클라이언트와 서버 풀 사이에서 작동하며, 서버에 부하가 집중되는 것을 방지하고 각 서버가 최적의 성능을 낼 수 있게 트래픽을 조절한다. 요청을 받은 로드 밸런서는 설정된 규칙에 기초해 서버로 요청을 전달하며, 첫 번째로 서버가 요청을 적절히 처리할 수 있는지를 확인한다. 그 후 건강한 서버 풀에서 서버를 선택한다. 로드 밸런싱에는 여러 기술이 사용되는데, 대표적으로 라운드 로빈, 소스 기반, 목적지 기반, 최소 연결, 그리고 최소 응답 시간 등이 있다[2].

2-3. 강화 학습

강화학습은 다양한 가능성을 스스로 탐색하며 장기적인 목표를 달성하기 위한 방법론이다. 이 학습 방식에서는 에이전트가 환경과의 직접적인 상호작용을 통해 훈련 데이터를 수집한다. 이렇게 얻어진 학습 데이터는 별도로 준비된 데이터 집합이 아니라 에이전트의 경험 자체로서, 연구자가 데이터를 선별하고 학습시키는 부담을 덜어준다. 강화학습 알고리즘은 적응적인 특성을 가지고 있어 환경의 변화에 대응할 수 있도록 설계되었다. 기존의 머신러닝 알고리즘과는 다르게, 강화학습에서는 시간이 중요한 요소로 작용하며, 에이전트가 수집하는 경험은 독립적이지 않고 일정한 분포를 갖지 않는다.

본 연구에서 사용하는 강화학습 알고리즘은 A3C 이다. A3C 는 Asynchronous Advantage Actor-Critic 의 줄임말로, 'Asynchronous'는 여러 에이전트가 동시에 실행되며 주기적이거나 비동기적으로 공유 네트워크를 업데이트한다는 뜻의 '비동기적'이라는 의미를 담고 있다[3]. 각각의 에이전트는 다른 에이전트와 독립적으로, 업데이트를 원하는 시점에 공유 네트워크를 업데이트하므로 '비동기적'이라는 용어가 이름에 포함되었다. 그림 1 은 A3C 알고리즘의 구조를 나타낸 것이다.



(그림 1) A3C 의 아키텍처 다이어그램

그림 1 에서 볼 수 있듯이, A3C 는 여러 A2C 에이전트들을 독립적으로 실행시키는 구조를 가지고 있으며, 이들은 글로벌 네트워크와 학습 결과를 주고받는다. 여러 에이전트들이 각자의 환경에서 독립적으로 행동하고, 보상을 받으며, 비평을 하는 방식으로 강화학습을 수행한다. 여러 에이전트를 사용함으로써 다양한 환경으로부터 얻은 풍부한 데이터를 바탕으로 학습을 진행할 수 있다는 장점이 있다.

3. 강화학습 기반 트래픽 로드 밸런서 제안

본 논문에서는 각 서비스별 트래픽에 대한 가중치를 결정하는 로드 밸런싱 메커니즘을 강화학습을 통해 제안한다. 이 메커니즘은 텔레메트리 기반의 트래픽 추적 모듈, 강화학습, 로드 밸런싱 컨트롤러로 구성되어 있다. 텔레메트리 기반의 트래픽 추적 단계에서는 텔레메트리 데이터를 활용하여 각 서비스가 수행한 작업의 정보를 수집하고, 이를 바탕으로 전체 시스템의 트래픽 패턴을 분석하며 문제점을 파악한다. MSA 환경에서는 서비스 간의 상호작용을 추적하는 것이 매우 중요하며, 이는 서비스의 의존성을 파악하고 성능의 병목 현상을 찾는 데에 큰 도움이 된다.

분산 추적을 위해서는 각 마이크로서비스에서 텔레메트리 데이터를 수집해야 한다. 이 데이터는 다양한 형태로 수집될 수 있고, 본 연구에서는 서비스 이름, 요청 URL, 요청 처리 시간, 응답 코드, 서비스 간의 호출 정보, 사용자 정보 등이 필요하다. 수집된 텔레메트리 데이터는 분산 추적을 수행하고 트래픽 패턴을 시각화하는 데 사용된다.

