https://doi.org/10.7840/kics.2024.49.1.79

분산 메시징 시스템에서 Topic과 Partition 관리를 통한 메시지 전송 최적화

남 범 준◆, 권 영 우°

Optimizing Message Transfers in Distributed Messaging Systems through Topic and Partition Management

Beomjun Nam◆, Young-Woo Kwon°

요 약

스트림 데이터의 처리 기술이 중요시되면서 대용량, 대규모의 데이터를 빠르고 손실 없이 전송하기 위해 Apache Kafka, RabbitMQ 그리고 ActiveMQ 등 많은 메시징 시스템이 사용되고 있다. Apache Kafka는 대표적인 분산 메시징 시스템이며, 실시간으로 생성되는 데이터를 빠르게 전달하는데, 장점이 있는 시스템이다. Apache Kafka의 구성요소 중 Topic은 Broker의 하위에 구성되며 Partition 개수가 다른 여러 개의 Topic으로 구성될 수 있다. Topic을 구성하는 Partition의 개수가 증가하게 되면 높은 처리 속도를 보여주지만 무분별하게 Partition의 개수를 증가시킴에 따른 문제점도 발생한다. 본 논문에서는 Topic을 구성할 때, 사용자의 목표 처리량에 맞는 Partition 개수로 Topic을 구성해야 하는 이유와 목표 처리량에 맞는 Partition 개수를 제안할 수 있는 실험에 대해 소개하고 Partition 개수를 최소한으로 사용하면서 목표 처리량에 맞는 Partition 개수를 제안하는 방법을 소개한다.

키워드 : 분산 메시징, 아파치 카프카, 파티션, 스트림 처리
Key Words : Distributed messaging, Apache Kafka, Topic, Partition, Streaming processing

ABSTRACT

As stream data processing technology becomes more important, messaging systems such as Apache Kafka, RabbitMQ, and ActiveMQ are being used to transfer large amounts of data fast and without loss. Apache Kafka is a representative distributed messaging system, which can deliver data generated in real time. In Apache Kafka, a broker is composed of multiple topics with different numbers of partitions. As the number of partitions increases, its processing speed also increases, but problems with CPU and memory usages also occur. In this article, we show why the number of partitions should be configured to reduce resource usages without impact on target performance. Based on our extensive experimental results, we propose a mechanism that can change the number of partitions according to the amount of transferred message under different execution environments.

※ 이 논문은 2021년도 교육부의 재원(No.2021R1I1A3043889)으로 한국연구재단의 지원과 2023년도 정부(과학기술정보통신부) 및 지자체(대구광역시)의 재원으로 (재)대구디지털혁신진흥원에서 주관하는 지역 디지털 혁신거점 조선지원 사업의 지원을 받아 수행된 연구임(NO.DBS1D-03).
◆ First Author : Kyungpook National University, ghi3740@knu.ac.kr, 학생회원
° Corresponding Author : Kyungpook National University, ywkwon@knu.ac.kr, 정회원
논문번호 : 202309-081-B-RU, Received September 13, 2023; Revised September 29, 2023; Accepted September 29, 2023

Ⅰ. 서 론

최근 실시간으로 처리해야 할 데이터 양이 증가하면서 데이터 스트림 처리 기술이 중요시되고 있으며 대표적인 스트림 처리 기술은 Apache Storm, Apache Spark 등이 있다. 스트림 처리 기술은 데이터를 빠르게 전송하는 것보다 손실 없이 전송하는 것이 중요하므로 데이터를 손실 없이 전송하기 위해서는 메시지 큐를 이용한 시스템이 주로 사용되고 있다. 메시지 큐 시스템은 메시지를 임시로 저장하고 Consumer가 구독한 Topic에 대한 데이터만 직접 가져가는 비동기식 처리 방식을 통해 동작하며, 대표적인 메시지 시스템은 Apache Kafka, RabbitMQ 그리고 ActiveMQ 등이 있다[1].

