Celery-MongoDB를 활용한 센서정보 관리시스템 설계 및 구현

[1]강윤희

Design and Implementation of Sensor Information Management System based on Celery-MongoDB

[1]Yun-Hee Kang

요 약

센서정보 관리를 위해서는 다양하고 수많은 센서의 정보를 신속하게 저장, 수정, 삭제 할 수 있는 기능을 제공해야 한다. 본 연구에서는 Celery 와 MongoDB 를 활용하여 위의 조건에 부합한 센서정보 관리 시스템을 설계 및 구현하였다. Celery 는 파이썬으로 개발된 비동기 통신을 기반으로 하는 큐구조를 제공하고 있다. 그리고 이것은 분산된 작업 큐 구조이고 단순하지만 많은 양의 메시지를 처리하기에 적합한 신뢰성 있는 분산 시스템이다. MongoDB 는 NoSQL 데이터베이스로써 다양한 정보 표현을 저장할 수 있고 검색할 수 있다. 본 연구에서는 개발한 시스템을 활용한 실험을 통해 IoT 환경에서 제공되는 다양한 센서를 관리할 수 있음을 확인할 수 있었다. 센서데이터를 갖는 메시지를 처리하기 위한 성능을 개선하기 위해 본 시스템은 클라우드 하부구조의 에지단에 배치되어 사용한다.

Abstract

The management of sensor information requires the functions for registering, modifying and deleting rapidly sensor information about various many sensors. In this research, Celery and MongoDB are used for developing a sensory data management system. Celery supplies a queue structure based on asynchronous communication in Python. Celery is a distributed simple job-queue but reliable distributed system suitable for processing large message. MongoDB is a NoSQL database that is capable of managing various informal information. In this experiment, we have checked that variety of sensor information can be processed with this system in a IoT environment. To improve the performance for handling a message with sensory data, this system will be deployed in the edge of a cloud infrastructure.

Keywords: *Sensor information, Asynchronous communication, Queue, NoSQL database, Python*

[1] *백석대학교 컴퓨터공학부 교수 (yhkang@bu.ac.kr)*

Received: Apr. 28, 2021, Revised: Jun. 01, 2021, Accepted: Jun. 24, 2021

I. 서론

IoT(Internet of Things)는 4 차 산업혁명의 주요 핵심기술로 등장하였으며, 이에 따라 센싱 기술을 활용한 분야가 확대되고 있다. 이러한 센서 데이터 활용 영역이 넓어지면서 데이터 분석 서비스가 활성화되고, 분석을 용이하게 할 수 있는 환경으로 진화하고 있다[1]. IoT 환경이 사회 전반에 보급됨에 따라 수많은 디바이스로부터 생성되는 센서와 센서데이터에 대한 체계적 관리에 대한 필요성이 증대하고 있다[2,3,4].

센서에서 발생하는 데이터는 시계열의 특징을 갖는다. 시계열 데이터는 일정 시간 간격으로 배치된 숫자 데이터들의 나열이다. 시계열 형태의 자료는 다양한 디바이스 연결에 적절히 대응하기 위해 데이터 처리의 계층화가 필요하며 구체적으로 상위 프로세스에서 데이터 형식을 지정하고 앞 단계에서 수신된 데이터를 정해진 데이터 형식으로 변환하여야 한다 [5,6].

센서데이터 관리 시스템은 디바이스에서 생산되는 센서데이터를 응용프로그램에서 소비되도록 안정적이고 안전하게 제공해야 한다. 또한 응용프로그램 입장에서는 사용하기 편리한 인터페이스가 필요하다. 현재 AWS IoT 를 활용한 클라우드 기반의 센서데이터 관리시스템이 많이 활용되고 있다. 그러나 AWS IoT 는 센서 데이터 관리 기능만 제공하고 있으며 센서에 관한 정보를 관리할 수 있게 하는 기능은 제공하고 있지 않다. 응용프로그램이 클라우드에서 제공되는 센서데이터를 활용하기 위해서는 센서정보를 체계적으로 관리하여야 한다.

본 논문에서는 센서정보에 대한 등록, 검색, 수정 및 삭제 기능을 갖춘 센서정보 관리시스템을 설계하고 구현하였다. 구현을 위해 분산처리 프로세스인 Celery 와 MongoDB 를 이용하여 효율적으로 센서 데이터를 관리한다. 센서에 관한 정보는 센서의 종류에 따라 다양하기 때문에 정형화되지 않은 경우가 많다. 따라서 본 논문에서는 관계형 데이터베이스대신 NoSQL 기반의 데이터베이스인 MongoDB 를 이용하였다. 그리고 많은 수의 클라이언트가 센서정보 관리시스템을 접속하고 이용할 수 있도록 Celery 를 채용하였다. Celery 는 세부적인 메시징을 숨긴 핵심 기능 태스크를 제공함으로 많은 수의 클라이언트가 동시에 사용 가능하다.

