논문 19-44-03-13 The Journal of Korean Institute of Communications and Information Sciences '19-03 Vol.44 No.03
https://doi.org/10.7840/kics.2019.44.3.554

엣지 클라우드 시스템에서 장애 원인 분석 설계 및 구현

김 민 욱◆, 김 영 한°

Design and Implementation of Root Cause Analysis in Edge Cloud System

Min-wook Kim◆, Young-han Kim°

요 약

가입자에 근접한 위치에서 서비스를 제공하는 엣지 클라우드 시스템의 장애 원인 분석을 효율적으로 수행하기 위한 체계가 필요하다. 기존의 단일 클라우드 시스템에 적용하던 방식을 중앙 클라우드에 적용하고, 이를 통해 원격지 다수의 엣지 클라우드 시스템의 장애 원인 분석에 적용할 경우, 다수의 개별 엣지 클라우드 내의 각 구성요소로부터의 이벤트 보고용 트래픽이 중앙 클라우드로 과도하게 증가할 수 있어, 실시간 장애보고에 문제가 발생하게 된다. 본 논문에서는 각 엣지 클라우드의 리소스 상태는 각 엣지에서 로컬하게 수집하고, 상호 연계된 정보로 취합한 후 중앙 클라우드로 전송하여 중앙으로부터 엣지로 트리 구성을 위한 정보 요청 메시지로 인해 발생하는 트래픽을 줄이되, 각 로컬 엣지로부터 받은 장애 상관 관계 트리를 중앙에서 계층적 장애 원인 트리로 구성하여, 기존 장애원인 분석 시스템에 적용하는 방식을 제안한다. 제안된 방식은 오픈스택의 기존 장애 원인 분석 프로젝트인 Vitrage를 기반으로 구현하여, 중앙 클라우드에서 엣지 클라우드로 전달되는 트래픽을 줄이면서도, Root Cause Analysis(RCA)가 효율적으로 수행될 수 있음을 검증하였다. 실험결과 제안한 구조와 단순히 RCA를 적용한 방식 간에 중앙에서 엣지로 전달되는 요청 메시지 트래픽 량과 전체 RCA 트리 구성 시간을 비교 분석하여 제안방식이 효과적임을 확인하였다.

Key Words : Edge Cloud, Root Cause Analysis, Failure analysis, Monitoring, 5G

ABSTRACT

We need a system to efficiently analyze the cause of failure of edge cloud system that provides service at a location close to subscribers. When applying the method applied to the existing single cloud system to the central cloud and applying it to the analysis of the failure cause of the multiple edge cloud systems at the remote site, the traffic for event reporting from each component in the multiple individual edge clouds is distributed to the central cloud which may cause a problem in real time fault reporting. In this paper, the resource status of each edge cloud is collected locally at each edge, collected as information related to each other, and transmitted to the central cloud to reduce the traffic caused by the information request message for tree configuration from the center to the edge. We propose a method to apply the fault correlation tree received from the edge to the existing fault analysis system by constructing a hierarchical fault tree as the center. The

※ 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 정보통신 연구기반구축사업[12221-14-1001, 차세대 네트워크·컴퓨팅 플랫폼연구 기반구축]의 일환으로 수행하였음.
◆ First Author : (ORCID:0000-0002-0127-8739)Soongsil Univ. Dep. of Electronic Engineering, delightwook@dcn.ssu.ac.kr, 학생회원
° Corresponding Author : (ORCID:0000-0002-1066-4818)Soongsil Univ. Dep. of Electronic Engineering, younghak@ssu.ac.kr, 종신회원
논문번호 : 201812-376-D-RN, Received December 4, 2018; Revised January 1, 2019; Accepted January 11, 2019

proposed scheme is implemented based on Vitrage, which is a project to analyze the existing failure cause of openstack, to verify that Root Cause Analysis(RCA) can be efficiently performed while reducing the traffic from the central cloud to the edge cloud. Experimental results show that the proposed scheme is effective by comparing the total amount of RCA tree construction time with the amount of request message traffic delivered from the center to the edge between the proposed structure and the simple RCA applied method.

Ⅰ. 서 론

가입자에 근접한 네트워크 엣지에서 서비스를 제공하는 엣지 클라우드는 5G를 구성하는 주요한 기술로 중요성이 강조 되고 있다.[1] 엣지 클라우드는 지리적으로 가입자와 가까운 위치에서 실시간으로 서비스를 제공하고 요청을 처리하기 때문에 응답 신속성이 높아지며, 불필요한 트래픽이 중앙 클라우드로 전송되는 것을 줄일 수 있어 전체적인 트래픽 감소의 장점을 갖는다. 엣지 클라우드는 차세대 네트워크의 주요한 서비스인 차량, AR/VR, 비디오 서비스 등의 기반 기술로 ETSI(European Telecommunication Standards Institute)를 비롯하여 국내외 많은 표준기관 및 오픈소스 프로젝트 등에서도 연구가 진행 되고 있다.[2-5]

한편 인프라 시스템의 안정성을 확보하기 위해 각 인프라의 장애 원인 분석 체계가 역시 기반환경으로서 같이 필요하게 된다. 관리의 범위가 커지는 엣지 클라우드 환경에서 장애 원인 분석 기법으로서 기존의 모니터링 시스템이 제공하는 장애 관리 기능을 엣지 사이트 까지 확장하는 방법이 일차적으로 가능하다.[6,7]

그러나 기존의 단일 클라우드 시스템에 적용하던 방식을 중앙 클라우드에 적용하고 이를 통해 원격지 다수의 엣지 클라우드 시스템의 장애 원인 분석에 적용할 경우 중앙 클라우드에서 다수의 개별 엣지 클라우드 내의 각 리소스 구성 요소 정보요청을 통해 발생되는 중앙 클라우드로의 트래픽이 과도하게 증가 할 수 있어 문제가 될 수 있다.