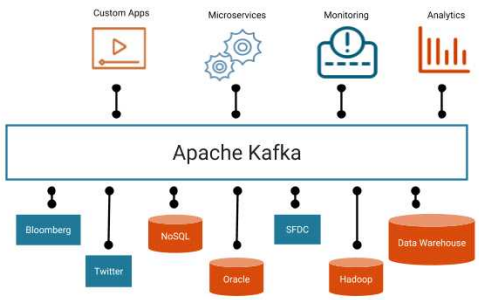


그림 1. Apache Kafka 사용
Fig 1. Using Apache Kafka.

Apache Kafka는 대용량, 대규모의 데이터를 빠르고 손실 없이 전송 가능한 분산 메시지 플랫폼이며, 링크드인(LinkedIn)에서 개발해 2011년 초 오픈소스로 공개되었다[2]. 실시간으로 발생하는 대용량 로그 처리에 특화되어 있고 메시지 큐를 사용하기 때문에 데이터를 유실 없이 전송할 수 있는 메시징 큐 시스템이다[3,4].

크게 Cluster, Broker, Topic 그리고 Partition으로 나누어지며 Broker는 한 개 이상의 Topic으로 구성된다. 하나의 Topic은 최소한 한 개 이상의 Partition으로 구성되며 Topic을 구성하는 Partition의 개수에 따라 전송 속도가 변화하며 Partition의 개수가 증가하면 높은 처리 속도를 보여준다[5]. 하지만 증가시킨 Partition의 개수는 감소시킬 수 없으며, Partition의 개수가 증가함에 따라 문제점도 발생하기 때문에 초기 Topic 생성 시에 사용자의 목표 처리량에 맞는 Partition 개수로 Topic을 구성하는 것이 중요하다.

본 논문에서는 Apache Kafka를 통해 데이터를 전송할 때, 사용자의 목표 처리량에 맞는 Partition 개수를 지정해야 하는 이유와 목표 처리량에 따른 Partition 개수를 수식화할 수 있는 실험을 진행하였고 Partition의 개수를 최소한으로 사용하면서 사용자의 목표 처리량을 충족할 수 있는 Partition 개수를 제안한다.

Ⅱ. 본 론

2.1 관련 연구

Apache Kafka에 관한 많은 연구가 이루어지고 있지만, 대부분의 연구에서는 Apache Kafka의 성능에 영향을 주는 요인에 대한 연구[6]이거나, Apache Kafka를 통해 데이터 전송을 하는 경우 최적화된 전송을 위한 Broker, Replication-factor 개수 변화에 따른 성능 변화 연구에 집중되고 있다[7-9].

Apache Kafka를 통해 데이터 전송을 할 때, Broker에서 여러 개의 Topic을 생성하고 Topic에 데이터를 저장하는 과정이 필요하다. 본 논문에서는 Broker에서 Topic을 생성할 때, 변화를 줄 수 있는 요소 중 하나인 Partition의 개수 변화에 따른 전송 속도, 지연 시간 그리고 메모리 사용량 변화에 대해 실험을 진행하였다. Partition의 개수가 증가하면 전송 속도는 빨라지지만, 메모리 사용량은 증가하게 된다. 따라서 최소한의 Partition의 개수를 사용하여 목표 처리량에 맞는 Partition 개수로 Topic을 구성해야 최적화된 전송이 가능하다. 또한, 증가시킨 Partition의 개수는 감소시킬 수 없으므로 초기 설정 시에 목표 처리량에 맞는 Partition 개수를 설정해야 한다. 하지만 Partition의 개수 설정에 대한 가이드가 없으므로 사용자가 Topic을 생성할 때 Partition의 개수를 변화시키면서 사용자의 환경에 맞는 개수를 찾아야 한다. 또한, 사용 가능한 메모리의 양이 충분하지 않은 환경에서 Topic을 구성하는 Partition의 개수가 많으면 전송 오류가 발생한다. 이 경우에 Topic에서 Partition 개수를 줄이는 것은 불가능함으로 Topic을 삭제한 후 다시 만들어야 한다는 번거로움이 있다. 이 때문에 사용자의 목표 처리량에 맞는 Partition의 개수를 제안하기 위해 Partition의 개수가 변화함에 따라 전송 속도, 지연 시간 그리고 메모리 사용량을 비교하였다.

