Journal of KIIT. Vol. 22, No. 3, pp. 41-49, Mar. 31, 2024. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2024.22.3.41

MSA 환경에서 스트림 데이터의 이상치 탐지를 위한 쉥글 사이즈 기반 RRCF 비교분석

최수빈*[1], 이승규*[2], 이주찬*[3], 윤건용*[4], 양단아**

Comparative Analysis of Shingle Size-based RRCF for Anomaly Detection in Stream Data within MSA Environments

Subeen Choi*[1], Seunggyu Lee*[2], Juchan Lee*[3], Geonyong Yoon*[4], and Dana Yang**

요 약

최근 들어 MSA(Microservices Architecture)시장은 빠르게 성장하고 있다. MSA 환경은 각각의 서비스들이 분산되어 독립적으로 구성되어 있기 때문에 서비스의 다운타임(Downtime)을 최소화하기 위해 이상치를 조기에 탐지하는 것은 필수적이다. 그러나 MSA 환경에서 수집하는 메트릭 데이터(Metric data)는 빠르고 변동성이 높은 스트림 데이터(Stream data) 형태이기 때문에 이상치 탐지 난도가 높다. 따라서 본 논문에서는 스트림 데이터의 이상치 탐지에 강한 RRCF 알고리즘을 선택하고, 주요 파라미터인 쉥글(Shingle)의 크기를 조절하여 RRCF 알고리즘이 IF(Isolation Forest), ARIMA(AutoRegressive Integrated Moving Average)와 같은 기존 알고리즘보다 MSA 환경에서 부하 테스트를 통해 수집된 스트림 데이터의 이상치를 효과적으로 탐지할 수 있다는 것을 검증했다.

Abstract

Microservices Architecture(MSA) market has been experiencing rapid growth recently. Since each service in the MSA environment is distributed and independently configured, early detection of anomaly traffic is essential to minimize service downtime. However, some challenges in MSA environments are difficult, due to the nature of the collected metric data which is characterized as fast and highly volatile stream data. Therefore, this paper adjusts the size of the main parameter as Shingle of the RRCF algorithm and verifies that the RRCF algorithm can effectively detect anomalies in stream data, compared with the traditional algorithms such as Isolation Forest(IF) and AutoRegressive Integrated Moving Average(ARIMA), through load testing of MSA.

Keywords

anomaly detection, microservices architecture, rrcf, shingle size, isolation forest, arima, stream data

* 한국성서대학교 컴퓨터소프트웨어학과
- OCRID[1]: https://orcid.org/0009-0002-3673-0913
- OCRID[2]: https://orcid.org/0009-0006-2684-8861
- OCRID[3]: https://orcid.org/0009-0008-2280-4057
- OCRID[4]: https://orcid.org/0009-0009-3248-2933
** 한국성서대학교 컴퓨터소프트웨어학과 조교수(교신저자)
- OCRID: https://orcid.org/0000-0001-6463-9784

· Received: Jan 03, 2024, Revised: Mar. 14, 2024, Accepted: Mar. 17, 2024

· Corresponding Author: Dana Yang
Dept. of Computer Software, Korean Bible University 32, Dongil-ro 214-gil, Nowon-gu, Seoul, Republic of Korea
Tel.: +82-2-950-5487, Email: dana1112@bible.ac.kr

Ⅰ. 서 론

MSA(Microservices Architecture)는 단일 애플리케이션을 작고 독립적인 서비스들로 구성하는 아키텍처 스타일이다. 비즈니스 기능 중심으로 분리된 서비스들은 서비스 간의 통신을 위해 경량 통신 프로토콜을 사용한다. MSA는 기존 애플리케이션의 복잡성을 해결하고, 뛰어난 독립성과 확장성을 제공했기 때문에 소프트웨어 개발 트렌드 중 하나로 부상했다[1]. 또한 MSA 시장은 전 세계적으로 성장하고 있으며, 2023년에서 2035년까지 연간 약 16.17%씩 지속적으로 성장할 것으로 예측된다[2].

각각의 서비스들이 분산되어 독립적으로 구성되어있는 MSA 환경에서 서비스의 다운타임(Downtime)을 최소화하여 이용자에게 안정적인 환경을 제공하기 위해서는 지속적인 모니터링과 함께 신속하게 이상 패턴을 찾는 이상치 탐지(Anomaly detection)가 중요하다고 할 수 있다[3][4]. 또한 MSA 환경에서 수집하는 메트릭 데이터는 빠르고 변동성이 높은 스트림 데이터(Stream data) 형태이기 때문에 고정되어있는 배치 데이터(Batch data)의 이상치를 탐지하는 것에 비해 탐지 난도가 높다는 특성이 있다[5]. 따라서 MSA 환경에서 모니터링되는 스트림 데이터의 이상치를 탐지하기 위해서는 이상치 탐지 속도가 빠르고, 안정적이고, 데이터의 최근 경향을 잘 반영할 수 있는 모델이 요구된다.

