KIPS Tr. Software and Data Eng.
Vol.5, No.7 pp.313~318 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2016.5.7.313

Handling Streaming Data by Using Open Source Framework Storm in IoT Environment

Yunhee Kang†

ABSTRACT

To utilize sensory data, it is necessary to design architecture for processing and handling data generated from sensors in an IoT environment. Especially in the IoT environment, a thing connects to the Internet and efficiently enables to communicate a device with diverse sensors. But Hadoop and Twister based on MapReduce are good at handling data in a batch processing. It has a limitation for processing stream data from a sensor in a motion. Traditional streaming data processing has been mainly applied a MoM based message queuing system. It has maintainability and scalability problems because a programmer should consider details related with complex messaging flow. In this paper architecture is designed to handle sensory data aggregated The designed software architecture is used to operate an application on the open source framework Storm. The application is conceptually used to transform streaming data which aggregated via sensor gateway by pipe-filter style.

Keywords : IoT, Sensory Data, Open Source Framework, Storm, Pipe-and-Filter

오픈소스 프레임워크 Storm을 활용한 IoT 환경 스트리밍 데이터 처리

강 윤 희†

요 약

IoT 환경에서 센싱 정보의 활용을 위해서는 센서로 부터 생성된 정보의 가공 및 처리를 위한 효율적인 소프트웨어 아키텍처 설계가 필수적이다. 특히 IoT 환경에서 사물은 인터넷에 연결되고 각종 센서를 탑재한 디바이스간 통신이 가능하여야 한다. 그러나 MapReduce 기반의 Hadoop과 Twister은 데이터의 배치 처리에 적합하지만, 스트리밍 센서 자료의 이동중 처리에는 제약점을 갖는다. 전통적인 스트리밍 데이터 처리 방법인 MOM 기반의 메시지 큐 시스템을 이용해 메시지 스트림을 처리하는 방식은 프로그래머가 메시지 흐름의 복잡도를 고려해야 상세한 처리를 프로그램 함으로써 유지보수 및 확장성을 갖기 어려움이 있다. 이 논문에서는 IoT 환경에서 수집된 센싱 자료의 처리를 위한 소프트웨어 아키텍처를 설계하였다. 또한 설계된 소프트아키텍처를 기반으로 오픈소스 프레임워크인 Storm의 응용 구성을 기술한다. 구성응용은 센서 게이트웨이(Sensor Gateway)를 통해 자료를 수집한 후 실시간 스트리밍 데이터를 파이프-필터 스타일로서 변환한다.

키워드 : 사물인터넷, 센서 데이터, 오픈소스 프레임워크, 스톰, 파이프-필터

1. 서 론

다양한 센서의 보급에 따른 IoT(Internet of Things) 구축, 모바일 스마트기기의 활성화, 트위터와 페이스북과 같은 소셜미디어의 사용자 증가로 인터넷 상의 데이터 양은 기하급수적으로 증가하고 있다. IoT 환경에서 센서정보의 활용을 위해서는 센서로부터 생성된 정보의 가공 및 처리를 위한 효율적인 아키텍처 설계가 필수적이다. 특히 IoT는 사물이 인터넷에 연결되고 각종 센서를 탑재한 디바이스간 통신이 가능하여야 하며 생성된 정보를 기반으로 데이터 분석 및 가시화와 같은 새로운 서비스 또는 가치 창출이 가능하여야 한다[1, 2].

과학연구 분야에서도 시뮬레이션 중심의 연구방법은 계측 및 센싱 장비의 도입으로 페타바이트(petabyte) 수준으로 데이터 폭발이 진행되고 있으며, 이 과정에서 데이터 중심(data intensive)으로 진화하고 있다[3, 4]. 연구방법의 진화에 따라 과학문제 해결은 이론기반의 실험에서 실데이터 분석 연구로 전환되고 있다. 일예로 극지연구 과제인 PolarGrid

※ 이 논문은 2016년도 백석대학교 대학연구비에 의해 수행되었음.
† 정 회 원 : 백석대학교 정보통신학부 부교수
Manuscript Received : April 22, 2016
Accepted : May 25, 2016
* Corresponding Author : Yunhee Kang(yhkang@bu.ac.kr)

에서는 센서와 측정장치를 활용한 IoT 환경이 구축되고 있으며, 실시간 및 대용량 데이터 처리에 대한 요구사항이 증가되고 있다. 전통적으로 대용량의 과학데이터처리는 병렬처리를 위한 MPI와 PVM과 같은 라이브러리를 활용하는 클러스터인 고성능 컴퓨팅(HPC, High Performance Computing)에 대한 기술이 적용되어 활용되고 있다. 그러나 HPC 환경에서의 프로그래밍 환경은 MPI와 PVM과 같은 낮은 수준의 통신 및 동기 처리 프리미티브를 사용하여야 하는 제약점을 갖고 있다[5].

