멀티 클러스터 환경에서의 이벤트 기반 오토스케일링 지원 구조 설계

조재은, 김영한*
*숭실대학교

jayj@dcn.ssu.ac.kr, *younghak@ssu.ac.kr

Design of the Event-driven autoscaling enabled structure in a multi-cluster environment

Cho Jae Eun, Kim Young Han*
*Soongsil Univ.

요 약

클라우드 네이티브 애플리케이션의 발전으로 인해 쿠버네티스 환경은 고도로 분산되고 있다. 이러한 흐름 속에서 최근에는 멀티 클러스터를 효율적으로 운용 및 관리하기 위한 프로젝트들이 생겨나고 있다. 하지만 오토스케일링(AutoScaling) 특히, 이벤트 기반의 오토스케일링 지원 여부는 아직 미흡하다. 필요에 따라 서비스를 빠르게 확장하거나 축소할 수 있는 유연성은 클라우드 컴퓨팅의 대표적인 장점으로 오토스케일링은 이 유연성을 돋보이게 하는 핵심기술이다. 시스템 자원들의 메트릭 값을 모니터링하여 서버의 사이즈를 자동으로 조절하여 예상치 못한 서비스 부하에 효과적으로 대응할 수 있다. 따라서 본 논문에서는 멀티 클러스터 환경에서 이벤트 기반 오토스케일링을 할 수 있도록 하는 구조를 설계했다.

Ⅰ. 서 론

클라우드 네이티브는 클라우드 컴퓨팅 모델을 사용하여 이의 장점을 활용하는 애플리케이션을 개발하고 실행하기 위한 접근 방식이다. 자체 시스템을 구축하는 온프레미스에서 필요한 만큼만 사용하고 비용을 지불할 수 있는 클라우드로의 전환은 비용과 더불어 직접 서버를 유지 보수해야 하는 시간을 절약할 수 있도록 하였다. 하지만 인프라를 잘 구성해 놓고 애플리케이션을 기존 방식으로 구현하여 클라우드의 장점인 확장성과 유연성을 제대로 활용하지 못하는 경우가 많다. 예를 들어 클라우드는 특정 애플리케이션에 트래픽이 증가할 때 유연하게 확장해 대응할 수 있는데 애플리케이션이 한 덩어리로 구성되어 있으면 애플리케이션 전체를 늘려서 확장해야 한다는 문제점이 있다. 이는 결국 유연성을 활용하지 못할 뿐만 아니라 비용 증가로도 이어진다. 애플리케이션을 하나로 만드는 대신 업무 영역별, 기능별로 쪼개 개발하는 마이크로서비스 구조를 활용하면 클라우드에 최적화된 클라우드 네이티브 애플리케이션을 만들 수 있다[1].

마이크로서비스를 구현하는 기술로 가장 주목받는 것이 컨테이너이다. 쿠버네티스는 컨테이너화 된 애플리케이션을 자동으로 배포, 스케일링 및 관리해 주는 오픈소스 시스템으로 클라우드 네이티브 애플리케이션의 발전으로 인해 쿠버네티스 환경은 더욱더 분산되고 있다. 기업에서는 개발, 테스트, 프로덕션 등으로 클러스터를 분산시킨 멀티 클러스터를 운용하기도 한다. 이러한 흐름 속에서 최근에는 멀티 클러스터를 효율적으로 운용 및 관리하기 위한 프로젝트들이 생겨나고 있다. 하지만 오토스케일링(AutoScaling) 특히, 이벤트 기반의 오토스케일링 지원 여부는 아직 미흡하다. 오토스케일링은 클라우드 컴퓨팅의 대표적인 장점인 유연성을 돋보이게 하는 핵심기술이다. CPU, 메모리, 디스크, 네트워크 트래픽과 같은 시스템 자원들의 메트릭(Metric) 값을 모니터링하여 서버의 사이즈를 자동으로 조절하여 예상치 못한 서비스 부하에 효과적으로 대응할 수 있다. 멀티 클러스터 프로젝트에는 아직 이러한 기능이 부족하기 때문에 본 논문에서는 멀티 클러스터 환경에서 이벤트 기반 오토스케일링을 할 수 있도록 하는 구조를 설계했다.

