센서 네트워크에서의 쿠버네티스를 활용한 클라우드 기반 센서 데이터 수집 시스템 설계

박수용*, 문주현*, 박슬우**, 신용태**
*숭실대학교 컴퓨터학과
**숭실대학교 컴퓨터학부
e-mail:waterdragon@soongsil.ac.kr

Design of Cloud-based Sensor Data Acquisition System Using Kubernetes in Sensor Networks

Soo-Yong Park*, Ju-Hyeon Moon*, Seul-Woo Park**, Yong-Tae Shin**
*Dept of Computer Science, Soongsil University

요 약

센서 네트워크는 스마트 시티와 같은 4차 산업혁명 분야의 핵심기술로 다양한 분야에 활용되고 있다. 기존의 센서 네트워크는 여러개의 센서 노드가 한 개의 싱크 노드를 통해 인터넷으로 데이터를 전달하였다. 그러나 과도한 트래픽 또는 외부적인 요인으로 인해 싱크 노드가 중지될 경우 그 싱크노드와 연결된 센서 노드로부터 데이터를 수집하지 못하는 단점을 가지고 있다. 제안하는 시스템은 도커를 사용하여 싱크노드를 컨테이너화 하고 쿠버네티스를 통해 중지된 컨테이너를 자동으로 재시작하여 시스템의 안정성을 높일 수 있다.

1. 서론

센서 네트워크는 정보의 수집이나 모니터링이 필요한 지역에 센서를 배치하고 배치된 센서를 통해 데이터를 수집하고 이를 응용하기 위한 기술이다. 현재 센서 네트워크는 제조, 농축산업, 스마트 시티 등의 분야에 다양하게 활용되고 있다.

기존의 센서 네트워크는 센서 노드와 싱크 노드로 구성된다. 한 개의 싱크 노드에는 다수의 센서 노드가 연결되어 있다. 센서 노드의 이벤트가 비정상적으로 증가하거나 어떤 외부적인 요인으로 인해 싱크 노드가 중지될 경우 그 싱크 노드와 연결된 모든 센서 노드로부터 데이터를 전송받지 못하는 단점을 가지고 있다.

제안하는 시스템은 도커 오케스트레이션 툴인 쿠버네티스를 이용하여 한 개의 마스터 노드와 다수의 슬레이브 노드를 생성하여 클러스터를 구축한다. 또한 도커를 이용하여 싱크 노드의 역할을 하는 이미지를 생성한다. 마스터 노드는 생성된 싱크 노드 이미지를 컨테이너화 하여 각각의 슬레이브 노드에 적재시키고 활성화하여 데이터를 수집할 수 있도록 한다. 제안하는 시스템에서 슬레이브 노드 또는 싱크 노드 컨테이너가 다운될 경우 쿠버네티스에서 자동으로 싱크 노드 컨테이너를 재시작한다. 제안하는 시스템은 쿠버네티스 클러스터에서 모든 슬레이브 노드에 문제가 발생하지 않는 한 지속적으로 센서 노드의 데이터를 전송받을 수 있다.

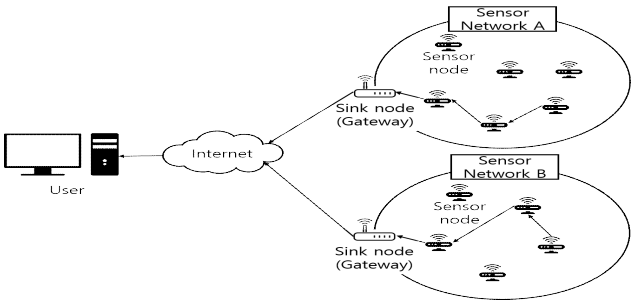
본 논문의 구성은 다음과 같다. 2장 관련 연구에서는 센서 네트워크에서의 쿠버네티스를 활용한 클라우드 기반 센서 데이터 수집 시스템 설계에 필요한 기반 기술에 대하여 연구한다. 3장 시스템 설계에서는 기반 기술을 바탕으로 데이터 수집 시스템을 설계한다. 4장 결론에서는 설계된 시스템을 바탕으로 결론을 도출한다.

