(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0070445
(43) 공개일자 2024년05월21일

(51) 국제특허분류(Int. Cl.)
H04L 67/5682 (2022.01) *H04L 12/66* (2006.01)
H04L 67/563 (2022.01)
(52) CPC특허분류
H04L 67/5682 (2022.05)
H04L 12/66 (2013.01)
(21) 출원번호 10-2023-0157138
(22) 출원일자 2023년11월14일
심사청구일자 2023년11월14일
(30) 우선권주장
1020220151602 2022년11월14일 대한민국(KR)

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
송충건
서울특별시 마포구 월드컵북로47길 10, 105동 1404호
유헌창
서울특별시 노원구 노원로 214, 6동 202호
(74) 대리인
김등용

전체 청구항 수 : 총 5 항

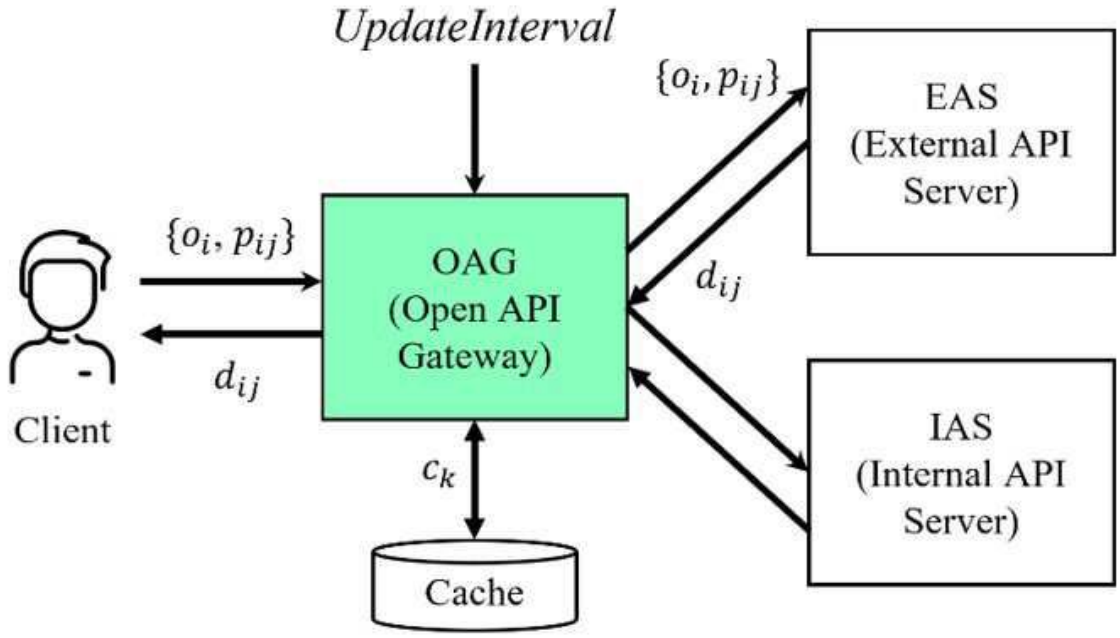
(54) 발명의 명칭 **오픈 API 기반 어플리케이션의 서비스 품질 보장을 위한 API 게이트웨이 및 이의 동작 방법**

(57) 요 약

오픈 API(Application Programming Interface) 기반 어플리케이션의 서비스 품질 보장을 위한 API 게이트웨이(API Gateway) 및 이의 동작 방법이 개시된다. 상기 API 게이트웨이의 동작 방법은 클라이언로부서 오픈 데이터 요청을 수신하여 상기 오픈 데이터 요청에 대응하는 오픈 데이터를 상기 클라이언트로 전송하는 API 게이트웨이

(뒷면에 계속)

대 표 도 - 도2



의 동작 방법으로, 상기 클라이언트로부터 상기 오픈 데이터 요청을 수신하는 단계, 상기 오픈 데이터 요청에 대응하는 오픈 데이터가 상기 API 게이트웨이에 캐싱되어 있는지 여부를 판단하는 단계, 상기 오픈 데이터가 상기 API 게이트웨이에 캐싱되어 있는 경우, 상기 오픈 데이터를 상기 클라이언트로 전송하는 단계, 상기 오픈 데이터가 상기 API 게이트웨이에 캐싱되어 있지 않은 경우, 상기 오픈 데이터 요청을 API 서버로 전송하고 상기 API 서버로부터 상기 오픈 데이터 요청에 대응하는 오픈 데이터를 수신하는 단계, 및 상기 API 서버로부터 수신된 상기 오픈 데이터를 상기 클리이언트로 전송하는 단계를 포함한다.