그러므로 본 논문에서는 위와 같은 문제점을 해결하기 위해 아래와 같이 제안한다.

- 실제 MSA 환경과 유사한 MSA 기반의 웹 앱을 구축하고 메트릭 데이터를 수집하여 현실적인 데이터셋을 구성한다.
- 스트림 데이터의 이상치 탐지에 강한 RRCF(Robust Random Cut Forest) 알고리즘의 쉉글(Shingle) 파라미터를 조절하여 이상치 탐지 성능을 최적화한 이상치 탐지 모델을 생성한다.
- 쉉글 파라미터를 조절한 RRCF 알고리즘을 다른 알고리즘들과 비교분석하여 스트림 데이터 환경에서 RRCF 알고리즘의 이상치 탐지 성능을 확인한다.

본 논문에서는 앞서 제안한 것처럼 이상치 탐지 알고리즘인 RRCF 알고리즘을 선택하여 RRCF의 쉉글 파라미터 조절이 RRCF 알고리즘으로 생성한 이상치 탐지 모델의 정확도에 얼마나 영향을 미치는지 평가하려고 한다.

이에 따른 논문 구성은 다음과 같다. 2장에서는 MSA, ARIMA(AutoRegressive Integrated Moving Average), IF(Isolation Forest), RRCF에 대한 관련 연구에 대해 서술한다. 3장에서는 전체 시스템 구조와 데이터 플로우에 대해 서술하고, RRCF의 쉉글 파라미터의 조절에 따른 이상치 탐지 성능의 변화를 확인한다. 4장에서는 쉉글 파라미터를 조절한 RRCF 알고리즘을 기존 알고리즘인 ARIMA, IF와 비교하고 분석하여 MSA 환경에서 수집된 메트릭 데이터에 대해 이상치 탐지 성능을 확인한다. 마지막 5장에서는 연구 내용 요약 및 기대효과에 대해 설명한다.

Ⅱ. 관련 연구

2.1 Microservices architecture

MSA[6]는 각 서비스를 독립적으로 확장할 수 있는 높은 확장성을 가지고 있기 때문에 서비스의 수요가 증가할 때 빠르게 대응하며, 필요한 서비스만 확장하여 리소스를 효율적으로 사용하고 높은 성능을 달성한다[7]. 또한, MSA는 각 서비스가 독립적으로 실행되어 한 서비스에 장애가 발생하더라도 다른 서비스로 전파되지 않아, 서비스 중 한 곳에 문제가 생겨도 전체 애플리케이션은 지속적으로 운영될 수 있기 때문에 높은 가용성을 보장한다. 이와 같은 MSA의 특성들을 고려할 때, 분산 환경에서 서비스의 장애를 실시간으로 감지하고 대응하기 위한 모니터링은 필수적이다. 실시간 모니터링은 이상과 장애를 빠르게 감지하여 조치할 수 있도록 하며, 이를 통해 MSA의 안정성과 신뢰성을 유지할 수 있다.

따라서 본 논문은 MSA의 안정성과 신뢰성을 유지하기 위해 MSA 기반의 HTTP 웹 앱을 구축하고, 수집된 스트림 데이터 형태의 메트릭 데이터에 대한 이상치 탐지 최적화를 진행하려고 한다.