2.2 Apache Kafka 전송과정

전송하는 방향(Producer)에서 메시지라고 정의된 데이터 단위를 저장소라고 불리는 Topic에 데이터를 저장하면, 수신받는 방향(Consumer)에서 구독한

Topic에 저장된 데이터를 직접 가져가는 방식으로 동작한다. Producer는 메시지를 Consumer에게 직접 전달하는 방식이 아닌 중간의 메시지 큐 시스템을 통해 간접 전달하는 방식으로 동작한다.

2.3 Apache Kafka 구조

먼저 Apache Kafka의 구조와 구성 요소에 대해 설명한다.

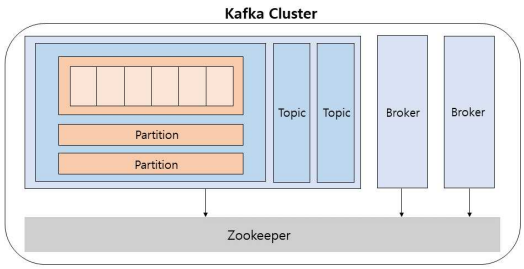


그림 2. Apache Kafka의 구조
Fig 2. Structure of Apache Kafka.

- Cluster : 여러 개의 Broker로 구성된 집합이며, Broker가 메시지를 나누어 저장한다.
- Broker : Topic 내부의 Partition을 관리하며 클라이언트와 데이터 송수신을 위해 사용하는 주체이며 데이터를 분산 저장하여 장애가 발생하더라도 안전하게 사용할 수 있도록 도와주는 역할을 한다.
- Topic : Apache Kafka를 통해 전송된 메시지를 구분하고, Partition의 그룹으로 구성된다.
- Producer : Apache Kafka의 Topic으로 메시지를 전송하는 역할을 하며, 데이터를 전송할 때, 리더 Partition을 가지고 있는 Broker와 직접 통신한다.
- Consumer : Partition에 저장되어있는 메시지를 소비하는 역할을 한다. Partition에서 데이터를 가져와서 Offset의 위치를 기록한다.
- Zookeeper : Apache Kafka 프로젝트 중 하나로 분산 애플리케이션 관리를 위해 안정적인 연결을 담당한다. 또한, Kafka의 메타데이터 관리 및 Broker 상태 점검이 가능하다.

2.4 Apache Kafka 특징

기존의 메시징 시스템은 Producer가 Consumer에게 메시지를 Push 해주는 방식인데 Apache Kafka는 Consumer가 Producer로부터 메시지를 직접 가져가는 Pull 방식으로 동작한다. Consumer는 자신의 처리 능력만큼의 메시지만 가져오기 때문에 최적의 성능을 낼 수 있는 특징이 있다. 또한, 기존의 메시징 시스템에서는 전송된 메시지가 즉시 삭제되지만, Apache Kafka에서는 메시지에 설정된 수명이 지나야 삭제된다.

2.5 Apache Kafka 장점

메시지를 메모리에 저장하는 기존 시스템과는 달리 Apache Kafka는 메시지를 파일에 저장하기 때문에 서비스를 재시작하여도 메시지 유실 우려가 감소한다. 데이터를 병렬로 처리하기 때문에 빠르고 효과적으로 처리할 수 있으며, 데이터를 디스크에 순차적으로 저장하기 때문에 임의 접근 방식보다 데이터를 빠르게 처리할 수 있다. 또한, 클러스터링에 의한 가용성이 높아 Scale - out이 가능하며 시스템 확장이 쉽다.