(52) CPC특허분류
H04L 67/563 (2022.05)

이 발명을 지원한 국가연구개발사업

과제고유번호	1405005547
과제번호	2022427C10-2224-0801
부처명	산림청
과제관리(전문)기관명	한국임업진흥원
연구사업명	산림기반사회문제해결실증기술개발
연구과제명	산림 정보 융합 서비스의 실시간 처리를 지원하는 지능형 엣지 클라우드 시스템 개발
기 여 율	1/1
과제수행기관명	고려대학교 산학협력단
연구기간	2022.04.01 ~ 2022.12.31

명 세 서

청구범위

청구항 1

클라이언로부서 오픈 데이터 요청을 수신하여 상기 오픈 데이터 요청에 대응하는 오픈 데이터를 상기 클라이언트로 전송하는 API 게이트웨이(Application Programming Interface Gateway)의 동작 방법에 있어서,

상기 클라이언트로부터 상기 오픈 데이터 요청을 수신하는 단계;

상기 오픈 데이터 요청에 대응하는 오픈 데이터가 상기 API 게이트웨이에 캐싱되어 있는지 여부를 판단하는 단계;

상기 오픈 데이터가 상기 API 게이트웨이에 캐싱되어 있는 경우, 상기 오픈 데이터를 상기 클라이언트로 전송하는 단계;

상기 오픈 데이터가 상기 API 게이트웨이에 캐싱되어 있지 않은 경우, 상기 오픈 데이터 요청을 API 서버로 전송하고 상기 API 서버로부터 상기 오픈 데이터 요청에 대응하는 오픈 데이터를 수신하는 단계; 및

상기 API 서버로부터 수신된 상기 오픈 데이터를 상기 클리이언트로 전송하는 단계를 포함하는 API 게이트웨이의 동작 방법.

청구항 2

제1항에 있어서,

상기 API 서버로부터 상기 오픈 데이터 요청에 대응하는 오픈 데이터를 수신하는 단계는,

상기 오픈 데이터에 대한 요청의 횟수에 대한 카운트값이 미리 정의된 임계치를 초과하는 경우, 상기 API 서버로부터 수신한 상기 오픈 데이터를 상기 API 게이트웨이에 캐싱하는 단계를 더 포함하는,

API 게이트웨이의 동작 방법.

청구항 3

제2항에 있어서,

상기 API 게이트웨이는,

미리 정해진 시간 주기에 따라 상기 API 게이트웨이에 캐싱되어 있는 오픈 데이터들 각각에 대한 요청의 횟수에 대한 카운트값을 감소시키는,

API 게이트웨이의 동작 방법.

청구항 4

제3항에 있어서,

상기 API 게이트웨이는,

상기 API 게이트웨이에 캐싱되어 있는 오픈 데이터들 중 상기 미리 정의된 임계치 보다 작은 요청의 횟수에 대한 카운트값을 갖는 오픈 데이터를 캐시에서 제거하는,

API 게이트웨이의 동작 방법.

청구항 5

제4항에 있어서,

상기 API 게이트웨이는, 내부 API 서버(Internal API Server, IAS)을 대상으로 하는 오픈 데이터 요청을 수신하는 경우, 수신된 오픈 데이터 요청을 상기 내부 API 서버로 전송하고, 상기 내부 API 서버로부터 수신되는 오픈

데이터를 상기 클라이언트로 전송하는,

API 게이트웨이의 동작 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 오픈 API(Application Programming Interface) 기반 어플리케이션의 서비스 품질을 보장하기 위해, API 게이트웨이에서 오픈 API를 통해 수집하는 데이터에 대하여 캐싱을 수행하는 방법에 관한 것이다.

배 경 기 술

[0002] API는 SW 모듈(software modules) 사이에서 기능을 호출하거나 필요한 데이터를 요청하기 위해 활용된다. 모듈화된 여러 백엔드 서비스에서 제공하는 API를 통합 관리하고 API를 사용하는 클라이언트에 대한 접근 제어를 효과적으로 수행하기 위해 API 게이트웨이를 활용한다. 이러한 API 게이트웨이는 사용자의 서비스 요청에 대한 시작 지점이라는 구조적 특징으로 인하여 기본적인 기능뿐만 아니라 리소스 보호를 위한 트래픽 제한 기능과 같은 특수한 목적의 기능이 구현된다.

[0003] MSA(Microservice Architecture) 형태의 SW 개발 방법론에서 사용자는 API는 게이트웨이의 DevOps에 대한 편의를 위해 클라우드 환경에서 SaaS 형태의 서비스를 이용하며, 이러한 방법론이 보편화됨에 따라 API 게이트웨이의 역할의 중요성이 높아지고 있다. 이에 따라 API 게이트웨이에 특수한 기능을 구현한 다양한 연구들이 등장하고 있다.

