컨테이너 환경에서 GraphQL 기반 고성능 웹 서비스

이강찬*, 김용성*, 최영*, 백승돈*, 송병권*, 김경훈**
*서경대학교, **한국전기연구원

kangchan091@skuniv.ac.kr, wiz@skuniv.ac.kr, cy@skuniv.ac.kr, dd2i@skuniv.ac.kr,
bksong@skuniv.ac.kr, kqh1001@keri.re.kr

High-performance GraphQL-based Web Services in a Container environment

Kangchan Lee*, Yongseong Kim*, Yong Choi*, SeungDon Baik*, Byungkwen Song*,
GyeongHun Kim**

*Seokyeong Univ., **Korea Electrotechnology Research Institute

요 약

현대 웹 환경에서는 실시간 데이터 처리와 확장 가능한 아키텍처의 중요성이 점점 더 강조되고 있다. 기존의 RESTful API 는 불필요한 데이터 전송, 다중 요청 문제, 그리고 실시간 데이터 처리의 제약으로 인해 대규모 데이터 환경에서 효율성이 떨어지는 한계를 보였다. 이러한 문제를 해결하기 위해 등장한 GraphQL 은 필요한 데이터만 선택적으로 요청할 수 있는 유연한 쿼리 방식을 제공하며, 구독(Subscription) 기능을 통해 실시간 데이터 처리까지 가능하게 한다. 특히, Kubernetes 기반 전력망 운영 플랫폼과 같이 대규모의 복잡한 실시간 데이터를 처리하는 환경에서는 이러한 기술이 필수적이다. 본 논문에서는 GraphQL 의 장점을 활용하여 Kubernetes 환경에서 전력망 운영 플랫폼을 위한 고성능 웹 모듈을 설계하고, 확장성과 실시간 데이터 처리 요구를 동시에 충족시키는 컨테이너 기반 고성능 웹 모듈을 제시한다.

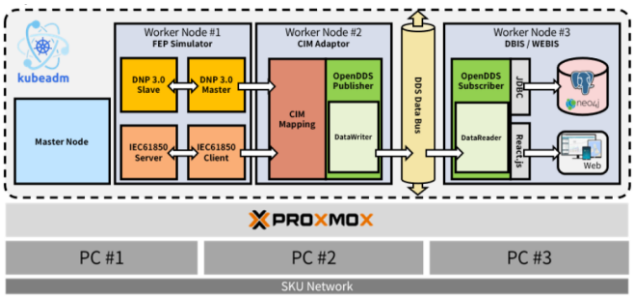
Ⅰ. 서 론

현대 웹 애플리케이션 환경에서는 방대한 데이터를 실시간으로 처리하고 확장 가능한 아키텍처를 제공하는 기술이 필수적이다. 그러나 기존 RESTful API 는 데이터 전송의 비효율성과 실시간 처리의 한계로 인해 대규모 데이터 환경에서 적합하지 않다. 이를 해결하기 위해 GraphQL 이 등장했으며, 통신 효율성과 실시간 데이터 처리의 유연성을 제공하여 RESTful API 의 단점을 보완하고 있다[1][2]. 현대 전력망 운영 플랫폼과 같은 방대한 데이터 처리 환경에서는 복잡한 데이터 요구와 실시간 처리 제약을 해결할 수 있는 통합 솔루션이 필요하다. 본 논문에서는 Kubernetes 기반 컨테이너 환경에서 GraphQL 서버를 활용하여 데이터를 처리하고 웹 애플리케이션을 구현하였다. 제안된 웹 모듈은 데이터를 실시간으로 처리하고 효율적으로 통신할 수 있는 구조를 제공하며, 기존의 한계를 극복하고 현대 전력망 운영 플랫폼과 같은 복잡한 환경에서 실질적인 기여를 할 수 있는 새로운 솔루션을 제시한다.