2.6 Partition 수에 따른 성능 및 리소스 사용량

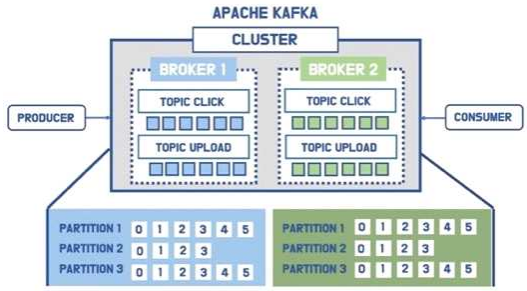


그림 3. Topic과 Partition의 구성
Fig 3. Configuration of Topics and Partitions.

하나의 Topic을 구성하는 Partition의 개수가 많으면 처리량이 많다는 장점이 있다. 하지만 이에 따른 단점도 존재한다. 첫 번째로, 파일 핸들러가 낭비된다. Partition마다 개별적인 저장소를 가지는데 Log Segment 당 2개의 파일(Index, Actual data)을 생성한다. Broker는 모든 Log Segment의 파일을 열기 때문에 Partition의 개수가 많아지면 파일 핸들러가 낭비된다. 두 번째로 장애 복구 시간이 증가한다. Apache Kafka는 높은 가용성을 위해 리플리케이션을 지원하는데 Broker는 여러 개의 Partition으로 구성된다. 각각의 Partition은 리플리케이션이 동작하며, 하나의 Partition은 리더 Partition이 되고 나머지 Partition은 팔로워 Partition이 된다. 만약 Broker에서 장애가 발생하면, Broker의 리더 Partition은 사용할 수 없게 되고 팔로워 Partition 중 하나를 리더 Partition으로 변경하여 클라이언트 요청을 처리하는 과정에서 시간이 오래 걸린다는 단점이 있다. 마지막으로 메모리가 낭

비된다. Partition은 개별적인 메시지 버퍼를 가지며 시간이 지남에 따라 Producer에서 전송된 메시지가 버퍼에 누적되고 Partition의 개수가 증가하게 되면 Producer가 메모리 양이 할당된 범위를 초과할 수 있다는 단점이 있다.

2.7 Topic과 Partition의 관계

Topic은 데이터 저장소를 의미하며 데이터를 저장하여 데이터를 구분하기 위한 단위이다. Partition은 Topic에서 분리된 공간을 의미한다. 하나의 Topic마다 한 개 이상의 Partition을 가지며, 메시지는 Round - Robin 방식으로 쓰여진다. Partition의 목적은 많은 데이터를 빠르게 송수신할 수 있도록 하는 것이다.

Ⅲ. 제안방법

본 논문에서 제안하는 방법의 개요는 그림 4와 같다.

그림 4. 제안하는 방식의 개요
Fig. 4. Overview of the proposed method.

사용자의 환경에서 Apache Kafka 모니터링 툴인 AKHQ, UI for Apache Kafka를 사용해 실시간으로 변화하는 전송 속도, 지연 시간 그리고 메모리 사용량을 입력 값으로 받는다. 하나의 Topic을 구성하는 Partition의 개수를 3개로 기준으로 하여 성능을 측정한 후 사용자가 목표로 하는 처리량에 따른 증감율을 구한다. Topic을 구성하는 Partition 개수를 3개로 기준하여 찾은 처리량과 처리량에 따른 증감률과 Partition 개수 변화에 따른 증감률 결과를 이용하여 사용자의 환경에서 목표 처리량에 맞는 Partition 개수를 찾을 수 있다. 사용자의 환경에서 Topic을 구성한 Partition의 개수가 3개일 경우에서 성능을 측정한 후 사용자가 더 높은 성능을 목표로 하면 Partition 개수를 증가시키며 성능을 모니터링한다.