2. 관련 연구

본 장에서는 센서 네트워크에서의 쿠버네티스를 활용한 클라우드 기반 데이터 수집 시스템 설계에 필요한 기반 기술인 센서 네트워크, 도커, 쿠버네티스에 대하여 연구한다.

2.1. 센서 네트워크

센서 네트워크는 온도, 기압 등과 같은 환경 조건뿐만 아니라 제조 공정, 교통, 건설 등 다양한 분야에 활용되고 있다[1]. 센서 네트워크는 크게 센서 노드와 싱크노드로 구성된다. (그림 1)은 센서 네트워크의 구성을 나타낸다.



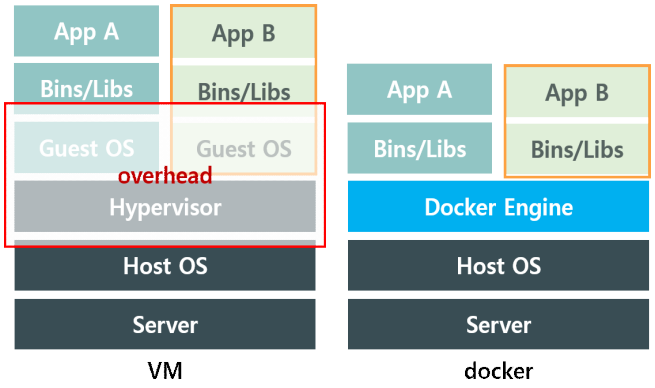
(그림 1) 센서 네트워크의 구성

센서 노드는 정보 수집이 필요한 지역에 망형으로 배치되어 데이터를 센싱하고 싱크 노드로 전송하는 역할을 한다. 싱크 노드는 센서 노드로부터 데이터를 전달받고 외부로의 게이트웨이 역할을 수행한다. 센서 노드는 싱크 노드로 데이터를 전송할 때 다른 센서 노드를 통해 전달하기 때문에 넓은 범위의 통신이 가능하다. 센서 네트워크는 센서의 종류별 또는 네트워크별로 그룹을 지어 각각 다른 싱크 노드가 담당하게 구축할 수 있다.

센서 네트워크에서의 무선 통신에는 Zigbee가 사용된다. Zigbee는 IEEE 802.15.4-2003을 기반으로 단거리 라디오 주파수를 사용하는 무선통신망의 표준이다[2].

2.2. 도커

도커는 컨테이너 기반의 오픈소스 가상화 플랫폼이다. 컨테이너는 호스트 컴퓨터의 컴퓨팅 파워를 격리된 공간에 가상화하여 동작하는 기술이다. (그림 2)는 기존의 가상화 방식과 도커의 가상화 방식을 나타낸다.



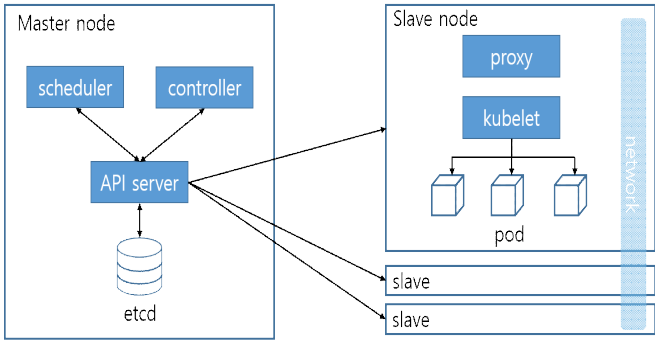
(그림 2) 기존의 가상화 방식과 도커의 가상화 방식

기존의 가상화 방식은 Hypervisor를 이용하여 호스트 OS 위에 게스트 OS 전체를 가상화하여 사용하는 방식이다. 게스트 OS 전체를 가상화하는 방식은 여러 OS를 사용할 수 있고 비교적 사용법이 간단하지만 가상화 이미지가 무겁고 느리기 때문에 운영환경에서 사용하기엔 어려움이 있다. 도커는 컴퓨터의 프로세스를 격리하는 방식인 리눅스 컨테이너 기술을 사용하여 호스트 OS 위에 게스트 OS를 설치하지 않고 Hypervisor 대신 Docker Engine 위에서 동작하기 때문에 가볍고 빠른 속도를 제공한다[3].