[0004] 오픈 API는 누구나 활용 가능한 오픈 데이터 수집을 위해 API 서버를 통해 제공되는 API이다. 이러한 오픈 데이터 제공처 또는 통합 플랫폼에서 제공하는 API를 이용할 시 사용량 제한과 불안정한 QoS(Quality of Service)가 발생할 수 있다. 사용자 요청을 처리하기 위해 외부 기관의 API의 응답 시간이 QoS에 영향을 미친다. 그러나 기존 연구들은 통제 가능한 백엔드 서버들을 가정하고 대상으로 설계되어 외부 API 서버를 활용할 경우 응답 시간을 보장할 수 없다. 이에 따라 불확실한 성능을 보이는 외부 API 서버를 대상으로 안정적인 성능을 보장하는 API 게이트웨이가 요구된다.

[0005] 본 발명에서는 오픈 API를 활용하는 애플리키이션의 QoS 보장을 위하여, OAG(Open API Gateway, 오픈 API 게이트웨이)로 불리는 API 게이트웨이를 제안한다. 제안하는 API 게이트웨이는 OARC(Open API Response Using Caching) 알고리즘 및/또는 COAC(Cached Open API Cleaning) 알고리즘을 포함할 수 있다. 본 발명의 기여점은 아래와 같다.

[0006] 외부 기관에서 제공하는 오픈 API를 활용하는 애플리케이션에 대한 QoS를 보장하는 API 게이트웨이인, OAG를 설계하였다.

[0007] 오픈 API를 활용하는 애플리케이션의 API 요청에 대한 QoS를 보장하는 OARC 알고리즘과 유동적인 오픈 API 이용 트래픽에 적응하기 위한 COAC 알고리즘을 제안한다.

[0008] 산림청 산불위험예보정보 오픈 API에 대한 성능 실험을 통해 OAG가 안정적으로 QoS를 보이는 사실을 확인하였다.

선행기술문헌

비특허문헌

[0009] (비특허문헌 0001) Zuo, Xianyu, et al. "An API gateway design strategy optimized for persistence and coupling." Advances in Engineering Software 148 (2020): 102878.

(비특허문헌 0002) Xu, Rongxu, Wenquan Jin, and Dohyeun Kim. "Microservice security agent based on API gateway in edge computing." Sensors 19.22 (2019): 4905.

발명의 내용

해결하려는 과제

[0010] 본 발명이 이루고자 하는 기술적인 과제는 오픈 API 기반 어플리케이션의 서비스 품질 보장을 위한 API 게이트웨이와 이의 동작 방법을 제공하는 것이다.

과제의 해결 수단

[0011] 오픈 API(Application Programming Interface) 기반 어플리케이션의 서비스 품질 보장을 위한 API 게이트웨이(API Gateway) 및 이의 동작 방법이 제공된다.

[0012] 본 발명의 일 실시예에 따른 API 게이트웨이의 동작 방법은, 클라이언로부서 오픈 데이터 요청을 수신하여 상기 오픈 데이터 요청에 대응하는 오픈 데이터를 상기 클라이언트로 전송하는 API 게이트웨이의 동작 방법으로서, 상기 클라이언트로부터 상기 오픈 데이터 요청을 수신하는 단계, 상기 오픈 데이터 요청에 대응하는 오픈 데이터가 상기 API 게이트웨이에 캐싱되어 있는지 여부를 판단하는 단계, 상기 오픈 데이터가 상기 API 게이트웨이에 캐싱되어 있는 경우, 상기 오픈 데이터를 상기 클라이언트로 전송하는 단계, 상기 오픈 데이터가 상기 API 게이트웨이에 캐싱되어 있지 않은 경우, 상기 오픈 데이터 요청을 API 서버로 전송하고 상기 API 서버로부터 상기 오픈 데이터 요청에 대응하는 오픈 데이터를 수신하는 단계, 및 상기 API 서버로부터 수신된 상기 오픈 데이터를 상기 클리이언트로 전송하는 단계를 포함한다.

발명의 효과

[0013] 본 발명의 실시예에 따른, 오픈 API 기반 어플리케이션의 서비스 품질 보장을 위한 API 게이트웨이와 이의 동작 방법에 의할 경우, 응답 시간이나 가용성에 대한 안정적인 QoS를 보장할 수 있다.

도면의 간단한 설명

[0014] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 상세한 설명이 제공된다.

도 1은 실시간 오픈 API의 응답 시간 측정 결과를 도시한다.