Apache Kafka에서는 하나의 Topic을 구성하는 Partition의 개수를 15개 이하로 구성하는 것을 권장하며 본 논문에서 실험한 결과를 보면 Partition의 개수가 15개 이하의 경우에서는 Partition의 개수가 증가하면 성능이 향상하는 것을 알 수 있고 Partition의 개수

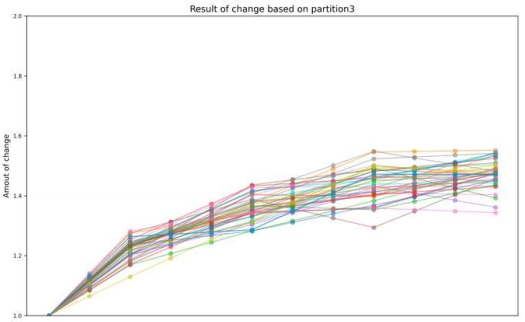


그림 5. Partition 변화에 따른 전송 속도 변화
Fig. 5. Transmission speed change according to partition change.

가 15개 이상이면 성능이 감소하는 것을 알 수 있다. 하지만 Topic을 구성하는 Partition의 개수가 12개 이상의 경우에서 전송 속도의 변화폭은 크지 않았지만, 메모리 사용량의 변화폭은 전송 속도 변화에 비교하여 많이 증가하는 것을 알 수 있었다. 따라서 본 논문에서는 메모리 사용량을 최소화하기 위해 Partition의 개수를 최소한으로 사용하면서 사용자의 목표 처리량을 충족할 수 있는 Partition 개수를 찾는 방법을 제안한다. 제안하는 방법은 Topic을 구성하는 Partition의 개수가 3개일 때의 성능을 측정한 후에 사용자가 목표로 하는 성능 증감률을 통해 Partition 개수를 동적으로 증가시키며 찾는 방법이다. Topic을 구성하는 Partition의 개수는 증가시키면 감소시킬 수 없고 Partition의 개수가 증가하게 되면 사용하는 메모리의 양이 증가하기 때문에 최소한의 Partition으로 Topic을 구성하고 증감률을 비교하여 목표 처리량에 맞는 Partition 개수를 찾는 방법을 제안한다.

Ⅳ. 실 험

4.1 Partition 개수 증가에 따른 성능 변화

가상 플랫폼 환경인 Docker에서 여러 컨테이너를 관리할 수 있는 Docker-Compose를 통해 컨테이너를 생성하였고 Apache Kafka와 Kafka가 정상 동작하는데 필요한 Zookeeper를 실행하였다. 실시간으로 변화하는 전송 속도, 지연 시간 그리고 메모리 사용량을 모니터링하기 위해 Apache Kafka 모니터링 도구 중 멀티 클러스터 관리와 카프카 연결 관리가 가능한 UI for Apache Kafka와 AKHQ(Apache Kafka HQ)를 사용하였다. 하나의 Broker에서 Partition의 개수가 다른 여러 개의 Topic을 생성하였고 Producer가 동일한 크기의 메시지를 동일한 횟수로 Consumer에서 전송

했을 때 전송 속도, 지연 시간 그리고 메모리 사용량을 고려하여 최소한의 메모리를 사용하면서 사용자의 목표 처리량을 달성할 수 있게 해주는 Partition의 개수를 제안하기 위한 실험을 진행하였다. 본 논문에서 Partition 개수를 지정하기 위해 Partition 개수를 20개 이하일 때 Partition의 개수를 한 개씩 증가시키면서 성능 변화량을 측정하였다. 실험 결과를 보면 본 논문의 세 가지 실험 환경에서 Partition의 개수를 15개 이하로 구성할 때 증가하는 성능을 보여준다는 결과를 알 수 있었다. 실험 결과를 바탕으로 Topic을 구성하는 Partition 개수는 3, 6, 9, 12 그리고 15개로 구성하였다. 서로 다른 세 개의 환경에서 동일한 크기의 메시지를 동일한 횟수로 전송할 때, Partition의 개수 변화에 따른 전송 속도, 지연 시간 그리고 메모리 사용량 변화를 측정하는 실험을 진행하였다.