도커는 이미지라는 개념을 제시하고 있다. 도커 이미지는 컨테이너 실행에 필요한 파일과 설정값 등을 포함하고 있다. 컨테이너는 이미지를 실행한 상태라고 정의할 수 있고 컨테이너의 상태가 바뀌거나 컨테이너가 삭제되더라도 이미지는 변하지 않고 남아있다. 예를 들어 ubuntu 이미지는 ubuntu를 실행하기 위한 모든 파일을 가지고 있고 하나의 ubuntu 이미지를 사용하여 여러 개의 컨테이너를 생성할 수 있다.

2.3. 쿠버네티스

쿠버네티스는 도커 오케스트레이션 툴이라고도 불리며 컨테이너를 빠르게 배포 및 확장하고 관리를 자동화해주는 오픈소스 플랫폼이다. 쿠버네티스는 구글이 내부적으로 사용하던 컨테이너 오케스트레이션 툴을 기반으로 2014년 프로젝트를 시작했다[4]. (그림 3)은 쿠버네티스의 구조를 나타낸다.



(그림 3) 쿠버네티스의 구조

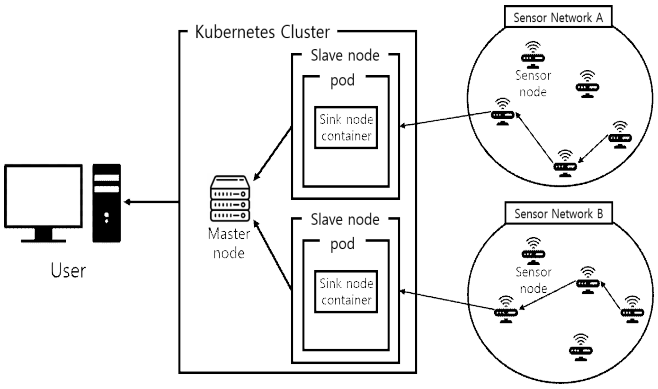
쿠버네티스는 마스터 노드와 슬레이브 노드로 구성되며 마스터 노드에서 API 서버를 통해 슬레이브 노드와 통신하는 단순한 구조를 가지고 있다.

슬레이브 노드는 pod, kubelet, proxy 모듈로 구성된다. pod 모듈은 쿠버네티스 애플리케이션의 기본 실행 단위이다. 쿠버네티스 상에서 동작할 실제 애플리케이션은 컨테이너화 되어 pod에 포함된다. pod에는 단일 컨테이너를 사용하는 애플리케이션뿐만 아니라 여러 개의 컨테이너를 사용하는 애플리케이션을 실행시킬 수 있다. 쿠버네티스는 컨테이너가 아닌 pod를 직접 관리한다. kubelet 모듈은 pod의 생명주기를 관리한다. pod의 생성 또는 삭제를 담당하며 pod에서 실행되고 있는 컨테이너에 이상이 없는지 주기적으로 확인한다. proxy 모듈은 pod 모듈의 네트워크를 관리하며 클러스터의 외부 IP 또는 port 번호로 접근할 수 있게 한다.

마스터 노드는 API server, etcd, scheduler, controller 모듈로 구성되어 있다. API server는 다른 모듈 또는 슬레이브 노드의 pod의 요청을 처리하는 역할을 한다. API server는 다른 모듈의 요청 처리 시에 권한을 확인하여 요청을 거부할 수 있는 기능이 있다. etcd 모듈은 key-value 기반의 저장소로 클러스터의 모든 설정, 상태 데이터를 저장하고 있다. etcd 모듈은 watch 기능을 포함하고 있기 때문에 상태가 변경될 경우 바로 감지하고 적용할 수 있다. scheduler 모듈은 새로 생성된 pod를 감지하고 어느 슬레이브 노드에 pod를 배치할 것 인지를 결정하는 역할을 한다. pod의 배치에는 CPU 사용량, 저장공간 등을 고려하여 결정한다. controller 모듈은 쿠버네티스의 다른 모듈들의 상태를 관리하는 역할을 한다.