본 논문에서는 중앙 클라우드에서 각 엣지로 세부정보 요청 시 각 엣지 클라우드의 리소스 상태는 각 엣지에서 로컬하게 수집하고 이를 상호 연계 된 정보 형태로 취합한 후 중앙 클라우드로 전송하여 발생되는 트래픽을 줄이되 각 로컬 엣지로부터 받은 장애 상관관계 트리를 중앙에서 계층적 장애 원인 트리로 구성하여 기존 장애 원인 분석 시스템에 적용하는 방식을 제안한다. 제안 된 방식은 오픈스택 Root Cause Analysis(RCA) 프로젝트인 Vitrage를 기반으로 엣지 클라우드 환경에 구현하고 중앙 클라우드에서 트래픽을 줄이면서도 효율적인 RCA가 수행 될 수 있음을 검증한다. 서론에 이어 제2장에서는 관련 배경 기술 등을 살펴보고 제 3장에서 제안된 구조의 상세한 설계를, 제4장에서 이를 구현한 결과와 실험 결과를 고찰하고 제5장에서 결론을 맺는다.

Ⅱ. 관련연구

RCA(Root Cause Analysis)라고 불리는 장애에 대한 상관관계를 분석하여, 특정 장애가 발생한 근본적인 원인을 찾는 기술은 원인을 식별하는 방법에 대한 연구로서 여러 가지가 진행되고 있다. CloudRanger는 서비스에 대한 이상 탐지를 감지하고 서비스 간의 상관관계 그래프를 추출 후 이에 대한 비정상적인 패턴을 측정 하는 기능을 제공한다.[8]

그림 1은 CloudRanger의 프레임워크를 나타낸다. CloudRanger는 서비스의 장애 발생을 감지하고, 서비스 리스트 중장애 발생 후보를 검색한다. 서비스의 상관관계를 기반으로 그래프를 추출하게 되며, 서비스 의존도에 따라서 점수를 계산한다. 이러한 점수를 통해 비정상적인 서비스의 순위를 얻으며, 이는 장애 원인이 된다. 하지만 CloudRanger는 문제 검출부터 장애 원인 분석까지 여러 단계 수행에 따라 시간 지연이 발생하기 때문에 이를 엣지 클라우드 환경에서 적용할 경우 실시간 장애 관리가 어렵다.

상관관계 점수를 계산하여 원인을 식별하는 방식대신에 사전에 모니터링 메트릭 간의 관계 정의를 통해 다 계층에서 장애 원인을 분석하는 연구도 진행 되었다.[9]

그림 2는 다 계층에서 관계 정의 기반의 원인 분석

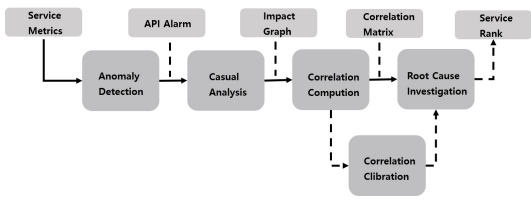

그림 1. CloudRanger 프레임워크
Fig. 1. CloudRanger framework

기능을 제공하는 CEP4Cloud의 구조를 나타낸다. 해당 연구에서는 멀티 레이어 모니터링을 위해 4가지의 모니터링 에이전트를 사용하게 된다. 기본적인 모니터링 시스템과 동일하게 모니터링 에이전트로부터 모니터링 데이터를 수집하고 관리자는 모니터링 데이터 간의 관계를 사전에 정의하여 분석 컴포넌트를 통해 장애 원인을 분석한다. 또한 결과에 따라 액션 매니저는 여러 계층에서 여러 복구 액션을 수행 할 수 있다. CEP4Cloud는 장애 관계를 사전에 정의하여 원인을 분석하기 때문에 빠른 장애 원인 분석 처리가 가능하지만 멀티 레이어 분석을 위한 다수의 모니터링 에이전트 사용으로 부하를 유발할 수 있어 엣지 클라우드 환경에 적용할 경우 전체적인 망에 대한 성능 저하의 원인이 될 수 있다.

오픈소스 프로젝트에서도 다양한 연구가 진행 되고 있다. 인프라 전체에 대한 다양한 기능을 통합하는 ONAP(Open Network Automation Platform) 프로젝트에서는 RCA 기능을 제공하기 위해 Holmes 프로젝트를 세부 프로젝트로 포함하고 있다.[10]

Holmes 프로젝트는 데이터 수집, 이벤트 관리, 분석을 담당하는 DCAE(Data Collection Analysis and Events) 프로젝트와 연계하여 동작하게 되며 NFV 환경에서 여러 구성요소들의 이벤트 상관관계를 분석한다. 이를 위해 관리자는 장애 관계 규칙을 정의하고 전달 된 이벤트를 기반으로 원인을 분석한다. 또한 소스로부터 알람을 수집하여 상관관계 분석 결과를 Policy 컴포넌트 통해 정의한 복구 액션을 수행 할 수 있다. Holmes는 단일 에이전트 사용이 가능하고, 장애 관계를 사전에 정의하여 적은 부하와 빠른 장애 원인 분석 처리가 가능하지만 컨테이너 환경등 여러 가지 확장이 필요한 상황이다.