Ⅱ. 관련 연구

i. GitOps

프로젝트에 데브옵스를 적용하는 방법의 하나이다. 애플리케이션의 배포와 운영에 관련된 모든 요소를 코드화하여 Git 에서 관리(Ops)한다. 선언형 기술서(Declarative Descriptions) 형태로 작성된 코드들을 Config Repository 에서 관리하여 이 선언형 기술서와 운영 환경 간 상태 차이가 없도록 유지해주는 자동화 시스템을 구성한다[2].

ii. Fleet

GitOps 를 멀티 클러스터로 확장할 수 있도록 하는 프로젝트이다. 유닛을 애플리케이션이 아닌 번들 단위로 배포하며 클러스터별로 개별화하여 헬름 차트로 배포한다[3].

iii. HPA(Horizontal Pod Autoscaler)

쿠버네티스에서 제공하는 오토스케일러 HPA 는 CPU 사용량 또는 메모리와 같은 항목들을 모니터링하여 POD 의 개수를 자동으로 확장한다[4]. 사용할 수 있는 메트릭의 수가 제한적이고 POD 의 개수를 0 개로 줄이는 것이 불가하다.

iv. KEDA(Kubernetes-based event-driven autoscaling)
이벤트 기반 컨테이너 오토스케일링을 지원하는 쿠버네티스용 오픈소스 컴포넌트이다. CPU 와 메모리 기반 확장이 아닌 이벤트 큐 기반 확장이 가능하며, 이벤트가 없을 때 POD 의 개수를 0 개로 줄일 수 있다[5].

Ⅲ. 본 론

Fleet 프로젝트를 이용하여 멀티 클러스터를 구성하고, 각 클러스터 안에 KEDA 컴포넌트를 설치하여 이벤트 기반 오토스케일링이 가능한 구조를 설계하였다. Fleet 에서 각 클러스터 그룹은 지역(region) 또는 개발 환경 등으로 구성할 수 있는데 본 논문에서는 개발 환경으로 구성하였고, 각 개발 환경 내에서도 클러스터별로 연관된 서비스를 묶어서 설계하였다.

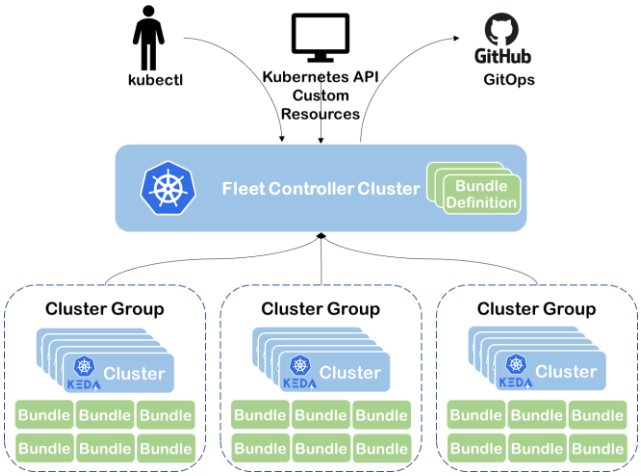

그림 1. 멀티 클러스터 환경에서
오토스케일링을 하기 위한 전체 구조

클러스터 그룹을 프로덕션 환경이라 생각했을 때, 그 안에서도 연관된 서비스를 묶어서 Users, Alarm, Posts 등의 역할을 하는 클러스터를 구성하였다. Users 에 관련된 번들은 Users 클러스터에 배포하고 Posts 에 관련된 번들은 Posts 클러스터에 배포하여 ScaledObject 를 생성한다. ScaledObject 는 KEDA 에서 사용되는 리소스로 레플리카의 개수나 트리거 조건 등을 설정할 수 있다. KEDA 컴포넌트에서 이벤트 소스를 지속적으로 읽어와서 이벤트가 발생했을 때 POD 의 개수를 늘린 후 이벤트가 종료되면 0 개로 줄인다.