도 2는 OAG와 다른 구성요소 사이의 상호 동작을 설명하기 위한 개념도이다.

도 3은 캐싱 기법의 동작 시나리오를 설명하기 위한 흐름도이다.

도 4는 API 응답시간 실험의 결과를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0015] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태들로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

[0016] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.

[0017] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.

[0018] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0019] 본 명세서에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0020] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0021] 이하, 본 명세서에 첨부된 도면들을 참조하여 본 발명의 실시예들을 상세히 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0022] **API 게이트웨이(API Gateway)**

[0023] 마이크로서비스 형태의 SW 개발 방법론이 보편화되고 API Gateway의 역할의 중요성이 알려지면서 다양한 연구가 등장하였다. Zuo et al.(Zuo, Xianyu, et al. "An API gateway design strategy optimized for persistence and coupling." Advances in Engineering Software 148 (2020): 102878.)는 비용(cost), 성능(performance), 그리고 유지보수도(maintainability) 사이의 균형을 위한 모델과 코어 서비스와 확장 기능을 분리할 수 있는 모듈화를 수행하였다. Xu et al.(Xu, Rongxu, Wenquan Jin, and Dohyeun Kim. "Microservice security agent based on API gateway in edge computing." Sensors 19.22 (2019): 4905.)는 보안 강화를 위해 엣지 시스템에서 API 게이트웨이 역할을 수행하는 마이크로서비스 에이전트를 제안하였다.

[0024] 하지만, 이러한 연구들은 API가 통제 가능한 온프레미스 환경을 가정한다. 외부 API 서버는 서비스 수요가 많은 시간에 응답시간이 일관적이지 않고, 성능 예측을 위한 메트릭을 직접 수집할 수 없다. 이에 따라 외부 API 서버와 상호작용 이후에 실행되는 애플리케이션의 작업은 API 서버의 성능에 종속된다. 그러므로, 본 발명에서는, 외부 API 서버의 종속성을 없애기 위한 방향의 연구를 진행하였으며, OAG의 동작에 반영하였다.

[0025] 오픈 API 기반의 애플리케이션의 QoS 보장을 위한 연구의 필요성을 확인하기 위한 동기 실험을 진행하였다. 실험은 특정 Open API 요청에 대하여 실시간 오픈 데이터를 제공하는 API 서버의 QoS가 안정적인지 확인하기 위한 목적으로 진행하였다. 실험에서 한국의 산림청에서 제공하는 산불위험예보정보 Open API를 활용했으며, 공공데이터 포털 API 서버를 대상으로 오픈 데이터 요청에 대한 응답시간을 측정하는 실험을 진행하였다(산림청 산불위험예보정보, https://www.data.go.kr/data/15084817/openapi.do). 성능 분석을 위한 도구는 본 발명의 발명자의 선행 연구 결과물인 OTT(Open data Testing Tool)를 활용하였다(오픈 데이터 기반 애플리케이션의 QoS 보장을 위한 자동화된 API 분석 도구, 컴퓨터교육학회, 2022.). OTT는 몇 가지 설정 정보 입력을 통해 성능 분석에 대한 요구사항을 명세하고, 이에 따라 자동화된 분석을 수행한다.

[0026] 외부 기관에서 제공하는 오픈 API는 시스템 가용성 유지와 무의미한 시스템 자원 낭비를 줄이고자 1일 사용량 제한을 가지고 있다. 이 점을 고려하여 API 요청 간격은 수학식 (1)과 같은 규칙을 정하고 쿼리 간격 queryInterval을 (runtime/limit)을 기반으로 6으로 결정하였다. 도 1에서는 동기 실험의 결과를 보여준다.

[0027] [수학식 1]

[0028] $$queryInterval = \begin{cases} \frac{runtime}{limit}, & if \;\; limit < runtime \\ 1, & if \;\; limit \geq runtime \end{cases}$$

[0029] 실험 결과 1:00 부터 12:00까지 매우 불안정한 응답 시간을 확인할 수 있다. 구체적인 성능 메트릭으로 가용성은 74%이며, 표준 편차는 4.11이다. 이에 따라 산림청 산불위험예보정보 오픈 API를 저녁에 활용할 시 QoS에 직접적인 영향을 미치는 사실을 확인할 수 있다. 제안하는 기법에서는 QoS 보장을 위해 데이터 캐싱 기법을 적용한 OAG를 제안한다.