오픈소스 클라우드 OS인 오픈스택 프로젝트에서도 RCA 프로젝트를 진행하고 있다. Vitrage는 장애 원인 식별을 위하여 관리자는 사전에 특정 템플릿을 작성하게 되고 Vitrage는 해당 템플릿을 기준으로 알람을 비교하여 특정 알람에 대해 연쇄적인 알람을 생성하거나 복구 액션을 취할 수 있다.

그림 4는 오픈스택 Vitrage의 구조를 나타낸다. Vitrage의 데이터 소스 드라이버는 여러 모니터링, 리소스 프로젝트들과 연동 된다. 각각의 드라이버는 상관관계 트리를 구성하기 위해 주기적으로 리소스와 장애 이벤트 정보를 수집하고, 엔티티 그래프 컴포넌트에서 네트워크 X 라이브러리를 기반으로 계층적 장애 원인 분석 트리를 구성한다. 관리자는 구성 된 계층적 트리를 통해 리소스, 가상머신, 이벤트에 대한 정보와 이들의 상관관계를 식별 할 수 있다. 장애 원인 분석은 템플릿을 기반으로 특정 이벤트 발생 여부에 따라 장애 원인을 분석할 수 있으며, 장애를 복구하기 위한 기능을 제공하고 있다. 또한 컨테이너 오케스트레이션 프로젝트인 쿠버네티스와 컨테이너 모니터링 프로젝트인 프로메테우스와도 연동 되어 있어, 컨테이너 환경에서의 RCA가 가능하다.[11][12] 본 논문에서는 이러한 기능의 다양성 및 처리 시간을 고려하여 Vitrage를 기반으로 엣지컴퓨팅 인프라의 장애 원

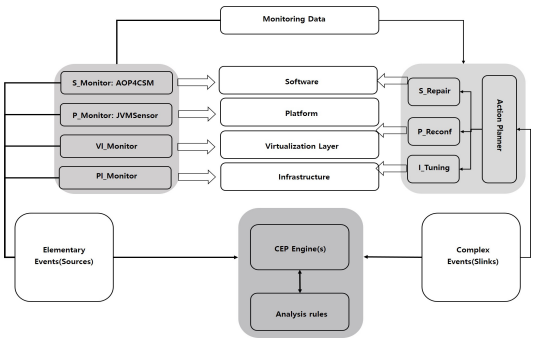

그림 2. CEP4Cloud 구조
Fig. 2. The architecture of the CEP4Cloud

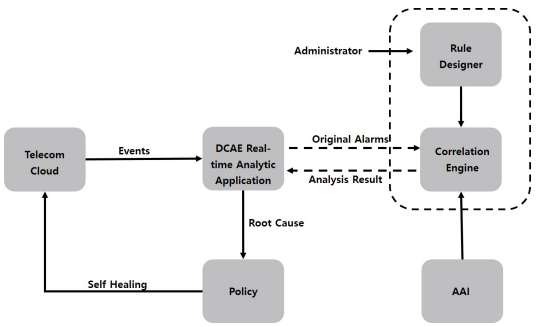

그림 3. Holmes에서의 플로 절차
Fig. 3. The Holmes' flow procedure

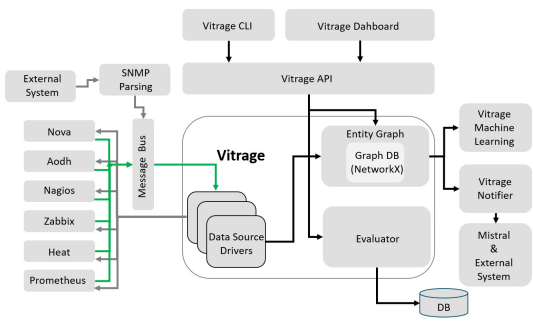

그림 4. Vitrage 구조
Fig. 4. The architecture of Vitrage

인 분석을 위한 시스템을 설계하고 구현을 통해 검증하였다. 본 장에 이어 상세한 설계 내용 및 구현 결과 등을 3, 4장에서 살펴본다.

Ⅲ. 엣지 클라우드 장애 원인 분석 기능 설계

엣지 클라우드의 장애 원인 분석기능의 구현 방법으로서 기존의 RCA기능을 그대로 중앙 클라우드에 적용하고 모든 엣지 클라우드의 상태를 직접 모니터링 및 장애 분석하는 방법이 가능 할 수 있다. 그림 5는 기존의 RCA 방식을 적용한 구조이다. 중앙 클라우드에는 트리를 구성하는 RCA 엔진과 시각화 기능을 담당하는 RCA 시스템으로 구성 된다. RCA 엔진의 데이터 소스 드라이버는 엣지의 각 구성 요소로부터 이벤트를 중앙으로 수집하고 트리를 구성한다.