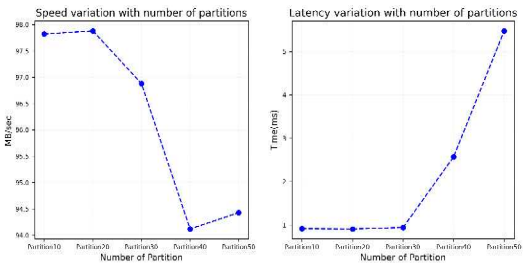


그림 6. Partition 개수 변화로 인한 영향(10~50)
Fig. 6. Effect of changing the number of partitions (10~50).

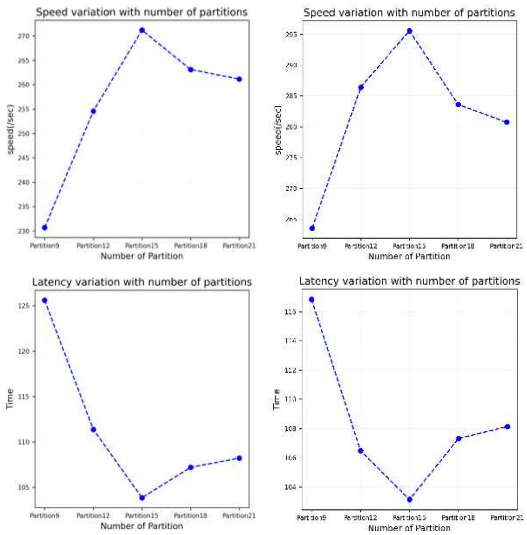


그림 7. Partition 개수 변화로 인한 영향(9~21)
Fig. 7. Effect of changing the number of partitions (9~21).

그림 6은 실험 환경에서 하나의 Topic을 구성하는 Partition 개수의 범위를 지정하기 위해 임의로 Partition의 개수를 10, 20, 30, 40 그리고 50개로 설정하고 전송 속도와 지연 시간을 비교한 결과이다. 그림 6의 왼쪽 그래프의 결과는 Partition의 개수 증가에 따른 전송 속도 변화를 보여주며 그림 6의 오른쪽 그래프의 결과는 Partition의 개수증가에 따른 지연 시간의 변화를 보여준다. 그림 6의 결과를 살펴보면 전송 속도의 측면에서 Partition 개수를 10, 20개로 구성하였을 때 가장 뛰어난 전송 속도를 보여주는 것을 알 수 있었고 Partition의 개수가 20개 이상이 되면 전송 속도가 감소하는 것을 알 수 있었다. 지연 시간 측면에서는 10, 20 그리고 30개로 구성하였을 때 가장 뛰어난 성능을 보여주는 것을 알 수 있었고 Partition의 개수가 30개 이상이 되면 지연 시간이 증가하는 것을 알 수 있었다. 그림 7의 위쪽 두 개의 결과는 Partition의 개수가 증가함에 따른 전송 속도 변화이고 아래 두 개의 결과는 Partition의 개수가 증가함에 따른 지연 시간 변화이다. Partition이 15개 이하일 때는 전송 속도와 지연 시간의 성능이 향상하는 것을 알 수 있지만 15개 이상일 때는 성능이 감소하는 것을 알 수 있었다. Partition이 증가함에 따른 전송 속도와 지연 시간을 비교해 본 결과 Partition의 개수가 15개 일 때 가장 우수한 성능을 보여주는 것을 알 수 있었다. 위 실험을 통해 본 논문에서 진행한 실험에서의 Partition 개수는 3, 6, 9, 12 그리고 15개로 구성하였다.