3. 시스템 설계

본 논문에서 제안하는 시스템의 목적은 센서 네트워크에서의 싱크 노드를 컨테이너화 하고 컨테이너 오케스트레이션 툴인 쿠버네티스를 이용하여 관리하는 것이다. (그림 4)는 제안하는 시스템의 구조를 나타낸다.



(그림 4) 제안하는 시스템의 구조

시스템 설계를 위해서는 다음의 세 단계를 거쳐야 한다. 첫 번째로 도커를 사용하여 싱크 노드를 이미지화하고 물리적인 클러스터를 구성해야 한다. 두 번째 단계는 한 개의 마스터 노드와 기존 센서 네트워크의 싱크 노드의 개수만큼의 슬레이브 노드를 생성한다. 마지막 단계는 마스터 노드는 pod를 사용하여 싱크 노드 이미지를 컨테이너화 하여 각각의 슬레이브 노드에 배포한다. 배포된 pod는 IP 기반으로 동작하기 때문에 센서 노드들은 기존의 설정을 유지한 상태로 데이터를 전송할 수 있다.

싱크노드를 이미지화하는 단계에서는 Dockerfile.yaml 파일을 생성하여 이미지에 대한 정보를 기록하고 docker 빌드 명령어를 통해 최종적으로 이미지를 빌드한다. (그림 5)는 이미지 빌드를 위한 DokcerFile.yaml의 소스코드를 나타낸다.

```
1  #싱크 노드 이미지의 베이스 이미지 설정
2  FROM ubuntu:16.04
3  #베이스 이미지에 서버 설치
4  RUN apt-get install  - y install nginx
5  #포트 지정
6  EXPOSE 80
7  #서버 실행
8  CMD [“nginx”, “-g”, “demon off;”]
```

(그림 5) DockerFile.yaml의 소스코드

(그림 5)의 2라인의 코드는 16.04 버전의 ubuntu 기반으로 이미지를 빌드한다는 의미이며 운영하고자 하는 환경에 따라 다른 리눅스 운영체제를 사용할 수 있다. (그림 5)에서는 센서 노드가 전송한 데이터를 수신할 서버를 nginx를 사용한다.

쿠버네티스의 클러스터를 구축하는 단계에서는 마스터 노드와 모든 슬레이브 노드에 쿠버네티스를 설치하고 마스터 노드에서 쿠버네티스를 실행시키면 클러스터에 슬레이브 노드를 추가시킬 수 있는 명령어가 출력된다. 이 명령어를 슬레이브 노드에서 실행함으로써 클러스터를 구축할 수 있다.

싱크 노드 컨테이너가 마스터를 통해 pod에 배포된 후에 마스터 노드의 scheduler 모듈이 pod의 상태를 주기적으로 확인하기 위해서는 배포 시에 추가적인 프로브 설정이 필요하다. 프로브는 쿠버네티스에서 제공하는 상태 검사 서비스이다. 쿠버네티스의 pod는 yaml 파일을 통해 배포되는데 yaml 파일에 프로브 관련 설정을 추가할 수 있다. 프로브는 컨테이너 별로 설정된다. (그림 6)은 프로브가 설정된 yaml 파일을 나타낸다.

```
1  containers:
2  #프로브 설정
3  livenessProve:
4  #상태를 확인할 때 http의 Get 통신을 사용
5  httpGet:
6  #컨테이너에 상태 확인 메시지를 보낼 주소
7  path: /prove
8  #컨테이너에서 프로브에 응답할 포트
9  port: 8080
10 #컨테이너가 생성되고 5초후에 프로브 시작
11 initialDelaySeconds: 5
12 #컨테이너로부터 응답을 받는 시간 설정
13 timeoutSeconds: 1
14 #컨테이너의 상태를 확인하는 주기
15 periodSeconds: 10
16 #컨테이너의 응답 실패 허용 횟수
17 failureThreshold: 3
```