강화학습 단계에서는 수집된 트래픽 패턴 데이터를 A3C 알고리즘에 학습시킨다. 앞서 언급한 것처럼 A3C 는 여러 에이전트가 동시에 작동하는 멀티 에이전트 구조를 가지고 있다. A3C 에서 워커(worker) 에이전트는 글로벌 네트워크를 초기화한 뒤 환경과 상호작용하면서 학습을 시작한다. 각 워커는 서로 다른 탐험 정책(exploration policy)을 사용하여 최적의 정책(optimal policy)을 학습한다. 가치 손실(value loss, critic)과 정책 손실(policy loss, actor)을 계산하고, 이들 손실의 기울기(gradient)를 이용하여 글로벌 네트워크의 기울기를 업데이트한다. 워커가 글로벌 네트워크를 다시 초기화하고 동일한 학습 과정을 반복하면서 학습은 계속된다. 어드밴티지 함수(advantage function)의 정의는 다음과 같다[3].

$$A(s,a) = R - V(s)$$

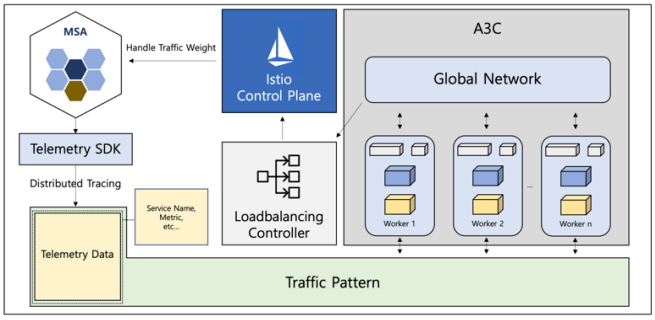
(수식 1) Definition of Advantage Function.

에이전트는 로드 밸런싱을 수행하는 주체로 각 마이크로서비스 인스턴스에 해당한다. 상태 공간에서 에이전트는 각 마이크로서비스의 현재 부하, CPU 및 메모리 사용률, 응답 시간과 같은 실시간 모니터링 지표와 최근 일정 시간 동안의 트래픽 패턴(예: 요청 수, 지연 시간)을 추적한다. 행동 공간에서 에이전트는 트래픽의 가중치를 조정하며, 가중치의 범위와 조정 단위를 설정한다. 보상 함수는 전체 시스템의 처리 시간 최소화, 각 서비스의 CPU 와 메모리 사용률의 균형 유지, 불필요한 트래픽 재분배 최소화 등을 목표로 설정한다.

네트워크 아키텍처에서는 Actor 네트워크가 트래픽 가중치 결정 정책을 학습하고, Critic 네트워크는 현재 상태의 가치 함수를 예측한다. 각 워커 에이전트는 비동기적으로 글로벌 네트워크에 자신의 학습 결과를 업데이트한다. 학습 절차는 워커 에이전트들이 독립적으로 환경과 상호작용하면서 경험을 수집하고, 이를 토대로 Actor 와 Critic 네트워크를 개별적으로 업데이트하는 것이다. 일정 주기마다 글로벌 네트워크를 업데이트하며, 탐험(Exploration)과 이용(Exploitation)의 균형을 맞추기 위한 정책도 사용된다.

강화학습에서 에이전트는 과거에 보상을 많이 받은 행동을 반복하는 경향이 있지만, 더 큰 보상을 위해 새로운 행동을 탐색하는 것이 필요하다. 이러한 탐색과 이용 사이의 균형이 중요하다. 에이전트의 행동은 트래픽을 처리하기 위해 목적지의 가중치를 결정하는 것이다.

로드 밸런싱 컨트롤러 단계에서는 앞서 강화학습 단계를 거친 에이전트들이 비동기적으로 글로벌 네트워크에 데이터를 업로드하고, 학습된 글로벌 네트워크를 바탕으로 각 마이크로서비스로부터 나가는 트래픽의 도착지 가중치를 조정한다. 그림 2 는 제안하는 시스템의 흐름도를 보여준다.