Google은 이러한 제약점을 해결하기 위해 MapReduce 프로그래밍 모델을 도입하였으며, 이를 기반으로 대용량 데이터의 저장과 처리를 위한 GFS(Google File System)을 개발하였다[6]. MapReduce는 응용 프로그램에서 낮은 수준의 세부사항을 프레임워크에서 처리하도록 추상화함으로서 대용량의 데이터셋을 병렬 및 분산처리를 용이하게 지원한다[7].

ASF(Apache Software Foundation)의 MapReduce 기반 오픈 소스 소프트웨어인 Hadoop은 Yahoo, Facebook, Amazon 등의 인터넷 기반 기업에서 자료 처리를 위한 응용 개발 환경으로 사용하고 있다[8]. Hadoop은 MapReduce 응용처리에 필요한 대용량 데이터를 저장하기 위해 분산 파일 시스템인 HDFS(Hadoop Distributed File System)를 지원한다. HDFS는 수천 대 규모의 저가 서버 클러스터를 묶어 단일 파일 시스템 이미지를 제공하는 높은 확장성과 데이터 안정성을 보장한다. Hadoop은 반복적인 작업을 통해 자료를 처리해야 하는 무감독 또는 감독기반 기계학습(machine learning) 경우 초기 작업준비 비용에 따른 성능 제약점을 갖는다. Twisters는 이러한 성능제약점을 해결하기 위한 MapReduce 프레임워크로서 응용 수행 시 태스크 생성을 포함한 초기화, 정적인 데이터의 재로딩 및 Map과 Reduce 태스크 간의 통신 및 데이터 전송 오버헤드를 해결함으로써 반복적인 MapReduce 처리를 효율적으로 지원한다[9-11].

Hadoop과 Twister은 배치 데이터 처리에 적합하지만, IoT 환경에서 센서의 스트리밍 데이터 처리에는 제약점을 갖는다. 전통적인 스트리밍 데이터 처리 방법에는 MOM(Message-Oriented Middleware) 기반의 JMS(Java Message Service), NaradaBrokering 등 메시지 큐 시스템을 이용해 메시지 스트림을 처리한다. 그러나 이는 프로그래머가 메시지 큐에서 자료흐름의 세부 동작을 작성함으로서 프로그램의 유지관리에 어려움이 있다[12, 13].

Storm은 분산처리, 신뢰적이고 결함포용을 지원하는 데이터 스트림 처리 프레임워크로써 데이터 스트림 처리를 위해 기능을 스파웃(spout)과 볼트(bolt) 컴포넌트로 위임한다[14]. Storm 클러스터의 입력 스트림은 스파웃 컴포넌트에 의해 처리된다. 스파웃은 파일, 소켓 및 시리얼 장치로부터 데이터 스트림의 입력을 읽는다. 스파웃은 입력받은 데이터를 볼트 컴포넌트에 전달한다. 볼트는 스파웃 또는 다른 볼트로부터 전달받은 데이터를 변환한다. 이 과정에서 볼트는 중간 처리 결과를 저장하거나 다른 볼트에 전달한다. 즉, Storm은 볼트 컴포넌트의 체인을 구성하며, 각 볼트는 스파웃에서 제공하는 데이터 스트림을 다양한 형태로 변환한다.

이 논문에서는 IoT 환경에서 수집된 센싱 자료의 처리를 위한 소프트웨어 아키텍처를 설계하였다. 또한 설계된 소프트아키텍처를 기반으로 Storm 응용 구성을 기술한다. 구성 응용은 센서 게이트웨이(sensor gateway)를 통해 자료를 수집한 후 실시간 스트리밍 데이터 처리를 위해 파이프-필터 기반으로 자료를 변환한다. 이를 위해 센서 게이트웨이는 다양한 데이터 타입을 지원하는 메시지 큐로 BSD 라이센스 기반의 Redis(REmote DIctionary System)을 활용한다. 이를 위해 필터링과 변환 처리를 전체 프로세스로 정의된 토폴로지로 구성하여 처리한다.