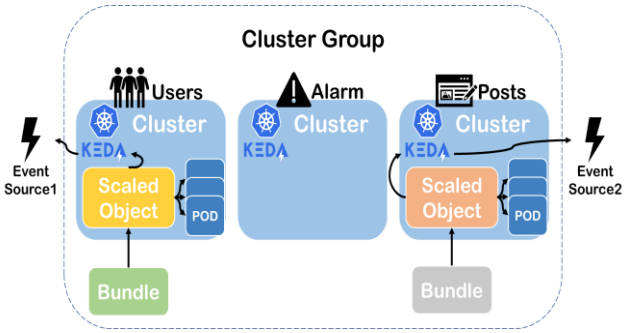

그림 2. 클러스터 그룹 안에서 이벤트 기반
오토스케일링 동작 과정

본 논문의 구조에서는 클러스터별로 연관된 서비스를 묶어서 분리하였지만, 여러 클러스터에 걸쳐서 같은 번들을 배포하고 싶을 경우에는 몇 가지 과정이 추가되어야 한다. 우선 동일한 이벤트 소스를 관찰하도록 해야 하고, 이를 확장할 때 어느 클러스터에 얼마나 확장해야 할지를 나누는 기준을 결정해야 한다. 각 클러스터의 리소스를 고려하고자 하면 리소스 모니터링이 별도로 필요할 것이다. 또한 이를 나눌 수 있는 로드밸런서를 어떻게 둘 것인지도 생각해야 한다.

Ⅳ. 결 론

본 논문에서는 오토스케일링과 멀티 클러스터 구조에 대해서 각각 서술하고 멀티 클러스터 환경에서 이벤트 기반 오토스케일링 지원 여부가 아직 미흡함을 설명하였다. 따라서 이를 보완하기 위해 Fleet 프로젝트를 이용하여 멀티 클러스터를 구성하고, KEDA 컴포넌트를 이용하여 이벤트 기반 오토스케일링이 가능한 구조를 설계하였다. 각 클러스터 그룹을 개발 환경으로 나누고, 개발 환경 내에서도 클러스터별로 연관된 서비스끼리 분리하여 스케일링 할 수 있도록 하였다. 하지만 본론에서도 서술했다시피 여러 클러스터에 걸쳐 같은 번들을 배포하고 싶을 경우 몇 가지 과정이 추가되어야 하고 로드밸런서를 배치하여 해결할 수 있지만 적절한 위치를 찾는 것이 필요하다. 또한, 두 가지 프로젝트를 모두 알아야 한다는 전문성이 요구되기 때문에 추후 연구에서는 오토스케일링이 가능한 멀티 클러스터의 구성 방법을 단순화하는 것을 목표로 진행할 계획이다.

ACKNOWLEDGMENT

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2020-0-00946, 하이브리드 클라우드 환경에서의 고속, 자동 서비스 복구 및 이전 소프트웨어 개발)

참 고 문 헌

[1] 장민, “마이크로서비스와 컨테이너로 완성하는 ‘클라우드 네이티브’ 애플리케이션 전략,” IDG Summary, pp. 1-4, 2020.

[2] Beetz, Florian. Kammer, Anja. Harrer, Simon. “GitOps,” 2020, (https://www.gitops.tech/).

[3] Fleet. “Fleet – GitOps at Scale,” 2020, (https://fleet.rancher.io/).

[4] Kubernetes, “Horizontal Pod Autoscaler,” 2021, (https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/).

[5] KEDA. “KEDA Concepts Latest,” 2021, (https://keda.sh/docs/2.2/concepts/).