(그림 2) 시스템 흐름도

4. 실험 및 평가

트래픽 추적이 수행된 환경은 Kubeflow 이다. Kubeflow 는 쿠버네티스를 기반으로 한 머신러닝 플랫폼으로서, 마이크로서비스 아키텍처를 적용하여 Jupyter Notebook, Kubeflow Pipeline, Katib 등의 컴포넌트를 제공한다. 또한 Kubeflow 는 서비스 메시 기술인 Istio 와 밀접하게 통합되어 있어, Istio 의 트래픽 관리, 보안, 모니터링 기능을 활용하여 Kubeflow 컴포넌트 간의 통신을 제어하고 관리하는 데 유용하다.

본 연구에서는 Kubeflow 환경을 기반으로 Apache JMeter 스크립트를 사용하여 몇 분 간격으로, 일정 시간 동안 다양한 요청을 마이크로서비스에 보내어 트래픽 데이터를 생성하였다. Apache JMeter 는 Apache Project 에 속하는 소프트웨어 성능 테스트 도구로, 성능 테스트, 스트레스 테스트, 스파이크 테스트 등을 수행할 수 있으며, TCP, HTTP(S), FTP, JDBC, LDAP, SMTP 등 다양한 프로토콜을 지원한다.

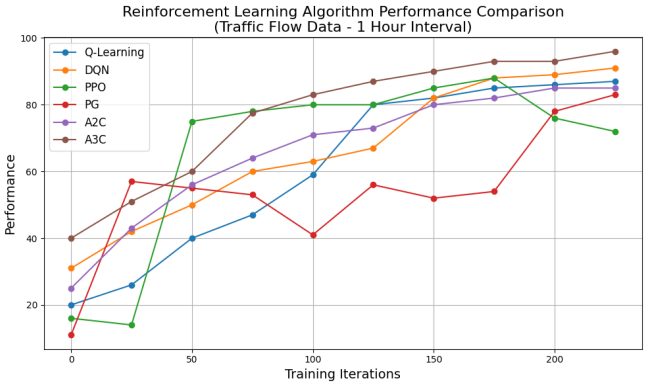
4-1. 분산 추적 수행

각 마이크로서비스에는 Telemetry SDK 가 설치되었으며, 서비스 코드에 트레이싱을 위한 설정이 추가되었다. 수집되는 텔레메트리 데이터에는 서비스의 이름, 요청 URL, 요청 처리 시간, 응답 코드, 서비스 간의 호출 정보, 사용자 정보 등이 포함된다. 이러한 데이터는 Jaeger 와 같은 분산 추적 시스템으로 전송되어, 시각화와 분석을 위해 사용된다. 이를 통해 서비스 간의 호출 관계, 시스템 내 병목 지점, 오류 발생 위치 등을 명확히 파악할 수 있었고, 시스템의 트래픽 흐름을 패턴화 하는 데에도 도움이 되었다.

4-2. A3C 모델 학습

분산 추적으로 수집된 트래픽 패턴 데이터는 A3C 모델을 학습하는 데 사용되었다. 상태 공간에는 각

마이크로서비스의 현재 부하, CPU 및 메모리 사용률, 응답 시간 지연, 그리고 최근 일정 시간 동안의 트래픽 패턴이 포함되었다. 행동 공간은 각 마이크로서비스로 전달되는 트래픽의 가중치를 조정하는 행위로 정의되며, 이때 가능한 가중치의 범위와 조정 단위를 설정한다. 보상 함수는 전체 시스템의 처리 시간 지연을 최소화하고, 각 서비스의 CPU 와 메모리 사용률을 균형 있게 유지하며, 불필요한 트래픽 재분배를 최소화하는 것을 목표로 설계되었다.