이 논문의 구성은 다음과 같다. 2장에서는 시스템 구성을 위한 관련연구 내용인 Storm과 Redis을 기술하고 3장에서는 IoT 환경의 데이터 스트리밍의 요구사항을 정의한 후 아키텍처를 설계한 후 Storm을 활용하여 센서 게이트웨이와 센싱정보 처리 응용을 구성한다. 마지막으로 4장에서는 결론과 향후 연구방향에 대해서 기술한다.

2. 관련 연구

2.1 Storm

스트리밍 데이터 처리를 위한 분산 프레임워크인 Storm은 트위터의 메시지 분석을 위해 BackType에서 시작한 프로젝트로서 2014년에 Apache의 최상위 오픈소스 프로젝트로 성장하였다[14]. Storm 아키텍처는 Hadoop의 JobTracker 역할을 하는 Nimbus 데몬과 Hadoop의 TaskTracker 역할을 하는 Supervisor 데몬이 있다. Nimbus는 수행할 코드를 배포하고 Supervisor 노드에 작업을 할당하며 장애 조치(fail-over) 등을 담당한다. Supervisor 노드는 할당된 토폴로지(Topology)의 일부를 처리할 작업 프로세스(worker process)의 구동을 담당한다. Nimbus와 Supervisor는 Zookeeper를 이용해서 장애 상황에 대응한다. Fig. 1은 Storm 클러스터의 논리적 구성도를 보인 것이다.

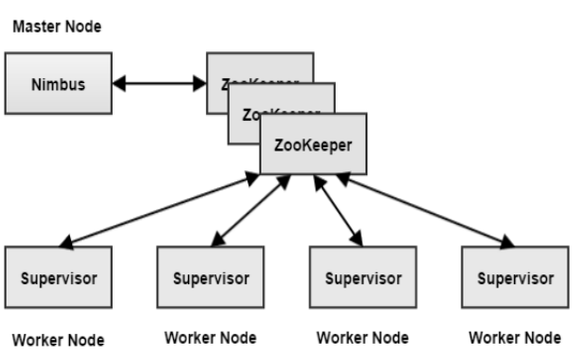

Fig. 1. Logical structure of Storm cluster

Hadoop의 MapReduce 작업과 토폴로지의 차이는, MapReduce 작업은 정해진 데이터 세트를 처리한 후 완료되지만 토폴로지는 계속해서 메시지를 처리한다는 점이다.

이러한 Storm 응용 토폴로지의 데이터 처리는 Hadoop에서 배치형식으로 대용량데이터를 처리하는 과정의 제약점을 해결하고 IoT 환경과 같이 스트리밍 형태로 전달되는 데이터의 실시간분석을 가능하게 한다.

전통적인 실시간 데이터 처리는 복잡한 메시지 큐를 이용해 메시지 스트림을 처리한다. 이는 개발자가 메시지 흐름의 복잡도를 고려해야 하는 어려움이 있다. 이러한 메시지 흐름의 복잡도를 해결하기 위해 Storm에서는 일련의 튜플(tuple)의 흐름을 스트림으로 정의하고 있으며 스트림을 분산 환경에서 신뢰성 있게 다른 스트림으로 전환할 수 있는 기능을 제공한다.

Storm은 DAG(Direct Acyclic Graph)로 표현되는 토폴로지로서 데이터의 소스, 처리와 출력모듈을 정의한다. 토폴로지를 사용한 데이터 스트림 구성은 통해 개발자의 복잡한 메시지 처리의 어려움을 해결한다. 이를 위해 Storm은 데이터 소스인 입력스트림을 스파우트로 처리와 출력모듈을 볼트로 정의한다. 스파우트는 스트림의 소스를 지칭하는 컴포넌트이다. 메시지는 튜플(Tuple)로 표현한다. 튜플은 기본 데이터 타입(primitive data)이나 바이트 배열(byte array)을 포함할 수 있고 사용자 정의 타입을 정의할 수도 있다. 스파웃은 스트림의 소스를 지칭하는 컴포넌트이다. 일반적으로 스파웃은 외부 소스로부터 튜플을 읽고 토폴로지로 튜플을 생성해 스트림을 생성하는 역할을 한다. 이 과정에서 볼트는 기존의 스트림을 데이터 처리에 필요한 다른 유형의 데이터 스트림으로 변환한다. Fig. 2는 2개의 스파우트와 4개의 볼트로 구성된 토폴로지를 보인 것이다.