그러나 이 방식은 중앙 클라우드의 단일 RCA 에이전트를 통한 장애 관계 정의로 빠른 장애 원인 분석이 가능하지만 엣지 클라우드 사이트가 증가할수록 중앙으로 전달할 트래픽 부하 증가로 인해 전체적인 효율이 감소할 수 있게 된다. 본 논문에서는 각 엣지 클라우드에 적용할 수 있는 에이전트를 경량화 하여 개발하고, 이를 기반으로 각 엣지에서 자체적으로 로컬 트리정보를 수집하고 이를 중앙으로 전달하여 최종적으로 중앙에서 계층적 장애 원인 분석 트리를 구성하는 구조를 제안한다. 이를 통하여 중앙에서 엣지로 보내야해야하는 정보요청 메시지를 최소화하면서도 장애 인지 속도는 개선되도록 한다.

현재 RCA는 전체 인프라 및 장애들이 상호 연계되는 상황을 트리 구조 형태로 사전에 정의한 상태에서 장애를 수집하고 임의의 장애 발생 시 정의된 트리를 통해 장애 원인을 찾아내게 된다. 그림 6은 제안된 분산 에이전트 방식의 엣지 RCA 구조를 나타낸다. 각 엣지에 위치한 에이전트 엔진은 기존 RCA 엔진을 경량화한 것으로 각 엣지에 분산 배치된다. 이를 통해 RCA 엔진으로부터 로컬정보를 트리형태로 상호연계성을 정리한 형태로 정보를 수집하여 중앙에서 전체 트리형태를 취합한다. 이러한 구조에서 단계를 나누어 정보를 취합하는 경우 로컬 트리의 복잡성이 증가하여 이를 단계로 나누는 것과 엣지 내부의 자원이 다시 여러 단계로 나누어져 이를 2~3 단계의 트리로 구성하는 경우가 될 수있다. 본 논문에서는 엣지를 하나의 트리로 나누고 중앙에서도 각 엣지를 하나의 가지로 분석하는 경우로 앞서 설명하는 형식의 다단계 구성 시에도 큰 변화없이 적용이 가능하다. 기존 방식의 경우 각 엣지 리소스를 수집하기 위해 중앙에서 엣지로 보내는 정보 요청 메시지 트래픽이 과도하게 발생하여 망 전체에 장애를 유발 할 수 있다. 그러나 제안된 방식의 경우 에이전트 엔진을 통해 리소스 및 이벤트 정보를 로컬하게 수집하여 상호 연계된 정보로 취합 후 중앙으로 전송하기 때문에 중앙에서 각 리소스별로 엣지에 요청하는 메시지를 줄일 수 있게 된다. 그림 6의 구성요소인 각 모듈의 핵심기능과 역할은 아래 Table. 1에 정리하였다. Table. 1에서와 같이 Vitrage에서 RCA API는 중앙으로부터 전달 요청을 수신하고 엣지의 정보를 전달한다. Data Source Drivers는 연계 된 모니터링 툴로부터 장애 알람을 요청하고 이를 수집한다. Graph DB는 일차적으로 기본적인 리소스 트리를 구성하고 전달 받은 장애 알람을

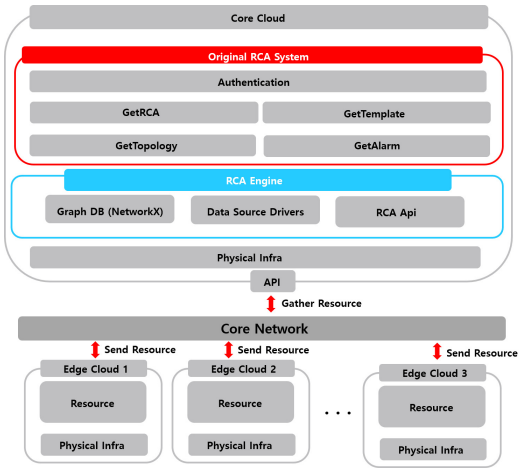

그림 5. 단일 클라우드 시스템 방식의 엣지 RCA 적용 구조
Fig. 5. Single cloud system based edge RCA architecture

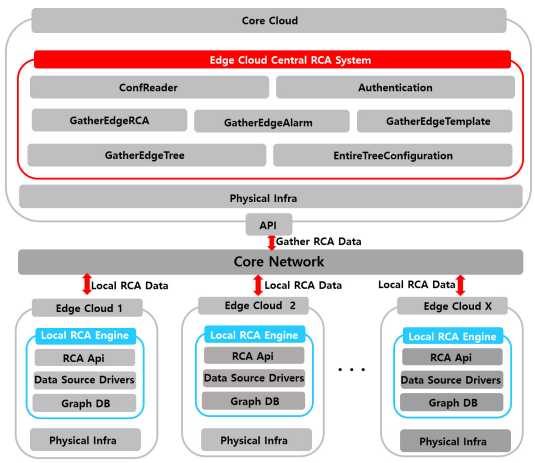

그림 6. 분산 에이전트 방식의 엣지 RCA 구조
Fig. 6. Distributed agent based edge RCA structure

해당하는 리소스와 연결하고 트리 구성 정보를 저장한다. RCA 시스템에서 ConfReader는 관리자가 작성한 각 엣지의 인증 정보를 파일로부터 읽어 들인다. Authentication은 각 엣지에게 트리, 알람과 같은 정보를 요청하기 위해 인증 정보를 기반으로 엣지 클라우드 접근을 위한 클라이언트를 생성한다. GatherEdgeRCA, GatherEdgeAlarm, GahterEdge Template, GatherEdgeTree는 각 엣지에게 RCA, 알람, 템플릿, 로컬 트리 정보를 요청하고 이를 수집한다. EntireTree Configuration은 수집 된 로컬 트리를 인덱스에 맞게 하나의 전체 트리로 재구성하고 이를 시각화하여 관리자가 확인 할 수 있도록 한다.