[0030] 이하에서는, 오픈데이터 캐싱 기법을 통해 애플리케이션의 QoS를 보장하는 API 게이트웨이(OAG라 명명될 수 있음)를 설명한다. OAG는 적어도 프로세서(Processor)나 컨트롤러(Controller)와 같은 연산 장치나 처리 장치 및/

또는 캐시나 메모리와 같은 저장 장치를 포함하는 컴퓨팅 장치로 구현될 수 있다. 컴퓨팅 장치는 서버(Server), PC(Personal Computer) 등을 포함할 수 있다. OAG는 클라이언트(또는 클라이언트 단말), 적어도 하나의 제1 API 서버, 및 적어도 하나의 제2 API 서버 중 적어도 하나의 유무선 통신망을 통하여 데이터, 요청, 메시지, 또는 패킷 등을 주고받을 수 있다. 구체적으로, OAG는 클라이언트의 요청에 응답하여 수신된 요청을 제1 API 서버로 전송하고 요청에 대응하는 데이터(오픈 데이터를 의미할 수 있음)를 제1 API 서버로부터 수신한 후 이를 클라이언트에게 전송할 수 있다. 또한, OAG는 클라이언트의 요청에 응답하여 응답에 대응하는 데이터(OAG에 저장되어 있는 오픈 데이터를 의미할 수 있음)를 클라이언트로 전송하거나, 응답에 대응하는 데이터를 제2 API 서버로부터 수신하여 클라이언트로 전송할 수도 있다. 여기서, 제1 API 서버는 내부 API 서버(Internal API Server, IAS)를, 제2 API 서버는 외부 API 서버(External API Server, EAS)를 의미할 수 있다.

[0031] 본 명세서에서 인용하는 기호에 대한 설명은 표 1에서 나타내고 있다.

[0032] [표 1]

[0033]

Symbol	Description
O	$\{o_i \mid o_i \text{ is } i\text{th open API}, 1 \le i \le M\}$
P_i	$\{p_{ij} \mid p_{ij} \text{ is } j\text{th parameter set of } o_i, 1 \le j \le N\}$
d_{ij}	Result value mapped to p_{ij} of o_i
$count[p_{ij}]$	Number of request for p_{ij}
C	$\{c_k \mid c_k \text{ is } k\text{th data stored in cache}, 1 \le k \le L\}$
λ	Threshold value of $count[p_{ij}]$ for classification of data

[0034] 캐싱 기법에서 저장되는 값은 c_i이며, o_i의 파라미터 특정 조합인 p_{ij}와 대응되는 오픈데이터 d_{ij}로 구성된다. 즉, $c_i = \{p_{ij}, d_{ij}\}$. p_{ij}는 C에 저장된 캐싱 데이터들 중 원하는 데이터를 조회하기 위한 키로 활용될 수 있다.

[0035] OAG는 사전 연구를 통해 개발된 OTT가 도출한 QoS 메트릭들을 활용할 수 있다. OAG의 동작은 오픈 API에 대한 사전 분석 단계와 분석 결과를 런타임에 활용하여 QoS를 보장하는 실행 단계로 진행된다.

[0036] 사전 분석 단계에서는 가용성(*Availability*)과 편차(*Deviation*)를 도출하고 서비스의 요구사항에 맞지 않는 오픈 API를 선별하여 오픈 API 집합 O를 구성한다. O는 캐싱을 수행하는 후보군으로 활용되며, 캐시에서 관리되는 오픈 API는 *UpdateInterval* 간격으로 외부 서버(외부 API 서버를 의미할 수 있음)에 쿼리를 보내 업데이트를 수행할 수 있다. 다음으로 OAG와 다른 구성요소 사이의 상호작용하는 구조를 설명하고, 오픈 데이터 요청 빈도에 따라 캐싱 데이터를 결정하는 OARC 알고리즘과 주기적으로 캐싱 데이터를 업데이트하는 COAC 알고리즘을 설명한다.

[0037] 예시적으로, 사전 분석 단계에서, OAG는 접속되어 있는 EAS들 중에서 응답 시간, 응답 지연, 및 응답 시간이나 응답 지연의 편차 중 적어도 하나가 미리 정해진 값을 초과하는 API 서버(외부 API 서버)를 캐싱을 수행할 후보 API 서버로 결정할 수 있다. 이를 위해, OAG는 접속된 API 서버들, 특히 외부 API 서버들을 모니터링할 수 있다.

[0038] **아키텍처(Architecture)**

[0039] 이하에서는, 제안하는 API 게이트웨이가 동작하는 구조를 설명한다. OAG는 클라이언트(클라이언트의 단말을 의미할 수 있음)로부터 받은 오픈 데이터 요청을 외부 API 서버(EAS) 또는 내부 API 서버(IAS)에 전달하여 결과를 반환할 수 있다. 그리고 OAG는 EAS에 대한 부분적 오픈 데이터를 캐싱할 수 있다. 도 2는 OAG와 다른 구성요소 사이의 상호 동작을 설명하기 위한 개념도이다.