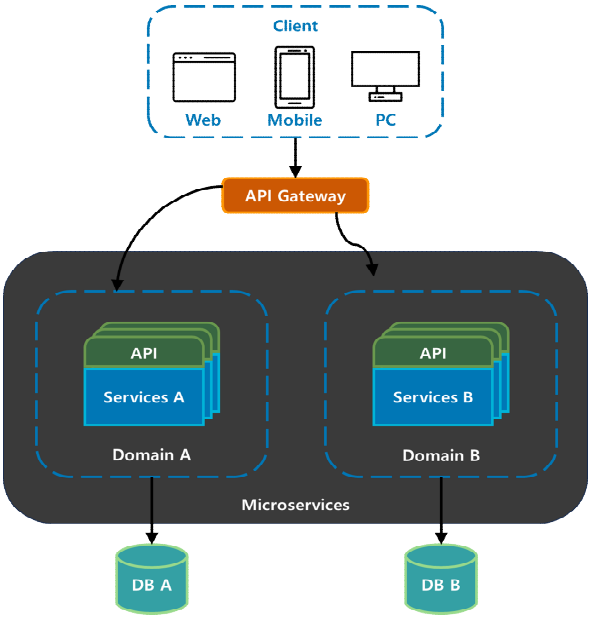

그림 1. MSA 구조
Fig. 1. Structure of MSA

2.2 AutoRegressive integrated moving average

ARIMA는 시계열 데이터가 자기회귀 특성을 가지고 있다는 가정을 바탕으로, 과거의 데이터로 미래의 데이터를 예측하는 알고리즘이다. ARIMA는 자기회귀(AR), 차분(I), 이동 평균(MA)의 3가지 요소로 구성되어 메트릭 데이터의 이상치를 탐지하게 된다[8]. ARIMA는 메트릭 데이터의 경향성(Trend)과 계절성(Seasonality)을 분석하여 높은 정확도로 미래의 데이터를 예측할 수 있기 때문에 널리 사용되고 있다[9][10]. 그러나 ARIMA는 기본적으로 안정적인 패턴을 가정하고 있기 때문에, 스트림 데이터의 갑작스럽고 동적인 패턴을 예측하기가 어려울 수 있다. 이런 제약을 극복하고자 스트림 데이터의 갑작스러운 패턴에 더 효과적으로 대응할 수 있는 RRCF를 기반으로 파라미터를 최적화하여 연구를 진행하려고 한다.

2.3 Isolation forest

IF는 이상 데이터가 정상 데이터에 비해 쉽게 고립될 것이라는 가정을 기반으로 하는 이상치 탐지 알고리즘이다. IF는 데이터셋에서 무작위로 가져온 데이터 포인트로 여러 개의 이진 트리를 제작하고, 데이터 포인트가 얼마나 적은 분할로 트리에서 고립되는지를 측정하여 이상치를 탐지한다[11][12].

이런 IF의 측정 방식은 대량의 배치 데이터 세트에서 빠르고 효율적으로 이상치를 탐지할 수 있게 한다. 하지만 IF는 모델의 동적인 업데이트가 어렵기 때문에 변동성이 높은 스트림 데이터에 취약할 수 있다는 단점을 가지고 있다. 이러한 점을 고려하여 모델의 동적인 업데이트에 유리한 RRCF의 연구를 진행하였다.

2.4 Robust random cut forest

RRCF는 널리 알려진 이상치 탐지 알고리즘인 IF의 단점을 보완한 이상치 탐지 알고리즘이다[13]. RRCF는 이상 데이터가 정상 데이터에 비해 쉽게 고립될 것이라는 가정에 기반을 두고 있다는 특징을 IF와 공유한다. 하지만 RRCF는 트리 구축 과정에서 노드를 무작위로 선택할 때 데이터 포인트의 분포를 반영하고, 이상치 탐지 과정에서 데이터 포인트를 트리에 남겨 최근 경향을 반영한다는 중요한 차별점이 있다[14].

RRCF가 이상치를 탐지하는 과정은 그림 2와 같이 크게 4단계로 정의할 수 있다. 첫 번째 단계는 데이터를 입력하는 것이다. 모델 생성에 사용할 데이터의 변수를 선택하고 전처리를 수행한 뒤, 데이터를 입력한다. 두 번째 단계는 데이터를 선택하는 것이다.

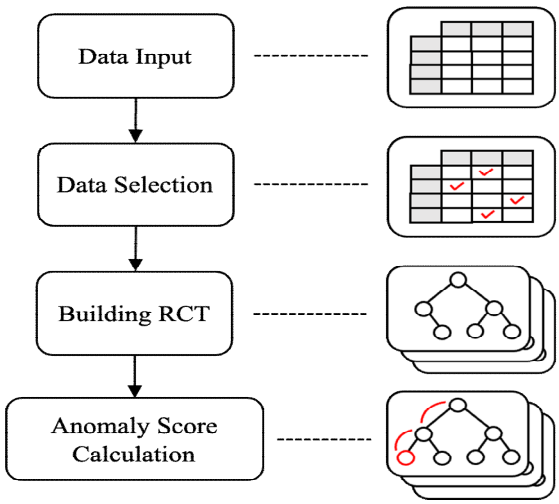

그림 2. RRCF의 탐지과정
Fig. 2. Detection process of RRCF

RRCF 알고리즘은 데이터 포인트들의 전체적인 분포를 반영하여 RCT(Random Cut Tree)를 구성하는 노드를 무작위로 선택한다. 세 번째 단계는 RCT를 구축하는 것이다. 사전에 지정된 개수만큼 이진 트리 구조로 이루어진 RCT를 구축하고, 여러 개의 RCT로 이루어진 이상치 탐지 모델을 생성한다. 네 번째 단계는 이상치 점수를 계산하는 것이다. 모델을 생성한 후, 각각의 RCT마다 그림 2에서 빨간색으로 표시된 노드와 같이 데이터 포인트를 삽입하고 데이터 포인트가 RCT의 루트(Root)로부터 얼마나 멀리 떨어져 있는지 계산한다. 데이터 포인트가 루트와 가까울수록 해당 데이터 포인트는 쉽게 고립된 것이기 때문에 이상치일 확률이 높다.