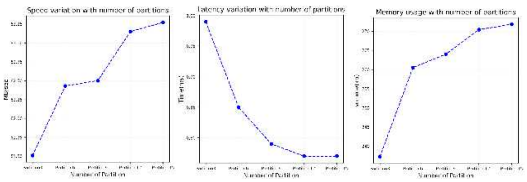


그림 8. 500Byte 데이터 500,000번 전송결과
Fig. 8. Result of transmitting 500Byte of data 500,000 times.

그림 8는 500Byte 데이터를 500,000번 전송한 전송 속도, 지연 시간 그리고 메모리 사용량의 변화를 보여준다. 좌측에서부터 전송 속도, 지연 시간 그리고 메모리 사용량의 결과를 보여준다.

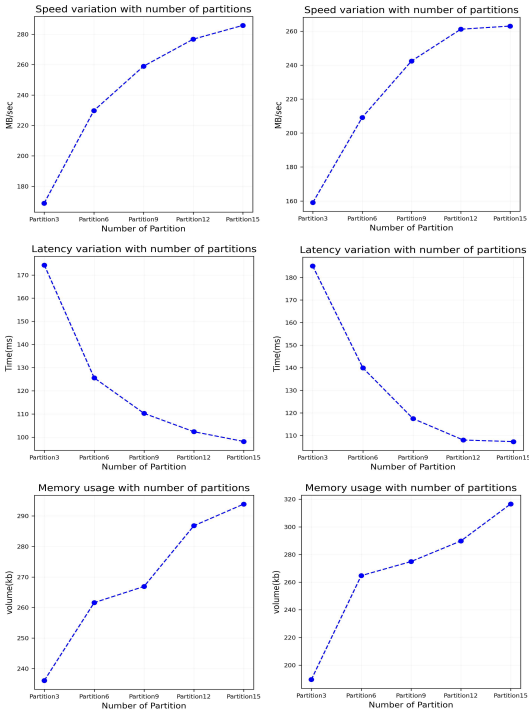
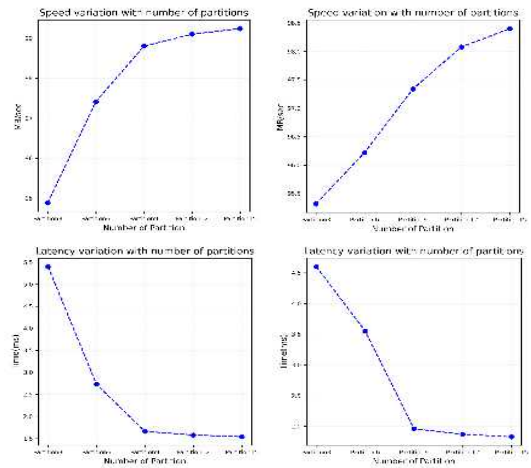


그림 9. 1,000Byte 데이터 1,000,000번 전송결과
Fig. 9. Result of transmitting 1,000Byte of data 1,000,000 times.

그림 9은 1,000 Byte 데이터를 1,000,000번 전송한 전송 속도, 지연 시간 그리고 메모리 사용량의 변화를 보여준다. 위에서부터 전송 속도, 지연 시간 그리고 메모리 사용량의 결과를 보여준다.



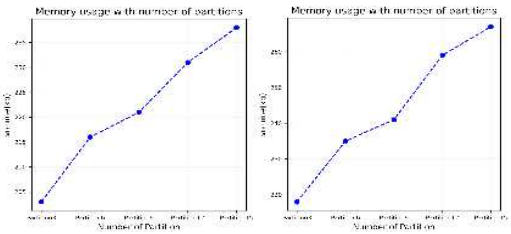


그림 10. 1,500Byte 데이터 1,500,000번 전송결과
Fig. 10. Result of transmitting 1,500Byte of data 1,500,000 times.