[0040] IAS는 사설 시스템에서 통제 가능한 자원이기 때문에 예상 가능한 QoS를 가진다. 이에 따라 캐싱 기법을 적용하지 않고 API에 전달을 라우팅하고 수집한 결과를 반환하는 동작을 수행한다. 그리고 EAS가 제공하는 오픈 데이터 요청이 발생할 경우 각 오픈 API 별 요청 빈도 저장한다. 캐싱이 수행되고 사용자에게 전달되는 시나리오는 도 3에서 설명하고 있다.

[0041] 시나리오 1(Scenario 1)은 요청한 오픈 API에 대한 응답을 수행하고 캐시에 데이터를 추가하는 순서를 나타내고 있다. 클라이언트는 API 게이트웨이에 API를 요청하고 API 게이트웨이는 클라이언트를 대신하여 EAP에 데이터를 요청 및 수집 후 전달한다. API는 사용 빈도가 높은 API의 파라미터 조합을 대상으로 클라이언트에 오픈 데이터

전달 전(전달 후나 전달과 동시에 캐싱하는 것고 가능함) 캐싱을 수행할 수 있다. 여기서 빈도가 높다는 것은, 해당 오픈 데이터에 대한 요청이 (미리 정해진 시간 구간 동안에) 미리 정해진 임계치를 초과하여 발생(또는 요청된) 경우를 의미할 수 있다.

[0042] 시나리오 2(Scenario 2)는 캐시에 저장된 데이터를 바로 응답하는 시나리오를 설명한다. API 게이트웨이는 사용자 요청에 대하여 EAP에 직접 요청하지 않고 API 게이트웨이에 캐싱된 캐싱 데이터를 집적 서비스(전달)할 수 있다. 즉, 이 경우에는, 클라이언트가 요청한 오픈 데이터가 OAG에 캐싱되어 있는(저장되어 있는) 경우이다.

[0043] **OARC(Open API Response Using Caching) 알고리즘**

[0044] OARC 알고리즘은 OAG가 오픈 데이터 요청 횟수를 측정하고 캐싱 대상을 설정하는 알고리즘이다. 이는 오픈 데이터 o_i의 p_{ij}에 요청이 발생한 시점에 실행된다. 1줄은 요청한 오픈데이터의 파라미터에 대응되는 오픈 데이터가 캐시에 존재하는지 확인하는 과정이다. 만약 캐시에서 오픈 데이터를 찾으면 EAS에 요청하지 않고 바로 반환할 수 있다. 4줄에서 8줄은 캐시에 오픈 데이터가 없을 시 p_{ij}.count에 1을 더하여 업데이트하고 5줄과 같이 임계치 이상의 요청이 발생할 경우 파라미터 조합과 오픈 데이터 조합을 캐시에 추가하는 작업을 진행한다.

[0045] [알고리즘 1]

Algorithm 1 OARC

Input: p_{ij}, λ
Output: d_{ij}
01: **if** $p_{ij} \in C$
02: $count[p_{ij}] \leftarrow count[p_{ij}] + 1$
03: return d_{ij}
04: **else**
05: $count[p_{ij}] \leftarrow count[p_{ij}] + 1$
06: **if** $count[p_{ij}] \geq \lambda$
07: send p_{ij} to EAS and
08: receive d_{ij} from EAS
09: add $\{p_{ij}, d_{ij}\}$ to C
10: **end if**
11: return d_{ij}
12: **end if**

[0046]

[0047] **COAC(Cached Open API Cleaning) 알고리즘**

[0048] COAC 알고리즘은 OAG가 시간에 따라 변화하는 오픈 데이터 요청 트래픽에 따라 주기적으로 캐싱 대상을 업데이트하기 위해 활용되는 알고리즘이다. 이는 관리자가 설정한 시간인 N주기로 실행된다. 1줄에서 5줄 사이에 있는 코드는 캐싱되어 있는 오픈 데이터의 횟수을 줄이고 요청 트래픽이 임계치 λ 미만으로 줄어든 경우 캐시에 제거하는 동작을 수행한다.