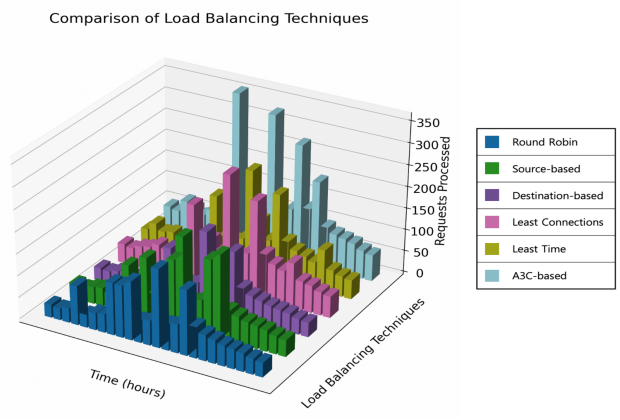
(그림 3) 트래픽 패턴에 따른 RL 알고리즘 성능 비교

그림 3 은 A3C 와 다른 강화학습 모델 간의 성능 비교를 보여준다. 비교된 강화학습 알고리즘은 Q-Learning, DQN, PPO, PG, A2C, A3C 이다. 실험 결과, A3C 모델은 뛰어난 학습 속도와 안정적인 수렴 특성을 보였다. 다양한 트래픽 패턴에 대해 높은 적응력을 보여주었고, 전체 시스템 성능을 효과적으로 향상시킬 수 있었다. 특히, Stress 가 발생하는 트래픽 상황에서도 강인한 성능을 보였다.

4-3. 로드 밸런싱 컨트롤러 구현

A3C 모델의 학습이 완료된 후, 학습된 글로벌 네트워크를 로드 밸런싱 컨트롤러에 적용하였다. 이 컨트롤러는 각 마이크로 서비스의 실시간 상태 정보를 받아들여 글로벌 네트워크의 정책에 따라 트래픽 분배 비율을 결정한다. 이렇게 결정된 가중치는 Istio 와 같은 서비스 메시 플랫폼을 통해 마이크로 서비스로 전달되며, 서비스 메시는 이를 기반으로 트래픽을 동적으로 라우팅한다.

그림 4 는 임의의 서비스 메시 환경에서 트래픽 통신이 이루어지는 가정하에 각 기법 별로 비교한 그래프이다. 이 환경에서는 일반적인 상황과 요청이 많은 시간대를 분리하여 트래픽을 임의 조정하였다. 그래프에서는 초당 처리량의 시간별 평균을 내어 기법 간 처리 성능을 시각화 하였다. 제안된 A3C 기반 로드 밸런싱 기법이 모든 상황에서 다른 기법들보다 뛰어난 성능을 보인 것은 아니지만, 학습이 주로 진행된 stressful 한 상황에서는 우수한 성능을 보여준다.



(그림 4) 로드 밸런싱 별 트래픽 처리량 비교

5. 결론

본 연구에서는 A3C 강화학습 기반의 적응형 로드 밸런싱 기법을 제안하였다. 제안된 기법은 분산 추적을 통해 수집된 실시간 트래픽 데이터를 기반으로 최적의 로드 밸런싱 정책을 학습하고 적용함으로써 시스템의 성능과 안정성을 향상시킬 수 있음을 입증하였다. 그러나 본 연구는 특정 MSA 환경에 국한되어 수행되었으며, 실험에 사용된 트래픽 데이터의 다양성이 제한적이었다는 한계가 있다. 향후에는 다양한 규모와 특성의 마이크로 서비스 환경에서의 검증이 필요하며, 모델의 경량화와 적응형 하이퍼 파라미터 튜닝 등의 최적화 방안에 대한 연구가 지속되어야 할 것이다. 또한, 제안된 기법과 다른 기법과의 비교, 그리고 실제 운영 환경에서의 장기적인 성능 평가 등도 향후 연구의 주요 주제가 될 것으로 보인다.

Acknowledgement

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 학석사연계 ICT 핵심인재양성사업의 연구결과로 수행되었음 (IITP-2024-RS-2023-00259867)

참고문헌

[1] Sigelman, Benjamin H., et al. "Dapper, a Large-scale Distributed Systems Tracing Infrastructure", Google Technical Report, 2010, pp. 1-14

[2] A. Khatri and S. Mathi, "Active Home Agent Load Balancing for Next Generation IP Mobility based Distributed Networks", International Conference on Ubiquitous Communications and Network Computing, 2017, pp 165-176

[3] Volodymyr Mnih, Adrià Puigdomènech Badia, et al. "Asynchronous Methods for Deep Reinforcement Learning", International Conference on International Conference on International Conference on Machine Learning, 2016, pp 1928-1937