(그림 6) 프로브가 설정된 yaml 파일

프로브는 http 통신을 이용하여 컨테너의 상태를 확인하기 때문에 싱크노드에서도 프로브의 http 요청에 대한 응답을 하는 기능이 추가적으로 구현되어야 한다. 프로브는 (그림 6)의 17번째 라인에 설정한 대로 컨테이너로부터 3번 이상 응답이 없을 경우 해당 컨테이너의 상태를 비정상으로 판단하고 컨테이너를 종료 후 재시작한다. 컨테이너가 종료 후 재시작 된다 하더라도 종료된 시간동안에는 센서 노드로부터 데이터를 전송받을 수 없다. 쿠버네티스에서는 레플리카셋을 제공한다. 레플리카셋은 특정 수의 동일한 pod가 동시에 구동되도록 하며 그 수를 항상 유지한다. 따라서 레플리카셋을 구축하면 컨테이너가 중지하고 재시작하는 동안에 복제된 다른 pod가 센서 노드의 데이터를 처리할 수 있기 때문에 시스템의 안정성을 확보할 수 있다.

컨테이너는 기본적으로 데이터를 컨테이너 내부에 저장소에 저장한다. pod가 삭제되거나 컨테이너가 재시작되면 컨테이너의 파일 시스템에 있는 모든 데이터는 삭제된다. 따라서 컨테이너에 저장된 센서 데이터를 노드에 있는 영구 저장소로 옮길 필요가 있다. 쿠버네티스에는 볼륨 마운트 기능이 있다. 볼륨 마운트 기능은 컨테이너 내부 저장소와 노드의 영구 저장소를 동기화시켜서 항상 같은 데이터를 저장하도록 하는 역할을 한다. 볼륨 마운트 기능은 프로브와 같이 yaml 파일에 설정한다. (그림 7)은 yaml 파일에 볼륨 마운트 기능 설정을 나타낸다.

```
1  #볼륨 설정
2  volumes:
3  #볼륨의 이름설정
4  - name: "sensor-volume"
5  #영구 저장소의 경로 설정
6  hostPath:
7  path: "/var/lib/sensor"
8  #볼륨과 컨테이너 내부 저장소를 마운트
9  volumeMounts:
10 - mountPath: "/data"
11 name: "sensor-volume"
```

(그림 7) yaml 파일에 볼륨 마운트 설정

위의 모든 설정이 완료되면 쿠버네티스의 명령어를 이용하여 바로 pod를 배포한다. pod 배포 순간부터 시스템은 동작을 하며 센서로 부터 데이터를 수집할 수 있다.

4. 결론

본 논문에서는 센서 네트워크에서의 쿠버네티스를 활용한 클라우드 기반 센서 데이터 수집 시스템을 설계 하였다. 센서 네트워크의 싱크 노드를 컨테이너화 하여 쿠버네티스로 관리 할 수 있다. 쿠버네티스는 컨테이너의 동작 상태를 실시간으로 감지하여 이상이 있는 컨테이너를 중지시키고 재시작할 수 있다. 따라서 싱크 노드 컨테이너 또는 쿠버네티스 클러스터의 슬레이브 노드에 이상이 발생하여도 쿠버네티스에서 컨테이너를 재시작하기 때문에 센서 노드로부터 데이터를 지속적으로 전송 받을 수 있다.

Acknowledgement

본 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.IITP-2019-0-00135 ,ICT 기반 환경 모니터링 센서 신뢰성 검증 및 평가 플랫폼)

참고문헌

[1] 박덕신, 윤영훈, 봉춘근 "환경 센서 및 센서네트워크 분야의 최근 연구동향", 한국대기환경학회지, 제27권, 제2호, pp.214-224, 2003

[2] 최경준, 최세영, 조영식 "Zigbee 기반 무선 센서 네트워크 소형화 설계 및 제작", 한국통신학회학술대회논문집, pp.1139-1140, 2019

[3] 임인구, 진현욱 "도커 컨테이너의 멀티코어 CPU 대역폭 영향 분석", 정보과학회컴퓨팅의실제논문지, 제24권, 제12호, pp.675-680, 2018

[4] Chia-Chen Chang, Shun-Ren Yang, En- Hau Yeh, Phone Lin, Jeu-Yih Jeng "A Kubernetes-Based Monitoring Platform for Dynamic Cloud Resource Provisioning", IEEE BLOBECOM, 2017