그림 7은 설계된 시스템의 워크플로우로서 이러한 절차를 세부적으로 나타내고 있다. ConfRead는 엣지를 구성하는 오픈스택에 접근을 위한 인증 파일을 읽어 들여 Authentication 컴포넌트에게 해당 데이터를 보낸다. Authentication 컴포넌트는 키스톤 프로젝트에게 인증 토큰을 요청하고, Vitrage 엔진에게 접근할 수 있는 클라이언트를 생성한다. 인증 된 클라이언트를 통해 알람, RCA, 템플릿, 트리 데이터를 가져오는 GatherEdge 컴포넌트는 해당 데이터를 각 엣지 클라우드에게 요청한다. 엣지 클라우드는 로컬을 통해 리소스 및 이벤트를 수집하고 로컬 트리를 구성하여

표 1. 비트라지 메인 모듈
Table 1. Main Modules in the Vitrae

Module	Description
RCA API	API response / request
Data Source Drivers	Collect fault alarms from monitoring tools
Graph DB	Save and configure tree information
ConfReader	Read the edge cloud authentication information file
Authentication	Create an authenticated client for edge cloud access
GatherEdgeRCA	Collect RCA information from each edge cloud
GatherEdgeAlarm	Collect fault alarm information from each edge cloud
GatherEdgeTemplate	Collect relationship template information from each edge cloud
GatherEdgeTree	Collect local tree from each edge cloud
EntireTreeConfiguration	Reconstruct the collected local tree into an entire tree

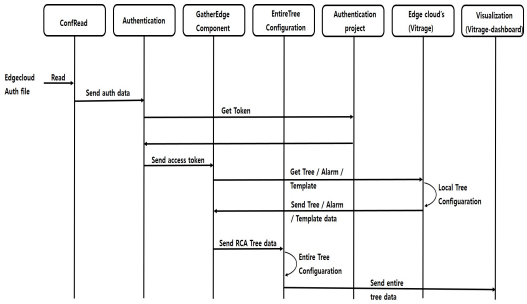

그림 7. 엣지 클라우드 RCA 시스템 워크플로우
Fig. 7. Workflow of Edge cloud RCA System

중앙으로 전달하게 되며, 이는 엣지 클라우드 RCA 시스템의 EntireTreeConfiguration 컴포넌트에 의해 인덱스에 따라 정렬되고 링크 데이터를 기반으로 서로 연결되어 계층적 장애 원인 분석 트리 구성된다.

Ⅳ. 구현 및 실험결과

그림 8에 구현된 엣지 RCA 시스템의 구조를 나타냈다. 각 엣지 클라우드는 컨트롤 노드와 컴퓨트 노드로 구성되어 지며, 인프라 장애 이벤트를 모니터링하기 위하여 오픈소스 모니터링 툴인 자빅스를 사용하였다.[13] 자빅스는 Vitrage 엔진과 연동 되어 있으며, Vitrage 엔진은 자빅스로부터 해당하는 리소스에 대한 이벤트가 있는지를 확인한 뒤 정보를 수집하여 트리를 구성한다. 각 엣지는 구성 요소 리소스를 로컬로 수집하여 취합한 후 중앙으로 전달하게 되는데, 리소스 데이터 및 리소스의 연결을 포함하는 링크 데이터를 사전 형태로 함께 전달한다.

엣지 클라우드 RCA 시스템에서는 각 엣지로부터 트리를 구성하는 정보 데이터와 각 정보를 연결하는

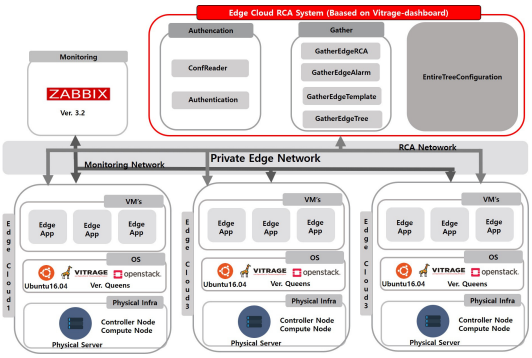

그림 8. 엣지 클라우드 RCA 시스템 구현 구조
Fig. 8. The structure of Edge cloud RCA system implemented

링크 데이터를 수집한 후 인덱스를 기반으로 계층적 장애 원인 분석 트리를 재구성하게 된다. 그림 9는 각 엣지로부터 로컬 트리를 수집하여 중앙에서 구성한 계층적 장애 원인 트리를 나타낸다. 제안한 시스템의 EntireTreeConfiguration과 GatherEdgeTree를 통해 로컬 트리를 각 엣지로부터 수집하고 이들을 중앙 클라우드에서 계층적 장애 원인 트리로 구성한다. 이를 통해 관리자는 전체 엣지 클라우드 인프라의 장애 원인을 파악 할 수 있게 된다.

