대규모 IoT 데이터 수집 및 제어를 위한 MQTT 및 Elastic 스택 기반 시스템 설계 제안

신승호, 이상범
SK 브로드밴드

sinius@sk.com, sb.lee@sk.com

A Proposal of System Architecture based on MQTT and Elastic Stack for Massive IoT Data Collection and Control.

Shin Seungho, Lee Sangbeom
SK Broadband

요 약

본 논문은 수백만 IoT 기기에서 발생하는 대량의 정보를 수집 및 분석하고, 기기들을 제어해야 하는 상황에서 적용할 수 있는 MQTT 브로커와 Elasitc 스택 기반의 시스템 구조를 제안한다. 제안구조는 클러스터 방식으로 시스템 확장(Scale-out)에 유연하게 설계되었으며 벤치마크 성능결과 대규모 IoT 기기를 위한 서비스에 해당 설계의 적용이 가능함을 보였다.

Ⅰ. 서 론

MQTT(Message Queuing Telemetry Transport) 프로토콜은 발행/구독 기반의 양방향 통신이 가능한 경량 프로토콜로 수백만 개의 IoT(Internet Of Things) 기기와 연결이 가능하도록 설계되었다[1]. MQTT 는 서버 역할을 하는 브로커를 통해 IoT 기기들과 메시지를 주고받는데 MQTT 브로커는 상용제품과 오픈소스로 다양한 종류가 공개되어 있다[2]. 브로커마다 지원하는 기능과 성능이 상이하며 이에 따라 시스템 구조와 투입되는 인프라 자원이 달라질 수 있다. 한편, MQTT 브로커는 메시지를 주고받는 중개 역할만 할뿐 메시지를 따로 보관하진 않는다. 그러므로 IoT 기기로부터 수집한 메시지를 별도로 보관하려면 메시지를 저장할 수 있는 데이터베이스 사용이 필요하며 이를 위해 MQTT 브로커와 데이터베이스를 연결하는 작업이 필요하다. 본 논문에서는 대규모 IoT 기기 접속환경에서 필요한 MQTT 브로커 기능을 살펴보고, Elastic 스택[3]을 활용하여 전송된 메시지를 저장하고, 분석할 수 있는 시스템을 설계하고자 한다.

Ⅱ. 본 론

다양한 MQTT 브로커 중에 대규모 IoT 환경에서 적합한 제품을 고르기 위해서는 각 브로커가 지원하는 기능의 비교가 필요하다[4]. 첫번째로 MQTT 프로토콜 버전을 모두 지원하는지를 살펴봐야한다. 현재 공식적으로 스펙이 공개된 MQTT 프로토콜 버전은 MQTT 5.0, 3.1.1, 3.1, MQTT-SN v1.2 가 있다[5]. 두번째로 SSL/TLS 와 WSS 같은 보안관련 기능을 제공하는지를 확인하여 송수신하는 메시지의 보안이 필요한 경우 암호화가 가능한지 확인해야한다. 세번째로는 메시지 전송레벨인 QoS 0, 1, 2 모두 지원하여 서비스 특성에 따라 메시지 관리가 가능해야 한다. 마지막으로 서비스 중 클라이언트 수의 증가에 따라 서비스 중단없이 브로커를 확장할 수 있도록 클러스터 기능을 제공하는지 확인이 필요하다. 여러 브로커들 중에서 위에서 언급한 대부분의 조건을 만족시키는 제품 중에 2013 년에 개발되어 오픈소스로 공개된 EMQ X 브로커[6]가 있으며 본 논문에서는 해당 브로커를 선택하여 설계를 진행하였다.

브로커를 비교하여 설계하고자 하는 목적에 맞는 브로커를 선택했다면 성능에 대한 검증이 필요하다. EMQ X 의 경우 8-core, 32G Memory Cent OS 단일 노드 서버에서 130 만의 커넥션이 가능했다고 나와있다[7]. 본 논문에서는 12-core, 16G Memory Cent OS 스펙의 서버 4 대에 각각 EMQ X 브로커를 설치하여 클러스터로 구성 후 MQTT 벤치마크툴인 emqtt_bench[8]를 사용하여 벤치마크를 진행하였다.

첫번째로 커넥션 테스트는 클라이언트 역할을 하는 18 대의 서버를 이용하여 초당 1,000 건씩 최대 50 만의 접속을 테스트해보았다.

테스트 결과 50 만 커넥션이 커넥션 성공 후 1 시간 이상 동안 1 건의 커넥션 끊김없이 커넥션이 유지되었으며 CPU 와 Memory 가 안정적으로 유지되었다.

표 1. CPU, Memory 정보(50 만 커넥션)

System Time	Conn.	CPU Info (1load/5load/15load)	Memory Info (used/total)
18:45:58	500,000	0.50/0.31/0.31	3.69G/4.72G
19:49:57	500,000	0.00/0.02/0.05	3.35G/4.62G

두번째로 토픽 구독 테스트는 첫번째와 동일한 조건으로 50 만 커넥션을 연결하는 동시에 특정 토픽을 구독하고, 해당 토픽에 메시지를 발행하는 것을 테스트해보았다.

테스트 결과 50 만 커넥션 및 구독이 동시에 정상적으로 이뤄졌으며 CPU 와 Memory 사용율도 안정적으로 유지되었다. 그리고 50 만 클라이언트가 구독하는 토픽에

메시지 1 건을 발행했을 때 해당 메시지가 약 10 초 이내 약 50,000TPS 수준으로 50 만 클라이언트 모두에 메시지가 정상적으로 전달되는 것을 확인할 수 있었다.