본 논문의 구성은 다음과 같다. 2 절에서는 관련 연구로써 본 연구에서 사용한 Celery, MongoDB 의 개념과 구조를 소개한다. 3 절에서는 본 연구에서 설계 및 구현한 센서정보 관리시스템의 구조 및 기능을 기술할 것이다. 마지막으로 4 절에서는 구현된 시스템의 사용방법을 소개한 후 5 절에서 결론을 제시한다.

II. 관련 연구

2.1 Celery

센서정보 관리시스템에서 동기 통신을 이용하여 센서정보를 수집하면 정보를 주고받는데 응답대기 시간이 발생하여 응용프로그램의 필요를 만족시키기 어렵다. IoT 환경에서는 빈번하게 센서정보 통신을 해야 하기 때문에 비동기통신이 적합하다. Celery [7,8]는 웹 서비스를 이용하여 비동기 작업 처리를 지원하는 통신 서비스를 제공하는 파이썬 프레임워크이다. 그림 1 은 Celery 시스템 구조를 보인 것이다.

Celery 사용자는 응용과의 요청과 응답 상호작용을 수행한다. Celery 에서 구동하는 응용프로그램은 복수의 작업들로 구성되어 있고 각 작업을 브로커에서 전달하면 워커가 작업을 처리하는 구조이다. 작업 요청을 받을 브로커는 큐구조로 작업 요청을 저장하고 먼저 도착한 요청부터 워커에 분배한다. 브로커로 사용되는 도구로는 Rabbitmq, Redis, Amazon SQS 등이 있다. 이들 중 Rabbitmq 가 기본 브로커로 등록되어 있다. Celery 시스템은 여러 작업자와 브로커로 구성되어 수평 확장이 가능하고 고가용성을 제공한다. 또한 사용 및 유지 관리가 쉽고 구성 파일이 필요하지 않기 때문에 단순하게 사용할 수 있다.

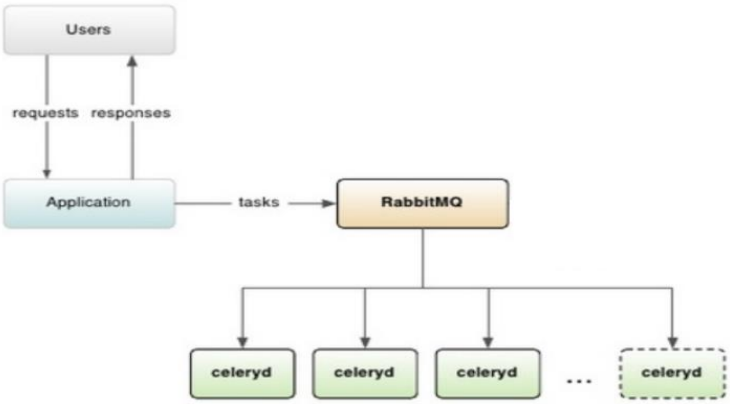

Figure 1. Celery System

2.2 MongoDB

MongoDB 는 JSON object 형식의 문서들을 저장할 수 있는 NoSQL 데이터베이스이다 [9]. NoSQL 이란 Not Only SQL 의 약자로서 기존의 관계형 데이터베이스의 한계를 극복하기 위한 새로운 형태의 데이터베이스이다 [10]. MongoDB 의 데이터베이스 구조는 Figure 2 와 같다. 그림 2 에서 문서(document)는 실질적인 데이터를 의미하며 관계형 데이터베이스의 레코드와 유사한 개념이다. 차이점으로는 관계형 데이터베이스의 레코드는 스키마에 따라 모두 같은 형태로 되어 있지만 MongoDB 의 문서는 JSON object 형태인 키-값 쌍으로만 이루어지면 될 뿐 문서끼리 같은 형태를 강요하지 않는다. 하나의 컬렉션(collection) 에는 복수의 문서들이 존재하며, 데이터베이스는 복수의 컬렉션들로 이루어진다. 컬렉션에 삽입되어지는 자료는 키와 값으로 구성된다. 또한 성능을 고려하여 JSON 문서를 바이너리로 인코딩한 포맷인 BSON(Binary JSON)을 사용한다.