다양한 형태의 클라우드는 다수의 컴퓨트 노드와 네트워크 노드, 스토리지 노드 그리고 이를 제어하기 위한 컨트롤 노드로, 이러한 노드들의 범위 내에서 구성된다. 기존 RCA 시스템은 네트워크, 컴퓨트, 스토리지 등 다양한 형태의 클라우드에 필수적으로 존재하는 컨트롤 노드를 통해 모니터링 된 결과로부터 정보를 받아 전체 클라우드에 대한 RCA를 진행한다. 그러므로 실제 클라우드의 구성방법이 다양하지만 RCA 분석을 위한 정보수집은 컨트롤 노드와의 통신이 중요한 요소이다. 본 논문에서 제안한 구조 역시 각 단위 엣지 RCA 에이전트와 중앙 RCA 시스템과의 정보 전달량 등에 영향을 받으므로 세부 컴퓨트 노드, 네트워크 노드 등 클라우드 구성에 따른 영향을 덜 받는다. 구현된 시스템의 동작을 확인하고 성능을 비교하기 위해 이를 위해 그림 10과 그림 11과 같은 중앙 클라우드에서 각 엣지 클라우드로 요청한 이벤트 데이터와 링크 데이터를 예로서 사용하였다. 이벤트 데이터는 각 리소스에 대한 인덱스값을 포함하고 있으며 이를 이용하여 중앙에서는 인덱스를 재구성하여 계층적 트리를 완성한다. 즉 전달 받은 이벤트 데이터들은 RCA 트리로 시각화 되는 과정에서 인덱스를 기준으로 매핑 되어 지는데 특정 이벤트 리소스의 경우 인덱스 데이터는 존재하지 않으며, 전달 된 순서에 의존한다. 따라서 인덱스를 재구성하는 경우 전달된 상관관계 트리 수, 이벤트의 수, 엣지 클라우드의 수를 고려하여 인덱스를 재구성한다. 그림 11의 링크 데이터 모델은 소스와 타깃 데이터로서 인덱스 값을 갖고 있고 이를 통해 리소스들의 연결정보를 제공한다.

그림 12는 구현된 시스템에서 완성된 계층적 장애 원인 분석 트리 화면이다. 본 트리로 부터 각 엣지 클라우드의 리소스와 이벤트 발생 현황을 파악할 수 있으며, 물리적인 리소스를 클릭할 경우 포함되어 있는

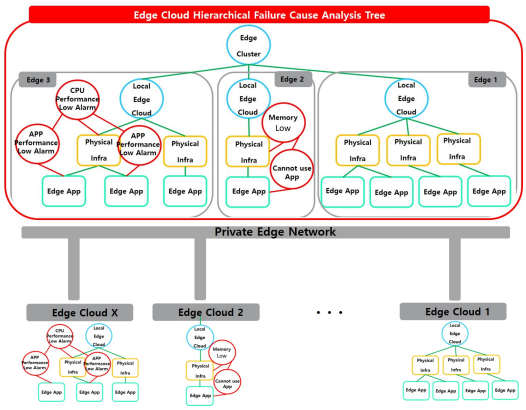

그림 9. 로컬 트리 정보로부터 계층적 트리 구성
Fig. 9. Hierarchical tree construction from local tree information


```
[{u'vitrage_id': u'a4075933-4db9-44f3-8fd9-3e1f6c761adb',
  u'name': 'MECsite1_nova',
  u'update_timestamp': u'2018-11-07 06:51:42.897029+00:00',
  u'vitrage_category': u'RESOURCE',
  u'vitrage_operational_state': u'OK',
  u'state': u'available',
  u'vitrage_type': u'nova.zone',
  u'vitrage_sample_timestamp': u'2018-11-07 06:51:42.897029+00:00',
  u'graph_index': 1,
  u'vitrage_aggregated_state': u'AVAILABLE',
  u'vitrage_is_placeholder': False,
  u'id': 'MECsite1_nova',
  u'is_real_vitrage_id': True,
  u'vitrage_is_deleted': False}]
```


그림 10. 리소스 및 알람 데이터 예
Fig. 10. Example of resource and alarm data


```
[{u'relationship_type': u'contains',
   u'source': 1,
   u'vitrage_is_deleted': False,
   u'key': u'contains',
   u'target': 8},
  {u'relationship_type': u'on',
   u'source': 3,
   u'vitrage_is_deleted': False,
   u'key': u'on',
   u'target': 8}]
```


그림 11. 링크 데이터 예
Fig. 11. Example of link data

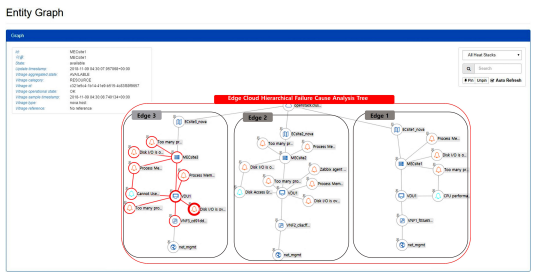

그림 12. 계층적 장애 원인 분석 트리 실제 구현
Fig. 12. Implementation result of the hierachical root cause analysis tree

하위 리소스들이 별도로 도출되어 각 리소스 식별을 용이하게 할 수 있다. 본 트리상에 알람이 표시되고 각 알람 이벤트와 연결 된 원인 이벤트로 부터 해당 장애가 발생하게 된 원인을 파악할 수 있다.