IF와 대조되는 RRCF의 특징 중 하나는 트리에 데이터 포인트를 삽입한다는 것이다. RRCF는 이상치를 탐지하기 위해 트리에 데이터 포인트를 추가하고, 트리가 가득 찼다면 오래된 데이터 포인트부터 삭제할 수 있기 때문에 스트림 데이터 환경에서 최근의 경향을 잘 반영할 수 있다. 이 특징으로 인해 RRCF는 IF보다 스트림 데이터의 이상치 탐지에 적합한 알고리즘으로 개선됐다. 또한 RRCF의 파라미터 중 하나인 윈도를 조정하는 것은 RRCF의 성능에 큰 영향을 미친다고 할 수 있다. 따라서 본 논문에서는 RRCF의 여러 파라미터 중, 특히 윈도 크기의 조정을 통해 이상치 탐지 모델의 탐지 성능을 크게 향상시킬 수 있다는 것을 실험으로 확인했다. 결론적으로 본 논문은 RRCF 알고리즘을 채택하여 MSA 환경의 시계열 데이터에 대한 이상치 탐지를 진행하려고 한다.

III. 시스템 제안

3.1 전체 시스템

그림 3은 본 논문의 전체 시스템 구조도이다. 본 논문은 구글 클라우드 플랫폼(Google Cloud Platform)의 인프라를 사용하여 컨테이너화된 애플리케이션을 대규모로 배포 및 운영하는 데 사용할 수 있는 관리형 쿠버네티스(Kubernetes) 서비스 GKE(Google Kubernetes Engine)[15]에 MSA 기반의 웹 앱을 배포하여 테스트 환경을 구성한다.

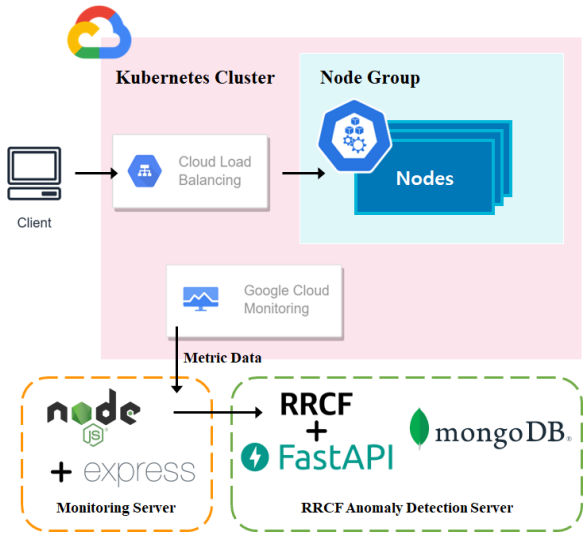

그림 3. 전체 시스템 구조
Fig. 3. Entire system architecture

위와 같이 구성된 테스트 환경에 부하테스트를 진행하여 발생한 메트릭 데이터로 이상치 탐지를 하는 것이 본 논문의 목적이다. GKE에 배포한 MSA 기반의 HTTP 웹 앱인 Bank of Anthos는 사용자가 인공 은행 계좌를 만들고 거래를 완료할 수 있도록 은행의 결제 처리 네트워크를 시뮬레이션하는 애플리케이션이다. 그리고 Node.js의 Express 기반 모니터링 서버는 MSA 기반의 웹 앱에서 메트릭 데이터를 수집한 뒤 이상치 탐지 서버에 전달한다. FastAPI로 구축된 이상치 탐지 서버는 전달받은 메트릭 데이터를 RRCF 알고리즘으로 생성된 이상치 탐지 모델을 통해 이상치 탐지를 진행한다.

3.2 데이터 플로우

그림 4는 전체 시스템의 데이터 플로우를 나타낸다. 스택드라이버 에이전트(Stackdriver agent)는 GKE 클러스터 노드들의 측정항목을 주기적으로 메트릭 데이터로 수집하고(Pull), 이를 구글 클라우드 모니터링(Google cloud monitoring)으로 전송한다. 수집된 메트릭 데이터는 구글 클라우드 모니터링 API를 통해 접근이 가능하며, Node.js 서버는 구글 클라우드 모니터링 API를 호출하여 받아온 메트릭 데이터를 필요한 정보와 함께 JSON 형식으로 전처리한다. 그리고 Node.js 서버는 전처리된 데이터를 REST API를 통해 FastAPI 이상치 탐지 서버로 전송한다(POST).