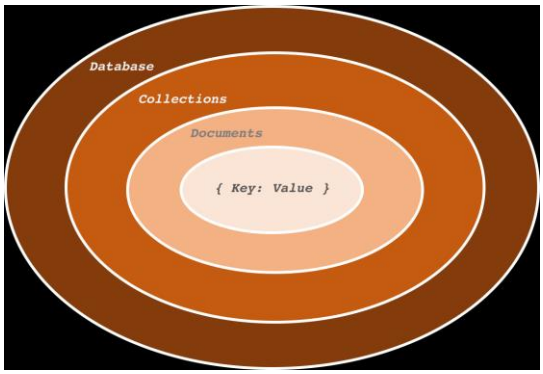

Figure 2. MongoDB Database Structure

III. 센서정보 관리 시스템의 구조 및 기능

3.1 시스템 구조

센서데이터가 수집되기 전에 먼저 사용할 센서에 대한 정보 등록이 이루어져야 한다. 본 연구에서는 이를 위해 센서정보 관리시스템을 설계 및 구현하였다. 센서정보 관리시스템은 센서정보를 저장, 검색, 수정 및 삭제할 수 있는 기능을 제공한다.

Figure 3 은 제안된 센서 정보시스템의 전체구조를 보인 것이다. 제안 시스템에서 1 개 이상의 센서로 구성된 가상센서(Virtual Sensor)는 센서 및 센서 메타데이터를 센서등록기(Sensor

Register)에 등록한다. 센서데이터는 데이터 검사기(Data Inspector) 에서 등록정보를 기반으로 센서의 제공데이터의 검사한다. 유효값은 설정된 데이터 윈도우에 따라 센서데이터의 평균, 최댓값 및 최소값을 암호화하여 다이제스트와 함께 전달한다. 센서데이터는 가상환경(Virtual Environment) 에서 수집기(Data collector)에 의해 수집된후 다이제스트를 기반으로 데이터 검증을 수행한다. 검증되어진 데이터를 기반으로 정의된 규칙에 따른 제어를 수행한다.

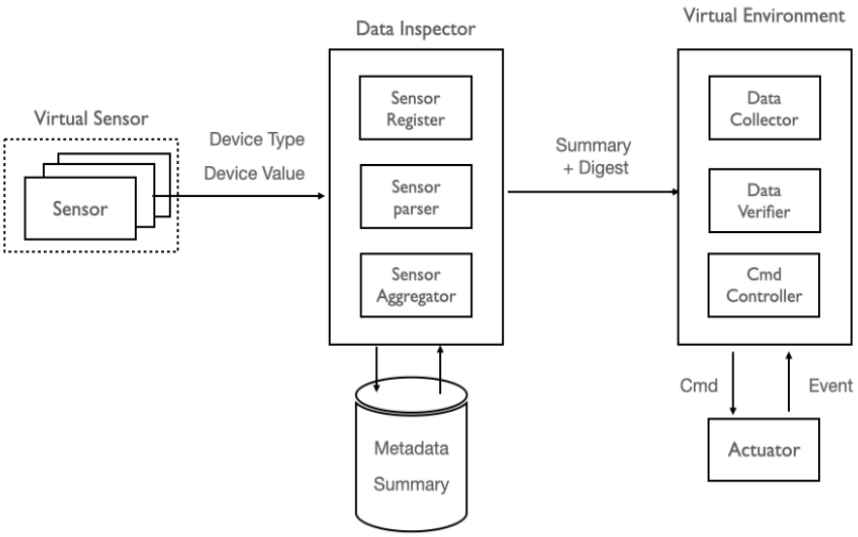

Figure 3 Overall structure of the proposed system

3.2 실험환경 및 실험결과

센서 등록을 위한 하드웨어 및 소프트웨어 환경은 다음과 같다. Celery 의 운영을 위한 메시지 브로커는 RabbitMQ 을 사용하며, 경량의 센서등록 환경 구성을 위해 Raspberry PI 를 사용한다. 구축된 하드웨어와 소프트웨어의 실험환경은 Table 1 과 같다

Table 1. Experimental HW and SW environment

Device	Raspberry Pi 3B
OS	Raspbian GNU/Linux 10
Python version	Python 3.7.3
MongoDB	2.4.14
Celery version	4.4.1 (cliffs)
RabbitMQ version	3.7.8

시스템은 기능별로 작업을 설계하고 구현하였다. 작업은 먼저 메시지 브로커를 통해 Celery Worker 에 전달된다. 워커에 전달된 작업을 통해 센서 정보를 저장하거나 검색 및 수정할 수 있다. 다음은 센서정보의 등록, 검색 및 수정의 처리결과를 중심으로 내용을 기술한다.