Ⅱ. 본 론

2.1 컨테이너 기반 고성능 웹 모듈을 위한 환경 구축

현대의 웹 환경은 대규모 트래픽 처리와 실시간 데이터 동기화를 요구하며, 이러한 필요성은 확장성과 안정성을 갖춘 분산 시스템의 설계를 필수 과제로 부각시켰다. 기존의 단일 서버 기반 시스템과 RESTful API 는 자원 활용의 비효율성과 실시간 데이터 처리의 한계로 인해 이러한 요구를 충분히 충족하지 못했다. 이를 해결하기 위해 컨테이너 오케스트레이션 플랫폼 Kubernete 와 유연한 데이터 처리 방식을 제공하는 GraphQL 이 주목받고 있다. Kubernetes 는 컨테이너화된 애플리케이션의 배포, 확장, 관리를 자동화하며, 장애 복구와 자원 최적화를 통해 대규모 환경에서도 안정성과 확장성을 제공한다[3]. GraphQL 은 클라이언트가 필요한 데이터만 요청할 수 있는 쿼리 언어와 웹소켓 기반 실시간 데이터 동기화를 지원하여 RESTful API 의 비효율성을 극복한다. 본 논문에서는 이러한 기술을 결합하여 Kubernetes 기반의 안정적인 클러스터와 GraphQL 의 효율적인 데이터 처리 방식을 활용한 통합 아키텍처를 설계하고, 이를 전력망 운영 플랫폼에 적용하였다. 제시된 시스템은 대규모 트래픽 환경과 실시간 데이터 동기화가 필요한 전력망 환경에서도 확장성과 고성능을 동시에 제공하는 솔루션을 제시한다. 설계된 전체 시스템의 구조는 [그림 1]에 나타나 있다.



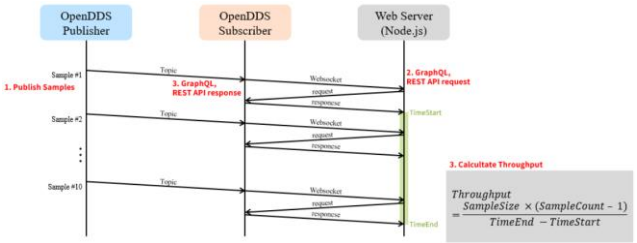
[그림 1] GraphQL based Container Architecture

2.2 GraphQL 구성 및 RESTful API 와 성능 분석

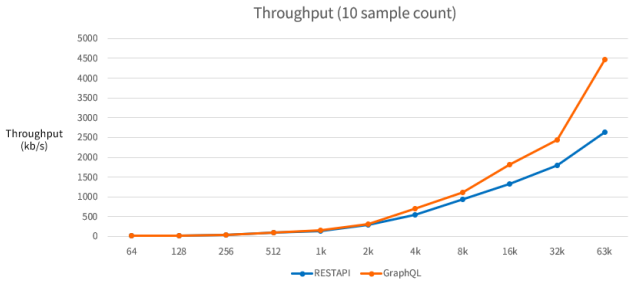
GraphQL 과 RESTful API 의 성능 비교는 동일한 Kubernetes 클러스터 환경에서 진행되었으며, 주요 성능 지표로 Throughput(처리량)을 측정하였다. 이 과정에서

데이터의 구조는 복잡한 계층과 관계를 가지도록 설계되어, 다양한 필드와 참조 관계를 포함한 복잡한 데이터 요청을 처리했다. 이러한 데이터 구조는 현실적인 데이터 처리 환경을 시뮬레이션하며, GraphQL 과 RESTful API 간의 성능 차이를 보다 명확히 비교할 수 있도록 설계되었다. 실험은 OpenDDS Publisher 에서 데이터를 생성하고 Subscriber 로 전송한 후, Web Server 를 통해 클라이언트로 데이터를 전달하는 과정을 기반으로 설계되었다. 각각의 측정 방식과 결과는 다음과 같다.