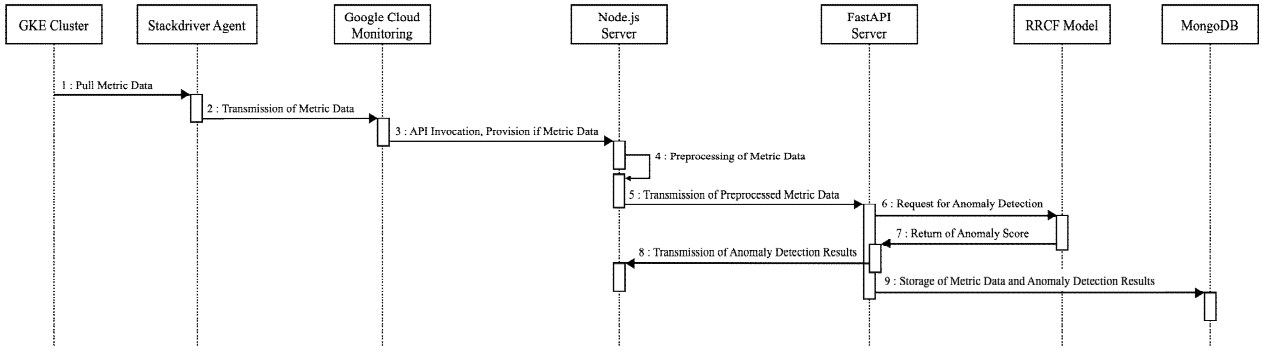

그림 4. 전체 시스템의 데이터 플로우
Fig. 4. Data flow of the entire system

FastAPI 서버는 수신한 전처리된 메트릭 데이터를 RRCF Model로 분석하여 이상치를 탐지하고, 이상치 점수를 결과로 반환한다. 이 반환된 이상치 탐지 결과는 FastAPI에 의해 Node.js 서버로 다시 전송되며, 그 과정에서 모든 메트릭 데이터는 MongoDB에 JSON 형식으로 저장된다.

3.3 RRCF 기반 최적값 도출

RRCF알고리즘을 사용하여 모델을 생성할 때의 주요 파라미터로는 트리(Tree)의 크기, 트리의 개수, 쉥글(Shingle)의 크기가 있다. 트리의 크기와 개수는 클수록 모델이 더 많은 데이터 포인트를 학습하게 되어 이상치 탐지 성능이 향상되지만, 탐지 속도가 증가하게 된다. rrcf의 파라미터 중에서 모델의 성능에 가장 크게 영향을 끼치는 파라미터는 쉥글의 크기이다. 쉥글의 크기를 정하는 과정은 다음과 같이 표현될 수 있다.

$$S(i) = [d_{i-n+1}, d_{i-n+2}, \ldots, d_i] \quad (1)$$

여기서 S(i)는 i번째 데이터 포인트로 생성되는 쉥글을 나타내고, di는 i번째 데이터 포인트를 의미한다. 그림 5에서 보이는 것과 같이 쉥글의 크기가 적정 수준으로 조정되면 이상치 탐지 모델의 f1 score가 크게 상승하는 것을 확인할 수 있다. 본 논문이 사용한 데이터셋의 경우에는 쉥글 크기가 13에 도달할 때까지 f1 score는 점점 증가하는 모습을 보였으며, 14부터는 오히려 f1 score가 감소하는 현상을 관측할 수 있었다. 이를 통해 쉥글의 크기를 데이터셋에 적합하게 조절한다면 이상치 탐지 모델의 성능이 효과적으로 증가하지만, 단순히 쉥글의 크기를 키우는 것은 모델이 오히려 지나치게 오래된 데이터를 고려하게 되어 이상치 탐지 성능이 낮아지는 것을 알 수 있다. 따라서 본 논문에서는 시뮬레이션된 시스템에서 최적의 성능을 보인 쉥글 크기 13을 기반으로 실험을 진행하고자 한다.

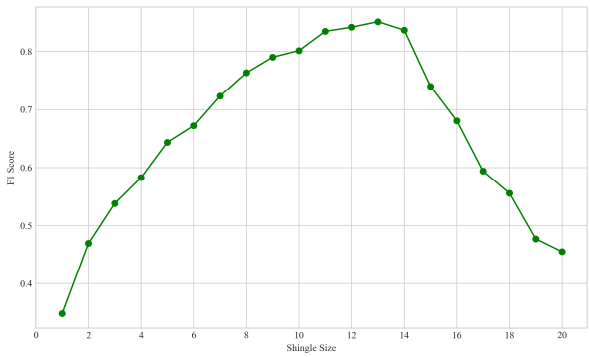

그림 5. 쉥글 사이즈에 따른 F1 score
Fig. 5. F1 score based on shingle size

IV. 평 가

4.1 Jmeter E2E 테스트

본 논문에서는 JMeter로 테스트를 진행하였으며, 웹 어플리케이션에서 실제 사용자가 시스템을 사용하는 방식을 모방하여 테스트 케이스를 작성하였다. 사용자가 회원가입을 한 뒤, 회원 가입한 계정으로 로그인을 한다.

사용자는 최초 계좌 연동을 통해 계좌에 일정 금액을 입금하고 임의의 액수를 타 계좌에 송금한다. 사용자는 송금이 마무리되면 로그아웃한다. 작성된 테스트 케이스 순으로 테스트를 진행하였다. 20개의 Thread에서 지속적으로 MSA 서버에 접근하여 부하를 생성하였다.