3.3.1 센서 등록 가능

클라이언트가 센서정보를 JSON 형식으로 작성해 Celery 에게 등록을 요청하면 Celery 는 등록되어있는 기능 중 센서정보 등록기능을 수행한다. 등록할 센서 정보는 JSON 형식으로 파일에 저장한다

Celery 는 등록 되어있는 기능 중 센서정보 등록을 수행하기 위해 데이터베이스와 연결한다. 클라이언트로부터 받은 데이터를 센서정보 컬렉션과 키 컬렉션에 저장한다. 키 컬렉션은 모든 센서 정보에서 사용중인 키들을 관리하며 센서정보 등록, 삭제 시 함께 등록, 삭제 된다. 센서

정보 컬렉션에 센서 정보를 등록하면 자동으로 고유한 ID 가 발급되는데 이 값은 센서정보에 대한 식별자로 활용되어 센서보를 검색할 때 사용될 수 있다. Figure 4 는 MongoDB shell 을 이용해 클라이언트가 요청한 센서 메타데이터의 결과를 보인 것이다. 센서데이터는 JSON 형식으로 표현되어 제공한다.

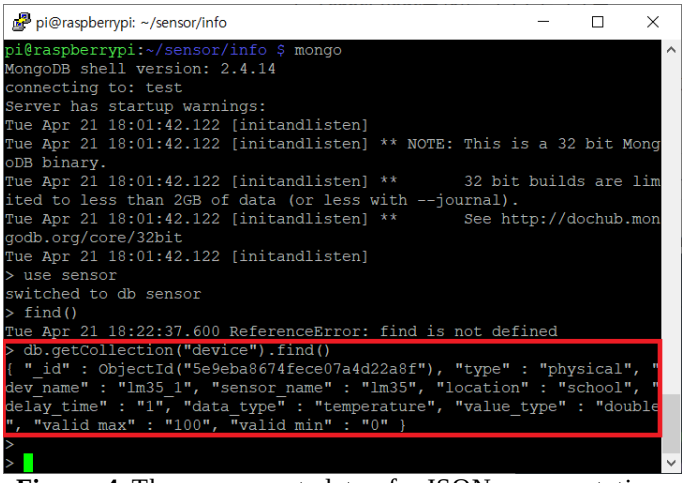

Figure 4. The sensor metadata of a JSON representation

클라이언트가 센서정보를 JSON object 형식으로 작성해 Celery 에게 등록을 요청하면 Celery 는 데이터베이스에 연결해 키 컬렉션과 센서정보 컬렉션에 저장한다. 키 컬렉션은 모든 센서정보에서 사용중인 키들을 관리하며 센서정보 컬렉션은 센서에 대한 정보를 관리한다. 센서정보 컬렉션에 센서정보를 등록하면 자동으로 고유한 ID 가 발급되는데 이 값은 센서정보에 대한 식별자로 활용되어 센서정보를 검색할 때 사용될 수 있다.

3.3.2 검색기능

검색기능에서는 키 컬렉션에 등록된 키에 대한 검색 기능을 제공하고 또한 센서정보 컬렉션에서 센서정보를 검색할 수 있는 기능을 제공한다. 센서정보를 검색하기 위해서는 센서 식별자를 이용하거나 키와 값을 제공해야 한다. 검색에 사용할 키가 유효한지 여부는 키 검색을 통해 알 수 있다. 키 컬렉션에서 키를 검색하기 등록된 키를 찾아 제공한다. Figure 5 은 센서정보 검색 결과를 보인 것이다.