표 2. CPU, Memory 정보(50 만 커넥션 및 구독)

Conn.	Sub.	CPU Info (1load/5load/15load)	Memory Info (used/total)
500,000	500,000	0.06/0.29/0.37	4.09G/6.93G

세번째로 토픽 발행 테스트는 1,000 개의 클라이언트가 동시접속 후 각 클라이언트가 초당 100 건의 메시지를 발행하였을 때를 테스트하였다.

약 1 분 동안 서버로 전송한 약 2,900,000 개의 메시지가 유실없이 발행되었고, 약 48,000TPS 수준으로 메시지가 전송되었으며 전송 중 CPU, Memory 사용율은 안정적으로 유지되었다.

표 3. CPU, Memory 정보(290 만 토픽 메시지 발행)

Conn.	Sub.	CPU Info (1load/5load/15load)	Memory Info (used/total)
1,000	1,000	0.50/0.46/0.73	274.97M/507.13M

토픽 구독/발행 테스트를 통해 대규모 IoT 기기가 연결된 상태에서 Admin Console 과 같은 관리자 Application 을 통해 MQTT 브로커로 직접 IoT 기기를 제어할 수 있는 메시지 전달이 유실없이 가능함을 확인할 수 있었다.

MQTT 브로커의 성능이 확인되었다면 클라이언트로부터 발행된 메시지를 브로커를 거쳐 데이터베이스에 저장하는 부분이 필요하다. 해당 부분은 Elastic 스택을 사용하며 데이터베이스 역할은 Elasticsearch 가 맡게 된다. 그리고 MQTT 브로커에서 메시지를 수집하여 Elasticsearch 에 전달해줄 연결체는 Filebeat 로 구성할 수 있다. 2020 년 5 월 14 일에 릴리즈된 Filebeat 7.7.0 버전부터 MQTT Input 모듈이 추가되었다[9]. 해당 모듈을 사용하여 브로커에 연결하여 토픽을 구독할 수 있으며 발행된 메시지를 Elasticsearch Output 모듈을 사용하여 Elasticsearch 에 저장할 수 있다. 한편 성능을 위해 Filebeat 프로세스를 복수개로 띄울수 있는데 클러스터로 구성된 MQTT 브로커에 여러개의 Filebeat 로 동일한 토픽을 구독하는 경우 클라이언트로부터 발행된 메시지는 모든 노드에 동일하게 전달되므로 중복된 메시지가 저장되는 문제가 발생할 수 있다. 이런 문제를 회피하기 위해서 MQTT 브로커는 Shared Subscription 기능을 제공하며 아래와 같은 토픽 주소로 해당 기능을 사용할 수 있다[10].

```
$share/{Group Name}/{Topic Name}
```

그림 1. Shared Subscription Example

클라이언트로부터 수집된 데이터를 Elasticsearch 에 저장하였다면 이를 분석하기 위한 도구가 필요한데 Elastic 스택에서 Kibana 라는 시각화 분석툴을 제공하고 있다[11].

앞서 살펴본 내용들로 MQTT 브로커와 Elastic 스택을 활용하여 최종으로 제안하는 시스템 설계는 그림 2 와 같다.

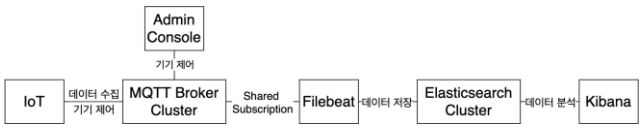

그림 2. MQTT 및 Elastic 스택 기반 시스템 설계 제안

수백만 클라이언트가 존재하는 환경에서 해당 설계를 바탕으로 구성된 시스템을 실제 서비스에 사용하고 있으며 해당 시스템에는 Elasticsearch 의 유지보수 작업 시 데이터가 유실되는 것을 방지하기 위하여 Filebeat 와 Elasticsearch 사이에 Kafka 를 두는 그림 3 과 같은 시스템 구성으로 운영하고 있다.

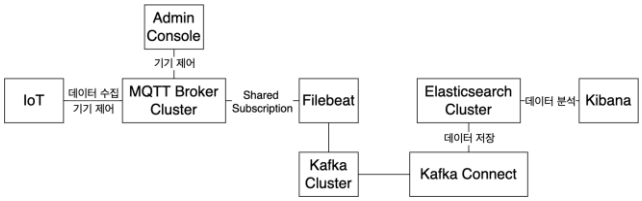

그림 3. MQTT 및 Elastic 스택, Kafka 기반 시스템 설계 제안

Ⅲ. 결 론

본논문에서는 MQTT 브로커, Elastic 스택을 활용하여 대규모 IoT 기기의 정보를 수집 및 분석하고, 제어할 수 있는 시스템을 설계하였다. 해당 설계는 클러스터 구조로 확장이 가능하고, 수백만 IoT 기기의 연결이 가능함을 벤치마크 테스트로 확인하였다.

참 고 문 헌

[1] MQTT https://mqtt.org/

[2] MQTT Servers/Brokers
https://github.com/mqtt/mqtt.org/wiki/servers

[3] Elastic https://www.elastic.co/kr/what-is/elk-stack

[4] Comparison of MQTT implementations
https://en.wikipedia.org/wiki/Comparison_of_MQTT_implementations

[5] MQTT Specifications https://mqtt.org/mqtt-specification/

[6] EMQ https://www.emqx.io/products/broker

[7] EMQ X Tuning guide
https://docs.emqx.io/en/broker/v4.3/tutorial/tune.html

[8] Erlang MQTT Benchmark Tool
https://github.com/emqx/emqtt-bench

[9] Filebeat MQTT Input
https://www.elastic.co/guide/en/beats/filebeat/7.x/filebeat-input-mqtt.html

[10] EMQ X Shared subscription
https://docs.emqx.io/en/broker/v4.2/advanced/shared-subscriptions.html

[11] Kibana https://www.elastic.co/kr/kibana