그림 6은 JMeter 대시보드의 Latency와 Request를 비교하는 그래프로써 응답 지연 시간(Latency)과 요청(Request) 간의 관계를 시각적으로 나타내는 그래프이다. X축은 초당 전체 요청수(Global number of requests per second)이고, Y축은 각 요청의 중앙값 대기 시간이며(Median latency in ms) 단위는 ms이다. 이는 요청이 처리되기 시작한 시점부터 응답이 완료될 때까지의 시간을 나타낸다. 그래프를 분석한 결과 요청 수가 증가함에 따라 대기 시간의 중앙값이 전반적으로 감소하는 경향이 있다. 이는 시스템이 더 많은 요청을 처리할 때 효율적으로 작동하고 있음을 알 수 있다. 하지만 일정 요청 수를 초과하면 대기 시간이 다시 증가하기 시작하는 경향이 보인다. 이는 시스템이 포화 상태에 이르렀거나 다른 병목 현상이 발생했음을 알 수 있다.

이 그래프를 통해 시스템이 특정 부하 수준에서 최적으로 작동한다는 것을 알 수 있었다. 특정 부하 수준을 넘는 경우 성능이 저하될 수 있다.

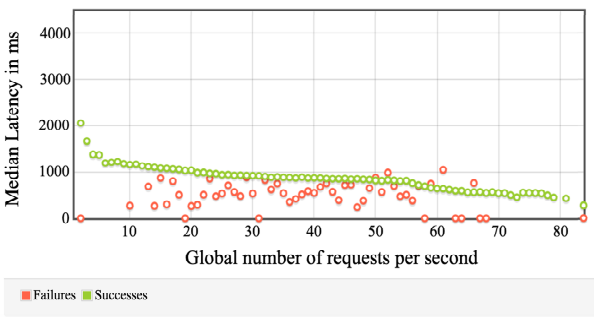

그림 6. JMeter 대시보드
Fig. 6. JMeter dashbord

4.2 알고리즘별 성능 평가

이전에 구성한 MSA 환경의 부하 테스트를 진행하여 수집한 스트림 데이터 형태의 서버 메트릭 데이터를 기반으로 ARIMA, IF, RRCF 알고리즘을 사용해 이상치 탐지 성능을 분석하였다. 그림 7에서 볼 수 있듯이 ARIMA, IF, RRCF 모두 이상치 구간에서 이상치 점수(Anomaly Score)가 뚜렷하게 상승하는 모습을 보인다. 하지만 정상 구간에서의 안정성은 각각의 알고리즘마다 명확한 차이가 있는 것을 확인할 수 있었다. RRCF는 IF 대비 정상 구간에서 약 70% 더 안정적인 모습을 보였고, ARIMA 대비 정상 구간에서 약 20% 더 안정적인 모습을 보였다.

표 1은 그림 7과 같이 본 논문에서 구성한 MSA 환경의 부하 테스트를 통해 수집한 메트릭 데이터에 대해 ARIMA, IF, RRCF 알고리즘들로 이상치 탐지를 진행한 후 산출된 성능지표를 나타낸다. 우선, 본 논문에서는 그림 5에서 확인한 것과 같이 RRCF 알고리즘에서 최적의 성능을 보인 슝글 크기 13으로 설정했다. 실험 결과 RRCF는 IF 대비 이상치를 메트릭 수집 간격 1개(60초) 빠르게 탐지했고, ARIMA 대비 메트릭 수집 간격 2개(120초) 빠르게 탐지했다. 또한 RRCF는 이상치 탐지가 까다로운 스트림 데이터 형태의 서버 메트릭 데이터에 대해 다른 두 알고리즘보다 F1 sore, Recall Precision같은 주요 성능 지표에 있어 우수한 결과를 나타냈다. 따라서 MSA 환경에서 생성된 스트림 데이터의 이상치를 탐지하는 메커니즘에서 RRCF가 다른 두 알고리즘인 ARIMA, IF보다 뛰어난 성능을 보인다는 것을 확인할 수 있었다.

표 1. 알고리즘별 성능지표
Table 1. Performance metrics by algorithm

Performance metrics \ Algorithm	ARIMA	IF	RRCF
F1 score	0.32	0.38	0.88
Recall	0.21	0.31	0.82
Precision	0.68	0.48	0.95
Delay in anomaly detection interval	2	1	0