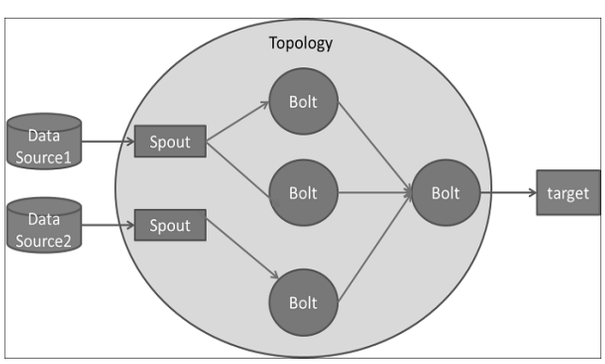

Fig. 2. The topology composed of spouts and bolts

2.2 Redis

Redis는 BSD 라이센스 기반의 오픈 소스 소프트웨어로서 C 언어로 작성되었으며, 인메모리(in-memroy) 기반의 키(Key)/값(Value) 저장소이다[15]. Redis는 검색을 위해 NoSQL을 지원함으로서 키에 대한 접근은 빠르다는 장점이 있으며, 이러한 이유로 단순한 질의 처리에 적합하다. 컬럼 중심 데이터베이스(Column Oriented Database)로서 작동하며 범위 검색에 대해서는 취약점을 갖는다. 클라이언트 SDK를 통해 Redis는 Java 언어 바인딩을 지원하는 Jedis를 사용한다. Redis는 C, Java, Python 등 다양한 프로그래밍 언어 바인딩을 지원한다. 다음은 Redis의 특징을 보인 것이다.

- Key-value 데이터 모델
- 인메모리(in-memory) 데이터 저장소
- NoSQL 기반의 데이터 처리

IoT 환경에서의 데이터 처리 패러다임은 정형적이고 종속성이 높은 데이터에서 반정형의 스트리밍 데이터 처리 성능이 중요해지고 있다. NoSQL 기반 저장시스템인 Redis은 IoT 환경과 같이 저장할 데이터의 유형이 다양하고 데이터 처리 시 특정 컬럼에 대한 통계기반 연산에 적합하다.

3. 시스템 설계 및 구현

3.1 요구사항 정의

IoT 환경에 설치된 센서는 센싱된 데이터를 스트리밍 방식으로 응용에 전송하고 응용에서는 데이터 스트림을 다른 스트림과 통합된 후 실시간 분석과 같은 다양한 서비스 및 응용에 활용하도록 제공한다. 그러나 스트리밍 데이터 시스템은 전통적인 데이터 처리와 다른 적시성(timeliness), 결함포용성(fault-tolerance), 확장성(scalability) 등의 비기능적 요구사항(non-functional requirement)을 만족해야 한다. 이 시스템의 운영을 위해서는 사물에서 생성되는 데이터를 적기에 수집, 분석 및 예측하여 보다 정교하고 유용한 정보를 사용자에게 제공하여야 한다. 다음 세 가지 항목은 이를 위한 주요 기능요구 사항을 기술한 것이다.

- IoT 디바이스 프로파일 관리
- IoT 사용자 프로파일 관리
- IoT 서비스 및 응용 프로파일 관리

이 논문에서는 기능요구사항 중 센서 데이터 처리 및 분석 서비스 및 응용 기능을 제공하는 IoT 서비스 및 응용 프로파일 관리 항목의 기능 항목을 중심으로 시스템 설계를 고려한다. 이를 위해 IoT 서비스 및 응용 프로파일 구현은 이형의 실시간 대용량 스트리밍 센서 데이터를 수집하여 처리를 통해 데이터에 대한 변환이 필요하다.