[0049] [알고리즘 2]

Algorithm 2 COAC

Input: C, λ
Output: $null$
01: **for** c_i **in** C
02: $count[p_{ij}] \leftarrow count[p_{ij}] - 1$
00: **if** $count[p_{ij}] < \lambda$
04: delete $\{p_{ij}, d_{ij}\}$ from C
00: **end if**
05: **end for**

[0050]

[0051] **성능 평가(Evaluation)**

[0052] OAG의 성능을 평가하기 위하여 동기 실험에서 활용한 산림청 산불위험예보정보 오픈 API를 대상으로 성능 분석을 수행하였다. 실험에서는 OTT를 활용하여 1~10초 사이의 랜덤하게 인터벌을 결정하고 API를 요청하는 실험을

진행하였다. 먼저 실험 환경에 대하여 설명하며, 다음으로 API 응답시간 실험 및 결과를 설명한다.

[0053] 실험은 2개의 시스템을 활용하였으며, 하나는 API 게이트웨이 시스템으로 활용하고 나머지는 클라이언트 시스템으로 활용하였다. API 게이트웨이 시스템은 IBM System x3500 M4 서버이며, Intel(R) Xeon(R) CPU E5-2650 CPU, 62 GB 메모리 그리고 1TB 저장소를 가지고 있다. 이는 Linux 4.15 kernel 기반의 CentOS 7를 활용한다. 클라이언트 시스템은 PC이며, Intel(R) Core(TM) i7-7700HQ CPU, 메모리는 16GB, 그리고 256G의 저장소를 가지고 있다. 이러한 시스템은 Linux 5.4 커널 기반의 Ubuntu 18.04를 활용한다. OAG는 Go언어를 활용하였으며, echo 웹 프레임워크 오픈소스를 활용하여 구현하였다. 그리고 오픈 데이터 캐싱을 수행하는 DB는 MongoDB 5.0.8를 도커 컨테이너 형태로 구성하였다.

[0054] OTT를 이용하여 실시간 산림청 산불위험예보정보를 수집하기 위한 오픈 API를 실행하였다. 런타임(runtime)은 약 한시간으로 설정하고 λ는 30으로 설정하고 진행하였다. OTT는 최초 오픈 API 요청 후 OAG롤 통해 최종적으로 전달받은 응답 시간을 측정하였다. 도 4는 API 응답시간 실험의 결과를 나타내고 있다.

[0055] 이상에서 설명된 장치는 하드웨어 구성 요소, 소프트웨어 구성 요소, 및/또는 하드웨어 구성 요소 및 소프트웨어 구성 요소의 집합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성 요소는, 예를 들어, 프로세서, 콘트롤러, ALU(Arithmetic Logic Unit), 디지털 신호 프로세서(Digital Signal Processor), 마이크로컴퓨터, FPA(Field Programmable array), PLU(Programmable Logic Unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(Operation System, OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술 분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(Processing Element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(Parallel Processor)와 같은, 다른 처리 구성(Processing Configuration)도 가능하다.

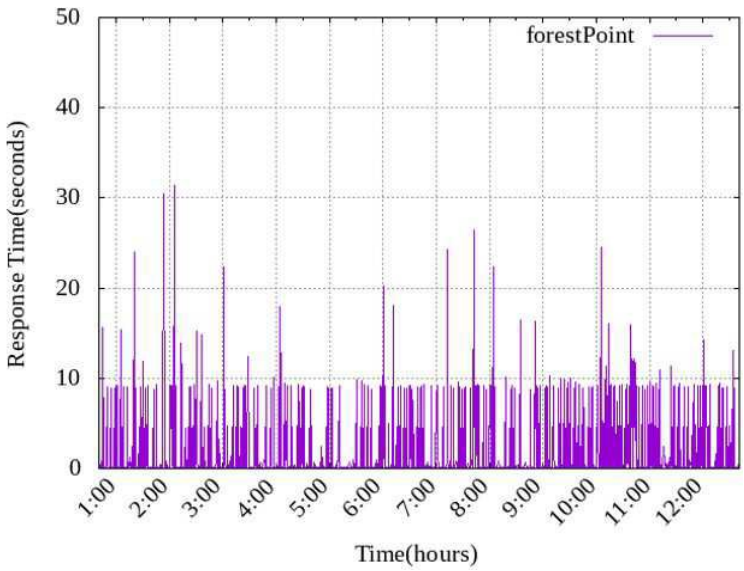
[0056] 소프트웨어는 컴퓨터 프로그램(Computer Program), 코드(Code), 명령(Instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(Collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성 요소(Component), 물리적 장치, 가상 장치(Virtual Equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(Signal Wave)에 영구적으로, 또는 일시적으로 구체화(Embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0057] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM, DVD와 같은 광기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-optical Media), 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