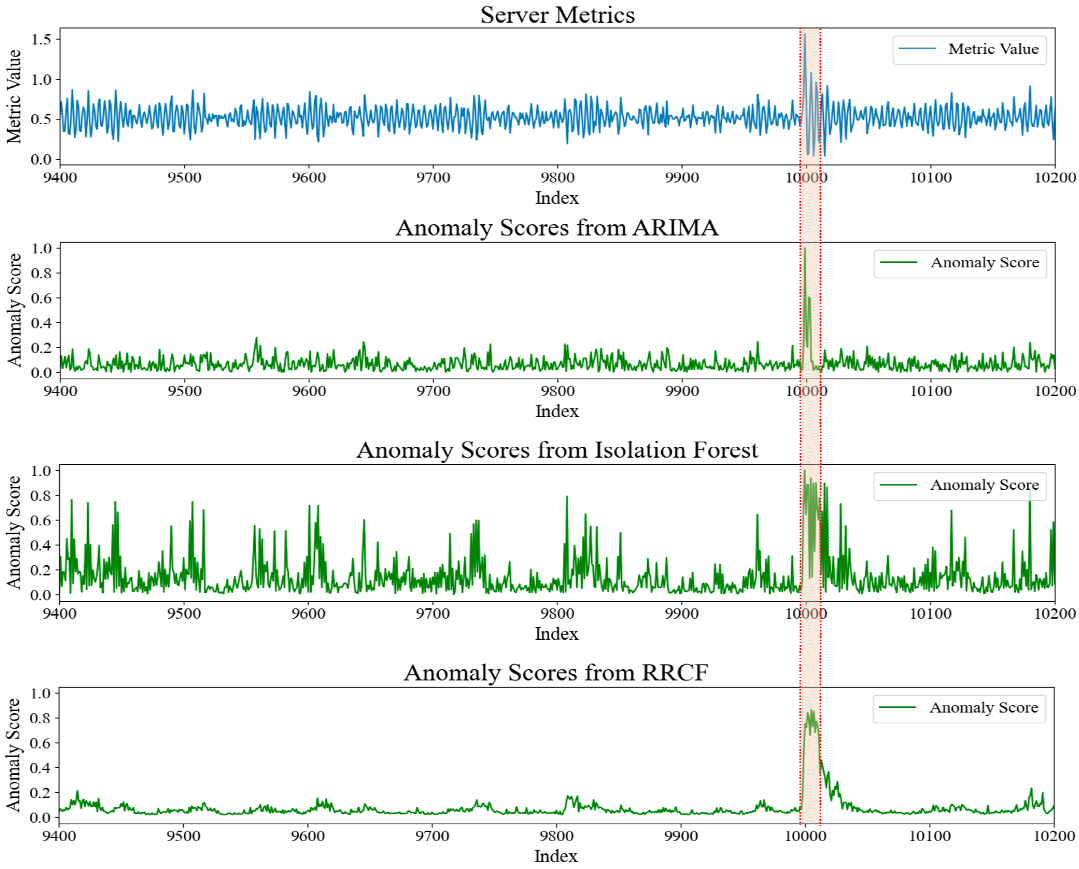

그림 7. 알고리즘별 이상치 탐지 그래프
Fig. 7. Anomaly detection graph by algorithm

V. 결 론

본 논문에서는 MSA로 구성된 환경에서 실제 서비스 환경과 유사한 방식으로 부하 테스트를 진행하여 서버의 메트릭 데이터를 수집하고, RRCF의 파라미터 중 쉉글의 크기에 주목하여 이상치 탐지를 수행하였다. 실험을 진행한 결과, RRCF는 MSA 환경에서 수집된 메트릭 데이터의 이상치 탐지 문제에서 기존의 알고리즘인 ARIMA, IF보다 우수한 성능을 보여주었다. 또한 RRCF 알고리즘으로 이상치 탐지 모델을 생성할 때, 쉉글 파라미터의 조절을 통해 모델의 성능을 효과적으로 향상시킬 수 있다는 것을 확인했다.

또한 쉉글 파라미터를 최적화한 RRCF 알고리즘을 통해 MSA 환경에서 이상치 탐지를 진행하고, 서비스의 다운타임을 최소화하여 이용자에게 안정적인 환경을 제공할 수 있을 것이라고 기대된다[16].

References

[1] P. Zaragoza, A.-D. Seriai, A. Seriai, A. Shatnawi, and M. Derras, "Leveraging the Layered Architecture for Microservice Recovery", 2022 IEEE 19th International Conference on Software Architecture (ICSA), Honolulu, HI, USA, pp. 135-145, Mar. 2022. https://doi.org/10.1109/ICSA53651.2022.00021.

[2] SDKI, "Microservice Architecture Market by Deployment (Cloud-based and On-premises), Service (Inventory, Accounting, Shipping, and Store), End User, and Region - Forecast from 2023 to 2035", https://www.sdki.jp/reports/microservice-architecture-market/107266#market_snapshot [accessed: Nov. 23, 2023]

[3] S. H. Jeon, C. L. Park, G. W. Lee, S. J. Kim, and B. G. Gu, "Threshold Determination Method in Anomaly Detection using LSTM Autoencoder", The Journal of Korean Institute of Information Technology, Vol. 21, No. 4, pp. 21-30, Apr. 2023. https://dx.doi.org/10.14801/jkiit.2023.21.4.21.

[4] Y. Song, R. Xin, P. Chen, R. Zhang, J. Chen, and Z. Zhao, "Autonomous selection of the fault classification models for diagnosing microservice applications", Future Generation Computer Systems, Vol. 153, pp. 326-339, Mar. 2024. https://doi.org/10.1016/j.future.2023.12.005.