3.2 아키텍처 설계

스트리밍 기반의 데이터 처리 환경에서 메시징 시스템은 응용에 필요한 데이터 전달을 효율적으로 지원하기 위해 사용한다. 응용 수행 과정에서 송신자는 송신 메시지가 수신자에 의해 수신되거나 처리되기를 대기하지 않는 비동기 방식으로 동작한다. 메시지는 독립적인 정보단위로서 각 메시지는 메시지를 처리하는 응용의 대상 업무가 필요로 하는 자료와 상태를 표현한다. 설계된 아키텍처는 IoT 환경에서 수집된 센서 데이터를 응용에 전달하기 위해 파이프-필터(pipe-and-filter) 패턴을 이용한다. 파이프-필터 패턴에서 각 컴포넌트는 입력과 출력 집합을 갖는다[17].

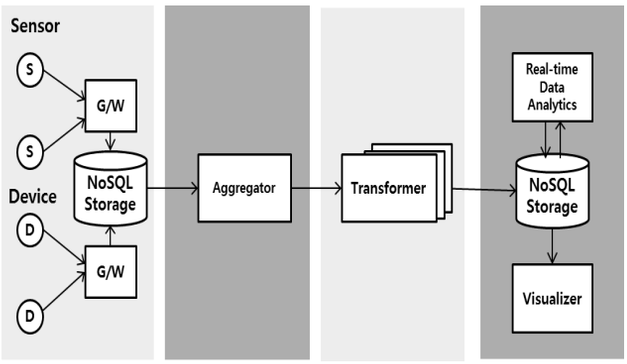

Fig. 3. Storm Platform based system architecture

Fig. 3은 설계된 시스템 아키텍처를 보인 것으로 센서 게이트웨이(G/W, gateway)는 센서 및 장치(device)로 부터의 원시 센싱 데이터를 획득한 후 메시지로 구성한다. 메시지는 특정시점에서 수집된 센싱 데이터를 표현한다. 센서 게이트웨이는 자체 저장장치에 수집 데이터를 저장하지 않고 NoSQL 저장소를 통해 수집기(aggregator)에 전달한다. 이 때 수집기는 일련의 변환기(transformer)를 통해 데이터의 특정자료를 추출 및 변환하는 필터링 기능을 수행한 후 최종처리 결과는 실시간 데이터 응용(real-time data analytics) 및 가시화기(visualizer)에서 사용할 수 있도록 NoSQL 저장소에 저장한다.

3.3 실험 환경 및 결과

이 실험에서는 Redis의 Storm 연계를 위해 오픈소스 자바 라이브러리인 Jedis를 사용한다. Redis는 Pub/Sub 방식 메시지 전달을 통해 솔기없이(seamless) 외부 IoT 환경과 Storm 응용과 연계하도록 지원한다. 센서 게이트웨이는 출판자(publisher)로서 센싱자료를 Storm응용 플랫폼인 클러스터에 동작하는 스파웃에 전달한다. 이 과정에서 스파웃은 수신자(subscriber)의 역할을 수행한다. 아키텍처 검증을 위한 IoT 응용은 다음의 운영체제와 관련 소프트웨어를 사용하여 응용을 개발하였으며 실험환경을 구축하였다.

- JDK 1.8.x
- Redis 4.2.2 및 Jedis 24.2
- Storm 0.9.5
- Ubuntu 12.0.4 (Kernel 2.6.x) Linux

이 실험의 데이터는 인텔 버클리 연구실 내부에 설치된 54개의 모트 센서(Mote sensor)로부터 2004년 2월 28부터 2004년 4월 5일까지 수집된 230만 레코드로 구성된 공개 데이터셋을 재플레이(replay)하여 활용한다. 각 레코드는 날짜(date), 시간(time), 메시지 순서(epoch), 센서 식별자(id), 온도(temperature), 습도(humidity), 조도(light), 및 센서전원(voltage) 의 8개의 자료값으로 구성된다. 실시간 데이터 분석을 위해 Storm 수집 데이터를 Redis에 저장한 후 스파웃을 통해 이를 제공한다. 이후 스파웃은 자료에 대한 파싱을 수행하여 자료유형 별로 볼트에 전달한다. Storm 응용의 형상인 파이프-필터 토플로지는 센싱 데이터 변환 태스크를 수집, 변환, 저장의 3개 단계로 정의하고 각 태스크는 Storm 컴포넌트로서 수행한다. 수집기는 JSON 형식의 입력 데이터를 센서 게이트웨이로부터 메시지로 전달받는다. 다음은 JSON 형식의 입력메시지 구성 예를 보인 것이다.

```
{"date":"2004-02-28","time":"00:59:16.02785
","epoch":"3","id":"1","temperature":"19.98
84","humidity":"37.0933","light":"45.08","v
oltage":"2.69964"}
```