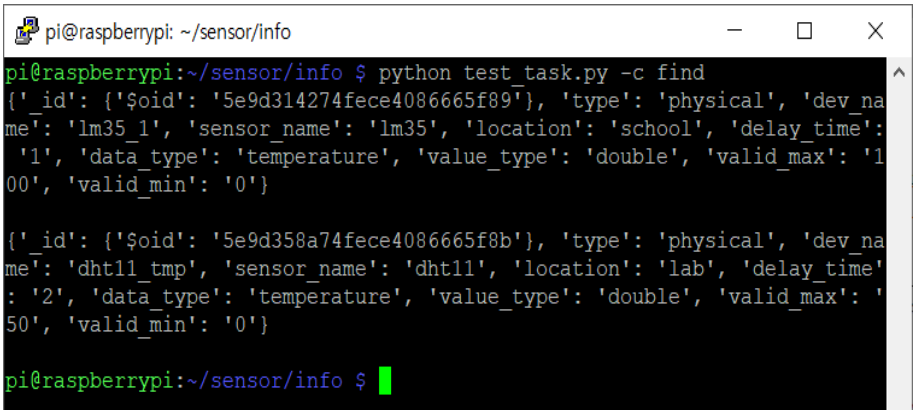

Figure 5. The result of browsing sensor metadata

3.3.3 수정기능

센서정보 컬렉션에 등록되어 있는 센서정보를 수정하기 위해서는 4 개의 매개변수, 찾을 센서정보에 대한 키와 값, 수정할 키와 새로운 값이 필요하다. 먼저 제공되는 2 개 매개변수는 수정하기 원하는 센서정보를 찾기 위해 사용된다. 이들 매개변수를 이용하여 특정 센서정보를 찾으면 나머지 2 개의 매개변수를 이용하여 센서정보를 수정한다. 수정을 위한 작업인 update 를 이용하여 수행한다. update 에는 4 개의 매개변수인 센서정보를 찾기 위한 키, 값 그리고 수정할 키, 값을 ‘,’ 로 구분하여 제공해야 한다. 수정 후에 수정된 결과를 전달한다. Figure 6 은 센서 데이터 수정 결과를 보인 것으로 센서 lm35_1 의 위치정보는 lab 으로 수정된다.

Figure 6. The result of updating sensory data

IV. 결론

센서데이터의 활용에 대한 필요성이 증가됨에 따라 센서정보에 대한 관리 및 이를 기반으로 한 센서데이터 수집은 센서 데이터의 활용도를 높이기 위한 주요한 작업이다. 이에 본 논문에서는 클라우드 기반의 센서정보 관리 시스템을 설계하고 구현하였다. 다양하고 많은 센서에 대한 정보들이 등록될 것을 고려하여 분산처리 프로세스인 Celery 와 반정형 자료의 유지를 위해 MongoDB 를 이용하였다. 구현된 시스템의 장점은 센서 정보를 등록하기 원하는 응용 프로그램은 어떤 것이든 비동기 통신으로 등록 가능하기 때문에 효율적으로 등록 가능하다는 것이다. 또한, MongoDB 를 활용함으로 센서의 특성에 맞는 형식으로 정보를 구성하여 등록을 할 수 있다. 시스템에서는 센서 정보에 대한 등록, 수정, 검색, 삭제 등 다양한 기능을 제공함으로 응용 프로그램에서 편리하게 사용 가능하다. 향후 센서 데이터 관리 시스템과 연동하여 이벤트나 소프트 센서와 같은 개념을 도입함으로 안정적인 IoT 환경을 제공할 수 있다.

V. 감사의 글

이 논문은 2021 학년도 백석대학교 대학연구비에 의하여 수행된 것임

VI. 참고문헌

[1] Ali, H., Soe, J. K., and Weller, S. R.. “A real-time ambient air quality monitoring wireless sensor network for schools in smart cities,” in Proceedings of the 2015 IEEE First International Smart Cities Conference (ISC2)

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito, “The Internet of Things: A survey,” Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 54, no. 15 pp. 2787-2805, 2010.
[3] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp 1645-1660, 2013.
[4] L. D. Xu, W. He, and S. Li, “Internet of Things in Industries: A Survey,” IEEE Trans. Industrial Informatics, vol. 10, no. 4, pp. 2233–2243, 2014.
[5] Reeves G, Liu J, Nath S, Zhao F. Managing massive time series streams with multi-scale compressed trickles. Proceedings of the VLDB Endowment. 2009;2(1):97-108
[6] Mohamed Ali Feki, Fahim Kawsar, Mathieu Boussard, and Lieven Trappeniers, "The Internet of Things: The Next Technological Revolution," Computer, vol. 46, no. 2, pp. 24 - 25, February 2013.
[7] http://docs.celeryproject.org/en/latest/getting-started/introduction.html
[8] https://medium.com/sunhyoups-story/Celery-b96eb337b9cf
[9] Kristina Chodorow and Michael Dirolf. 2010. MongoDB: The Definitive Guide (1st. ed.). O'Reilly Media, In
[10] Nicola Iarocci. Developing RESTful Web APIs with Python, Flask and MongoDB. https://speakerdeck.com/nicola/developing-restful-web-apiswith-python-flask-and-mongodb

저자 소개

강윤희 *(Yunhee Kang)*

1993 년 8 월 동국대학교 대학원 컴퓨터공학과 석사
2002 년 8 월 고려대학교 대학원 컴퓨터과학과 박사
2000 년 3 월 ~ 현재 백석대학교 컴퓨터공학부 부교수

관심분야 : 분산시스템, 인공지능, 클라우드컴퓨팅