[5] C. H. Park, "Anomaly Pattern Detection on Data Streams", 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), Shanghai, China, pp. 689-692, Jan. 2018. https://doi.org/10.1109/BigComp.2018.00127.

[6] L. D. Lauretis, "From Monolithic Architecture to Microservices Architecture", 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Berlin, Germany, pp. 93-96, Oct. 2019. https://doi.org/10.1109/ISSREW.2019.00050.

[7] F. Tusa, S. Clayman, A. Buzachis, and M. Fazio, "Microservices and serverless functions—lifecycle, performance, and resource utilisation of edge based real-time IoT analytics", Future Generation Computer Systems, Vol. 155, pp. 204-218, Jun. 2024. https://doi.org/10.1016/j.future.2024.02.006.

[8] D. Barrientos-Torres, E. A. Martinez-Ríos, S. A. Navarro-Tuch, J. L. Pablos-Hach, and R. Bustamante-Bello, "Water Flow Modeling and Forecast in a Water Branch of Mexico City through ARIMA and Transfer Function Models for Anomaly Detection", Water 2023, Vol. 15, No. 15, pp. 2792, Aug. 2023. https://doi.org/10.3390/w15152792.

[9] Z. Luo, Y. Zhang, C. Yin, M. Yang, and J. Li, "Application of ARIMA Model in Infectious Disease Prediction", 2023 5th International Conference on Decision Science & Management (ICDSM), Changsha, China, pp. 3-6, Mar. 2023. https://doi.org/10.1109/ICDSM59373.2023.00012.

[10] F. Zhao, R. Sun, X. Chen, K. Zhang, and S. Han, "Flight Incidents Prediction Based on Model of X-12 and ARIMA", 2019 5th International Conference on Transportation Information and Safety (ICTIS), Liverpool, UK, pp. 855-860, Jul. 2019. https://doi.org/10.1109/ICTIS.2019.8883751.

[11] J. Wang, J. Cao, and Z. Liu, "Unsupervised machine learning-based multi-attributes fusion dim spot subtle sandstone reservoirs identification utilizing isolation forest", Geoenergy Science and Engineering, Vol. 234, pp. 212626, Mar. 2024. https://doi.org/10.1016/j.geoen.2023.212626.

[12] F. Feng, Z. Liu, and J. Zhang, "Detection of GPS Abnormal Data of Sanitation Vehicles Based on Isolation Forest Algorithm", 2022 International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI), Zakopane, Poland, pp. 460-463, Dec. 2022. https://doi.org/10.1109/ICDACAI57211.2022.00097.

[13] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams", Proceedings of The 33rd International Conference on Machine Learning, New York NY USA, Vol. 48, pp. 2712-2721, Jun. 2016.

[14] Z. Pang, J. Cen, and M. Yi, "Unsupervised concept drift detection method based on robust random cut forest", International Journal of Machine Learning and Cybernetics, Vol. 14, No. 12, pp. 4207-4222, Jun. 2023. https://doi.org/10.1007/s13042-023-01890-x.

[15] Google Cloud, "GKE Overview", https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview?hl=ko [accessed: Nov. 17, 2023]

[16] L. Yufeng, Y. Guangba, C. Pengfei, Z. Chuanfu, and Z. Zibin, "MicroSketch: Lightweight and Adaptive Sketch Based Performance Issue Detection And Localization in Microservice Systems", 20th International Conference on Service-Oriented Computing (ICSOC), Sevilla, Spain, pp. 219-236, Nov. 2022. https://doi.org/10.1007/978-3-031-20984-0_15.

저자소개

최 수 빈 (Subeen Choi)

2018년 3월 ~ 현재 :
한국성서대학교
컴퓨터소프트웨어학과 학사과정
관심분야 : 이상치 탐지, 인공지능, 머신러닝

이 승 규 (Seunggyu Lee)

2018년 3월 ~ 현재 :
한국성서대학교
컴퓨터소프트웨어학과 학사과정
관심분야 : DB, 서버

이 주 찬 (Juchan Lee)

2021년 3월 ~ 현재 :
한국성서대학교
컴퓨터소프트웨어학과 학사과정
관심분야 : 머신러닝, 딥러닝, 데이터 분석

윤 건 용 (Geonyong Yoon)

2019년 3월 ~ 현재 :
한국성서대학교
컴퓨터소프트웨어학과 학사과정
관심분야 : 딥러닝, 데이터 처리

양 단 아 (Dana Yang)

2014년 8월 : 한국성서대학교
컴퓨터소프트웨어학과(공학사)
2023년 8월 : 이화여자대학교
컴퓨터공학(석.박사통합)
2023년 ~ 현재 : 한국성서대학교
컴퓨터소프트웨어학과 조교수
관심분야 : 블록체인, 머신러닝, 딥러닝, 네트워크 보안