제안한 구조와 기존의 중앙 집중 구조의 비교분석을 위해 먼저 각 각의 구조에서 엣지로의 정보 요청 트래픽 량을 분석하였다. 먼저 기존 RCA 시스템을 수정 없이 엣지 클라우드의 정보 수집에 그대로 적용할 경우 중앙 클라우드에서 엣지로 전달되는 정보 요청 메시지로 부터의 총 트래픽 량은

$$S_{\mathrm{or}iginal} * (R * N_e) \quad (1)$$

이 된다. 여기서 $S_{\mathrm{or}iginal}$ 은 중앙에서 각 엣지의 리소스의 상태를 점검하기 위해 주기적으로 보내는 메시지로서 최소 512 Bytes 부터 리소스의 종류에 그 이상이 된다. R은 엣지 클라우드의 개수를 나타내며 N_e는 각 엣지내의 리소스의 개수가 된다. 그러므로 모든 리소스별 요청 메시지의 크기를 최소 크기로 동일하다고 가정하면 (1)과 같은 총 트래픽 량이 도출된다. 한편, 제안한 구조의 경우 중앙에서 각 엣지의 로컬 RCA 엔진에게만 한 번에 트리 정보 요청 메시지를 송신하게 되어 이때 요청한 메시지의 크기를 $S_{suggestion}$라 하면 모든 엣지에 요청한 메시지로 부터의 총 트래픽 량은 다음과 같게 된다. 다만 $S_{suggestion}$는 기존 메시지 크기보다는 요청정보구성이 증대되어 본 구현에서는 789 Bytes가 되었다.

$$S_{suggestion} * N_e \quad (2)$$

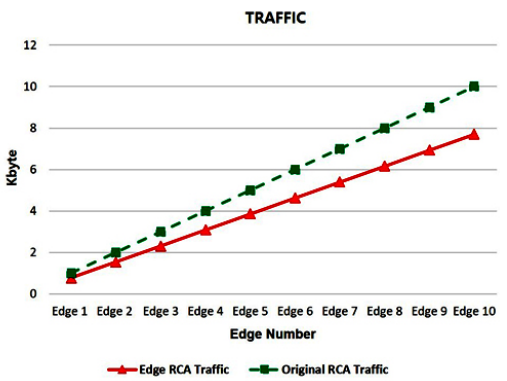

그림 13. 중앙 클라우드에서 엣지로의 요청 메시지로부터 발생된 총 트래픽
Fig. 13. Total traffic of the information request messages from the center cloud to the edge

이로부터 엣지 클라우드의 개수를 증가 시키면서 총 트래픽 량의 변화를 도시하면 그림 13과 같은 결과를 얻을 수 있다. 그림에서 리소스의 개수는 엣지 클라우드가 기본적으로 한 개의 compute node와 내부 network로 구성된 것으로 가정하여 2로 설정하였다. 그림에서 제안 된 구조는 개별 엣지 클라우드내의 리소스의 수가 증가하여도 영향을 받지 않아 상대적으로 기존 구조보다 엣지 개수를 늘려도 트래픽 증가가 적음을 알 수 있다.

이러한 결과에 따라 기존 RCA 시스템은 리소스 정보 요청을 위해 중앙 노드에서 정보 수집을 위한 요청 메시지를 전송하게 된다. 요청 메시지는 리소스의 종류에 따라 상이하지만 최소 512 Byte의 크기를 가지게 된다. 반면 제안한 RCA 시스템은 RCA 정보 요청을 위해 중앙 RCA 시스템에서 각 엣지 클라우드에게 RCA 정보 수집을 위한 요청 메시지를 전송하게 되며 별도의 메시지 프로토콜을 사용하기 때문에 789 Byte의 일정한 크기를 가진다. 따라서 제안한 RCA 시스템은 클라우드 수가 증가하여도 각각의 엣지클라우드로 일정한 크기를 가지는 요청 메시지를 각각 보내기 때문에 엣지 클라우드의 수가 증가하여도 비슷한 형태의 결과를 나타낼 것으로 예상된다.

또한 기존 RCA 시스템에서 RCA를 수행하기 위해 가장 기본이 되는 모니터링 시스템은 모니터링 대상에 설치 되는 특정 에이전트를 통해 정수 혹은 실수 형태의 데이터를 수집하여 모니터링 서버로 전달하게 된다. 수집 된 데이터는 관리자가 설정한 조건과 비교하여 데이터의 값이 같거나, 크거나 작을 경우 장애 알람을 발생시키게 된다. 기존 RCA 시스템은 주기적으로 모니터링 시스템으로부터 장애의 종류에 상관없이 발생하는 장애 알람을 일괄적으로 가져와 상관 관계 트리를 구성하게 된다. 따라서 제안한 RCA 시스템 또한 장애의 종류와 관계없이 단순히 모니터링 시스템으로부터 발생한 장애 알람을 일괄적으로 가져오기 때문에 처리시간은 장애 종류보다 클라우드 수에 지장을 받게 된다.