Fig. 4는 Storm 응용 토폴로지인 Indoor의 UI 결과화면을 보인 것으로 응용 컴포넌트는 indoor-pubsub-spout과 temp, temp-avg, report 및 redis-sub의 볼트로 구성된다. 파란색은 스파웃을 표현한 것으로 스파웃 indoor-pubsub-spout은 데이터 스트림을 해석한 후 볼트 temp에 전달한다.

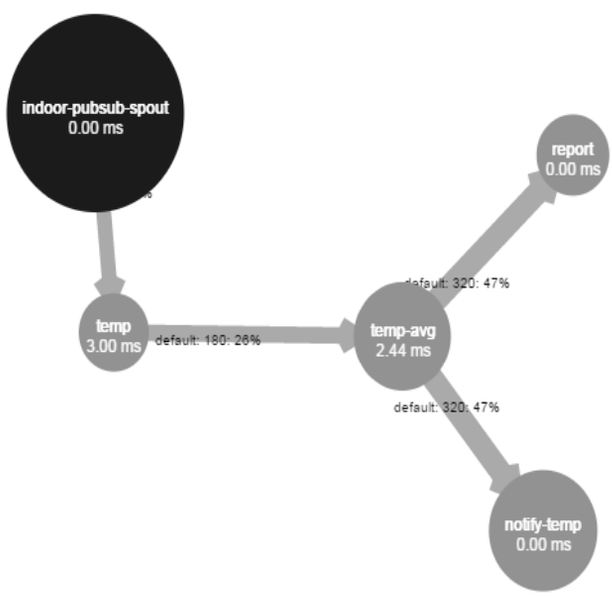

Fig. 4. UI of topology Indoor

Storm 컴포넌트는 볼트 체인을 통해 기본 데이터 셋인 튜플을 생성하고, 센서데이터에 대한 데이터 레코드를 구성한 후 변환기를 이를 전송한다. 스파웃 indoor-pubsub-spout은 단일 JSON 메시지를 수신한 후 키 json_message와 수신 JSON 메시지 값을 개별 <키, 값>의 쌍으로 구성하여, 변환기 역할을 수행하는 볼트에 전달한다. 다음은 수신된 JSON 형식 메시지를 <키,값> 쌍으로 구성한 예를 보인 것이다.

변환기(transformer)의 역할을 수행하는 temp와 temp-avg 볼트는 입력된 튜플(<키, 값>)을 주어진 센서값을 추출하여

요약된 온도 정보를 생성하여 변환을 수행한다. 볼트 temp는 센서의 값의 범위를 기반으로 비정상적인 자료값을 제거한 후 이를 볼트 temp-avg에 전달한다. temp-avg는 전달받은 온도 데이터를 기반으로 센서별 평균값을 구하고 이를 notify-temp와 report 볼트에 전달한다. 변환 결과는 저장 수행 결과 생성된 하나 이상의 데이터 레코드는 튜플 형식으로 저장기에 전달된 자료에서 전달된 값 중 주요한 데이터 값에 대한 정렬 후 상위자료를 notify-temp 볼트를 사용하여 Redis 저장소에 저장한다. notify-temp는 계산된 온도 데이터를 통지하는 출판자 역할을 하며, report 볼트는 센서별 평균값을 파일로 저장하여 유지하도록 한다.

키	값
json_msg	{"date":"2004-02-28","time":"00:59:16.02785","epoch":"3", "id":"1","temperature":"19.9884","humidity":"37.0933", "light":"45.08","voltage":"2.69964"}

4. 결 론

IoT는 센서를 갖는 사물이 사물 및 사용자와 상호작용이 솔기없이 연결된 분산 컴퓨팅 환경으로 IoT 시스템은 서비스 및 응용간의 자료전달을 위한 서비스 인터페이스 기술은 IoT 구성 요소들을 서비스 및 응용과 연동하도록 하는 역할을 수행한다.