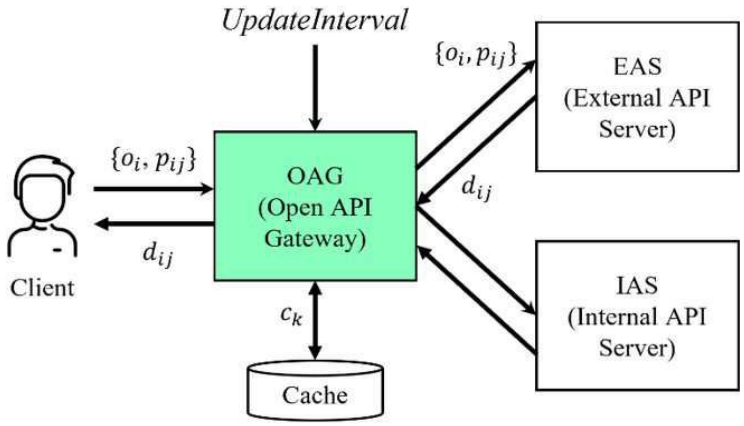
[0058] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성 요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

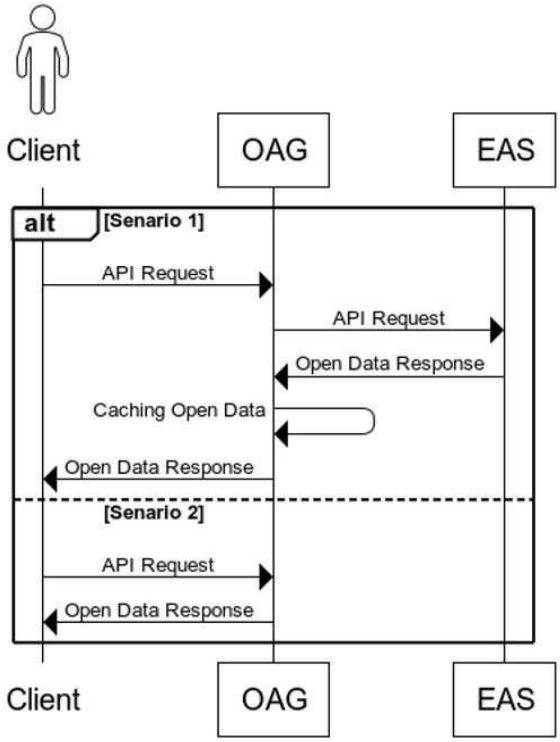
도면1


forestPoint
Response Time(seconds)
Time(hours)
0
10
20
30
40
50
1:00
2:00
3:00
4:00
5:00
6:00
7:00
8:00
9:00
10:00
11:00
12:00

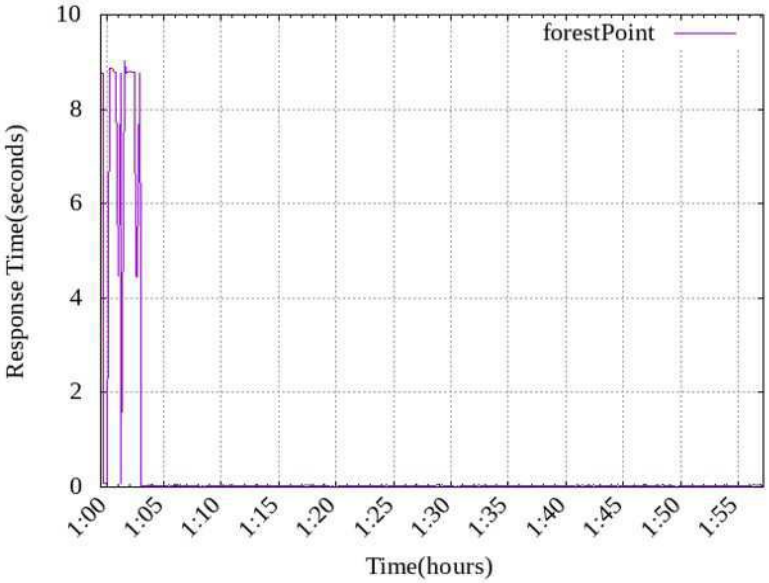
도면2


UpdateInterval
$\{o_i, p_{ij}\}$
EAS
(External API
Server)
$\{o_i, p_{ij}\}$
OAG
(Open API
Gateway)
d_{ij}
d_{ij}
Client
c_k
IAS
(Internal API
Server)
Cache

도면3


Client
OAG
EAS
alt
[Senario 1]
API Request
API Request
Open Data Response
Caching Open Data
Open Data Response
[Senario 2]
API Request
Open Data Response
Client
OAG
EAS

도면4


forestPoint
Response Time(seconds)
Time(hours)
0
2
4
6
8
10
1:00
1:05
1:10
1:15
1:20
1:25
1:30
1:35
1:40
1:45
1:50
1:55