Throughput(처리량) 측정은 Publisher 에서 데이터를 임의로 생성하여 수신한 후, Subscriber 가 이러한 데이터를 수신하고 데이터의 변경을 동적으로 웹 서버에 알리게 되고, 웹 서버가 이를 인지하여 Subscriber 에 변경된 데이터를 요청하면 Subscriber 가 웹서버의 요청에 대해서 Subscriber 에 전송된 값을 웹서버에 전송한다. 이때 Throughput(처리량)은 데이터를 처음에 전송받은 데이터를 기점으로 마지막까지의 데이터를 송수신하는 시간을 분석한다. [그림 2]는 Publisher, Subscriber 와 웹서버간의 Throughput(처리량) 측정 과정을 시각적으로 나타낸 시퀀스 다이어그램이며, [그림 3]은 이러한 Throughput(처리량) 실험 결과를 그래프로 나타낸 것이다. 실험 결과, GraphQL 은 RESTful API 보다 약 27.7% 높은 Throughput(처리량)을 기록하며 대규모 및 복잡한 데이터 처리 환경에서 뛰어난 성능을 보였다.



[그림 2] Throughput(처리량) 측정 시나리오



[그림 3] Throughput(처리량) 측정 결과

Throughput(처리량) 결과는 GraphQL 의 설계적 장점을 명확히 드러낸다. GraphQL 은 클라이언트가 요청한 데이터만 반환하여 불필요한 데이터 전송을 최소화하고, 단일 엔드포인트 구조를 통해 다양한 데이터 요구를 효율적으로 처리할 수 있다. 반면 RESTful API 는 고정된 데이터 구조와 다중 요청 방식으로 인해 네트워크 부하가 증가하고 지연 시간이 길어지는 한계를 보였다.

이를 기반으로 GraphQL 이 실시간 데이터 처리와 대규모 데이터 통신 환경에서 RESTful API 보다 우수한 성능을 제공하며, 컨테이너 기반 고성능 웹 모듈 설계에서 중요한 기술적 대안임을 입증한다. GraphQL 의 효율적인 데이터 처리 방식은 대규모 데이터 요구와 실시간성을 동시에 충족해야 하는 시스템에 있어 필수적인 선택지가 될 수 있음을 시사한다.

Ⅲ. 결 론

본 연구에서는 Kubernetes 기반의 클러스터 환경에서 GraphQL 서버와 클라이언트를 결합하여 RESTful API 의 한계를 극복하고, 확장성과 실시간 처리가 요구되는 데이터 중심 애플리케이션에 적합한 고성능 웹 서비스 통합 모듈을 설계하고 구현하였다. 제안된 모듈은 Kubernetes 의 자원 관리와 동적 서비스 확장을 통해 안정적인 운영을 지원하며, GraphQL 의 효율적인 데이터 요청 및 구독(Subscription) 기능을 활용하여 실시간 데이터 동기화를 효과적으로 처리함으로써 데이터 통신의 효율성과 확장성을 동시에 확보하였다. 이를 통해 스마트 그리드, IoT, 분산형 에너지 자원 관리와 같은 다양한 실시간 데이터 처리 응용 분야에서 높은 활용 가능성을 보여준다. 향후 연구에서는 제안된 모듈의 이식성과 성능을 다양한 환경에서 추가적으로 검증하고, 보안성과 클라우드 네이티브 특성을 강화하여 클라우드 기반 웹 서비스 모델로서의 활용 가능성을 더욱 높이는 데 중점을 둘 것이다. 또한, 대규모 사용자 요청과 복잡한 데이터 처리 요구를 충족할 수 있는 고성능 확장 모듈을 개발하여 연구의 범위를 확장하고자 한다.

ACKNOWLEDGMENT

본 연구는 산업통상자원부(MOTIE)와 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구 과제입니다.(No. 20225500000060).

참 고 문 헌

[1] G. Brito and M. T. Valente, "REST vs GraphQL: A Controlled Experiment," ASERG Group, Department of Computer Science (DCC), Federal University of Minas Gerais, Brazil, 2020, p. 17.

[2] M. Seabra, M. F. Nazário, and G. Pinto, "REST or GraphQL? A Performance Comparative Study," XIII Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS '19), 2019, pp. 14-16.

[3] Yonghwan Kim, Seongjin Park, and Dongkyun Kim, "Implementation of an open API-based virtual network provisioning automation platform for large-scale data transfer," Journal of the Korea Institute of Information and Communication Engineering, vol. 26, no. 9, pp. 1320- 1329, Sep. 2022, 10.6109/jkiice.2022.26.9.1320.