이어서 제안한 구조와 기존 구조 각각에 대해서 중앙 클라우드에 계층적 장애 원인 분석 트리가 완성될 때까지의 시간을 측정을 통하여 비교 하였다. 기존 방식은 각 엣지의 리소스 정보를 요청하고 중앙에서 취합하지만, 제안 방식은 각 엣지가 구성한 트리 정보를 중앙에서 취합한다. 그에 따른 최종 RCA 구성까지 걸린 시간을 비교하기 위해 리소스 수를 변화시켜가면서 트리 구성 완료까지의 시간을 측정하였다. 그림 14에서 제안구조가 리소스를 증가에 관계없이 전체적

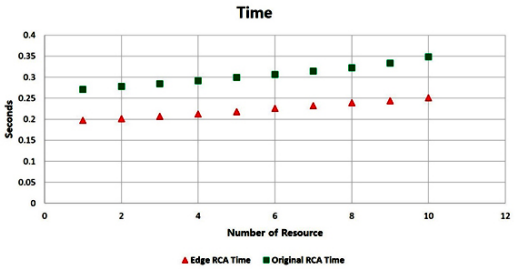

그림 14. 리소스 타입 증가에 따른 RCA 구성 시간
Fig. 14. RCA configuration time vs number of resources types

으로 기존 구조 보다 최종 RCA 트리 구성에 걸린 시간이 적음을 확인 할 수 있다. 이는 제안구조에서는 각 엣지 클라우드가 사전에 로컬 리소스를 수집하고 중앙으로 전달하고 이를 이용하여 중앙에서 최종 RCA 트리를 구성하므로 매 리소스별로 정보를 요청하여 중앙에서 트리를 구성하는 기존 방식보다 효과적임을 보이고 있다.

V. 결 론

본 논문에서 효율적인 엣지 클라우드의 장애 원인 분석 시스템을 제안하고 구현을 통하여 동작을 확인하였다. 제안된 시스템에서 각 엣지 클라우드의 리소스 상태는 각 엣지에서 로컬하게 수집하고 상호 연계된 정보로 취합한 후 중앙 클라우드로 전송하도록 하여 중앙으로 보내지는 트래픽을 줄이면서도 최종적인 계층적 장애원인 트리의 구성 시간은 줄일 수 있도록 하였다. 구현된 시스템은 기존의 RCA를 단순히 적용하여 모든 엣지의 리소스 상태를 직접 취합하여 분석하는 시스템과 엣지로의 총 트래픽 양과 최종 RCA 트리를 구성할 때까지의 시간을 비교하여 성능이 개선됨을 확인 하였다. 두 구조 간의 중앙 클라우드로부터의 트래픽 량과 전체 RCA 트리 구성 시간을 비교 분석하여 제안 방식이 효과적임을 확인 하였다. 제안된 시스템에 의해 수집된 정보를 바탕으로 인공지능 기반 분석시스템 등과 같은 별도의 상세 분석 시스템을 통합할 경우 궁극적인 자동화된 엣지 클라우드 장애분석 시스템이 완성될 수 있을 것이다.

References

[1] T. Taleb, et al., “On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration,” *IEEE Commun. Surv. & Tuts.*, vol. 19, no. 3, pp. 1657-1681, Thirdquarter, 2017.

[2] ETSI, *ETSI GS MEC-IEG 004* v1.1.1(2015), Retrieved Oct. 3, 2018, from https://www.etsi.org.

[3] M. Patel and A. Pandya, “Edge computing: Design a framework for monitoring performance between datacenters and devices of edge network,” *IJCMS*, vol. 6, no. 6, pp. 74-77, Jun. 2017.

[4] B. Varghese and R. Buyya, “Next generation cloud computing: New trends and research directions,” *Future Generation Comput. Syst.*, vol. 79, no. 3, pp. 849-861, Feb. 2018.

[5] M.-S. Son, S.-H. Chung, and W.-S. Kim, “Fog-server placement technique based on network edge area traffic for a fog-computing environment,” *J. KIISE*, vol. 45, no. 6, pp. 598-610, Jun. 2018.

[6] J. Lin, et al., “Automated anomaly detection and root cause analysis in virtualized cloud infrastructures,” *2016 IEEE/IFIP Network Operations and Management Symp.(NOMS)*, pp. 550-556, Istanbul, Turkey, Apr. 2016.

[7] J. Weng, et al., “Root cause analysis of anomalies of multitier services in public clouds,” *IEEE/ACM Trans. Networking*, vol. 26, no. 4, pp. 1646-1659, Jun. 2018.

[8] P. Wang, et al., “CloudRanger: Root cause identification for cloud native systems,” *2018 18th IEEE/ACM CCGRID*, pp. 492-502, Washington DC, USA, May 2018.

[9] A. Mdhaffar, et al., “Reactive performance monitoring of Cloud computing environments,” Cluster Computing, vol. 20, no. 3, pp. 2465-2477, Sep. 2017.

[10] ONAP, *Holmes Project Documentation*(2017), Retrieved Sep. 10, 2018, from https://wiki.onap.org.

[11] Vitrage, *Kubernetes Datasource*(2018), Retrieved Sep. 10, 2018, from https://docs.openstack.org/vitrage

[12] Prometheus, *Prometheus Documentation*(2018),

Retrieved Sep. 10, 2018, from https://prometheus.io

[13] Zabbix, *Zabbix Manual*(2018), Retrieved Sep. 11, 2018, from https://www.zabbix.com/documentation/3.2.

김 민 욱 (Min-wook Kim)

2017년 2월 : 안양대학교 정보통신공학과 졸업

2019년 2월 : 숭실대학교 정보통신소재융합 석사

<관심분야> NFV, Cloud, Edge Computing, 모바일 네트워크

김 영 한 (Young-han Kim)

1986년 2월 : 한국과학기술원 전기 및 전자 공학과 석사

1990년 2월 : 한국과학기술원 전기 및 전자 공학과 박사

1994년~현재 : 숭실대학교 정보통신전자공학부 교수

<관심분야> 모바일 네트워크, 이동성 관리 기술, SDN/NFV, 센서 네트워크