IoT 환경 센서로 부터의 스트리밍 데이터를 체계적으로 수집하고 분석을 통한 시스템 개발이 요구되고 있으나 Hadoop은 실시간 데이터 처리가 용이하지 않으며 이를 해결하기 위해서는 스트림 기반의 센서 데이터 처리를 위한 확장이 필요하다. 이 논문은 센서 데이터 소스로 부터 체계적인 수집 및 저장 기능을 위한 아키텍처를 설계하였으며 오픈소스 Storm기반으로 응용 처리를 위한 인메모리 저장소인 Redis을 활용하여 시스템을 설계 및 구현하였다. 이를 위해 실내환경의 수집된 센서 데이터 셋을 활용하여 응용을 구성한 후 실험을 수행하였다.

향후 NodeMCU, Arduino 및 RasberryPi와 같은 오픈소스 하드웨어 플랫폼을 사용하여 센싱정보를 수집하여 설계된 아키텍처에서의 응용 성능평가를 수행할 예정이다.

References

[1] 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.

[2] Yunhee Kang, “New Approach to the Platform for the Application Development on the Internet of Things Environment,” *Journal of Platform Technology*, Vol.3, No.1, pp.21-27, 2015.

[3] Z.(G.) Guo, R. Singh, and M. Pierce, “Building the PolarGrid Portal Using Web 2.0 and OpenSocial,” *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC'09)*, Porland, OR, ACM Press, p.5, 2009.

[4] Yunhee Kang, “Enabling BigData Platform for MapReduce Applications in the Science Cloud,” *Journal of Platform Technology*, Vol.2, No.1, pp.42-52, 2014.

[5] W. Gropp and E. Lusk, “Goals guiding design: PVM and MPI,” *Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on*, pp.257-265, 2002, doi: 10.1109/CLUSTR.2002.1137754.

[6] J. Dean and S. Ghemawat, “Mapreduce: Simplified Data Processing on Large Clusters,” *Communications of the Acm*, Vol.51, pp.107-113, Jan., 2008.

[7] J. Dean and S. Ghemawat, “MapReduce: A Flexible Data Processing Tool,” *Communications of the Acm*, Vol.53, pp. 72-77, Jan., 2010.

[8] Hadoop [Internet], http://hadoop.apache.org/.

[9] J. Ekanayake et al., “Twister: A Runtime for Iterative MapReduce,” The First International Workshop on MapReduce and its Applications (MAPREDUCE'10) - HPDC2010, 2010.

[10] Yunhee Kang and Y. B. Park, “The performance evaluation of k-means by two MapReduce frameworks, Hadoop vs. Twister,” *2015 International Conference on Information Networking (ICOIN)*, Cambodia, pp.405-406, 2015.

[11] Yunhee Kang and Geoffrey C. Fox, “Performance Evaluation of MapReduce Applications on Cloud Computing Environment, FutureGrid,” FGIT-GDC, pp.77-86, 2011.

[12] Curry Edward, “Message-Oriented Middleware,” in *Middleware for Communications*, ed. Qusay H Mahmoud, 1-28. Chichester, England: John Wiley and Sons, 2004.

[13] S. Pallickara and G. Fox, “NaradaBrokering: a distributed middleware framework and architecture for enabling durable peer-to-peer grids,” *the Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, Rio de Janeiro, Brazil, 2003.

[14] Apache Storm [Internet], http://storm.incubator.apache.org.

[15] J. Carlson, “*Redis in Action*,” 1st ed., New York: Manning, 2013.

[16] J. Pokorny, “NoSQL databases: a step to database scalability in web environment,” in *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, 2011, pp.278-283.

[17] M. Shaw and D. Garlan, “*Software Architecture: Perspectives on an Emerging Discipline*,” Prentice-Hall, 1996.

강 윤 희

e-mail : yhkang@bu.ac.kr

1989년 동국대학교 컴퓨터공학과(공학사)

1991년 동국대학교 컴퓨터공학과(공학석사)

2002년 고려대학교 컴퓨터과학과(이학박사)

2000년~현 재 백석대학교 정보통신학부 부교수

관심분야 : 클라우드컴퓨팅, 그리드 컴퓨팅, 결함포용