그림 10는 1,500Byte 데이터를 1,500,000번 전송한 전송 속도, 지연 시간 그리고 메모리 사용량의 변화를 보여준다. 위에서부터 전송 속도, 지연 시간 그리고 메모리 사용량의 결과를 보여준다.

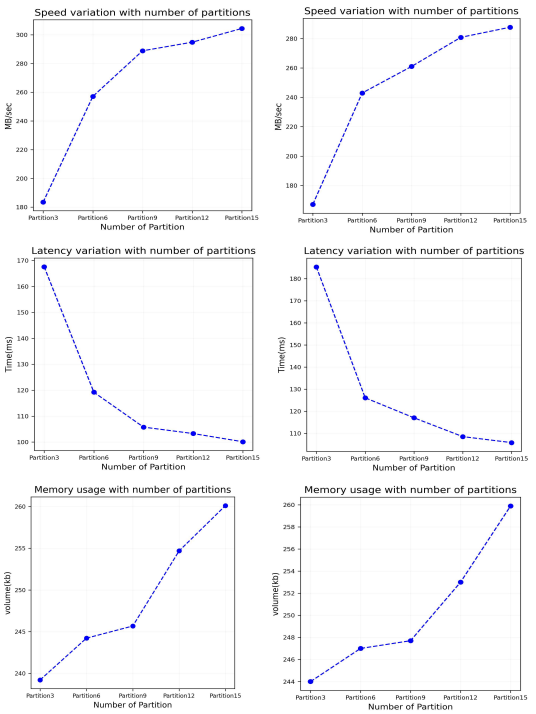


그림 11. 2,000Byte 데이터를 2,000,000번 전송한 결과
Fig. 11. Result of transmitting 2,000Byte of data 2,000,000 times.

그림 11은 2,000Byte 데이터를 2,000,000번 전송한 전송 속도, 지연 시간 그리고 메모리 사용량 변화를 보여준다. 위에서부터 전송 속도, 지연 시간 그리고 메모리 사용량의 결과를 보여준다.

서로 다른 세 개의 실험 환경에서 네 번의 실험 결과를 종합한 결과를 살펴보면 전송하는 데이터 크기와 전송 횟수와는 관련 없이 Topic을 구성하는 Partition 개수가 증가할수록 전송 속도는 빨라지고 지

연 시간은 감소하며 메모리 사용량은 증가한다는 것을 알 수 있었다.

전송 속도와 지연 시간의 측면에서 결과를 살펴보면 Topic을 구성하는 Partition의 개수를 3 ~ 9개 사이로 구성할 경우 전송 속도와 지연 시간의 변화가 크고 Partition의 개수를 9개 이상으로 구성하면 변화가 크지 않는다는 것을 알 수 있었다. 다음으로 메모리 사용량 측면에서 결과를 살펴보면 Topic을 구성하는 Partition의 개수를 9 ~ 15개 사이로 구성할 경우 변화가 크고 Partition의 개수를 9개 이하로 구성할 경우 변화가 작다는 사실을 알 수 있었다. Producer에서 Consumer로 메시지를 전송할 때 전송 횟수와 메시지 크기를 변경하면 Partition 개수에 따라 전송 속도, 지연 시간 그리고 메모리 사용량의 값은 변화하지만, 값의 변화하는 폭은 비슷하다는 것을 알 수 있다. 다음 표 2, 표 3 그리고 표 4는 서로 다른 세 개의 환경에서 네 번의 실험을 통해 Partition의 개수 3개를 기준으로 전송 속도, 지연 시간 그리고 메모리 사용량의 변화량을 측정한 결과이다.

표 1. Partition3을 기준으로 전송 속도 변화
Table 1. Change in transfer rate based on partition3.

Transfer Speed(MB/s)	P3	P6	P9	P12	P15
Experiment1	0	+80.76	+111	+135.3	+147.6
Experiment2	0	+71.45	+93.24	+117.6	+122.8
Experiment3	0	+58.61	+92.18	+124.27	+125.48
Experiment4	0	+47.08	+68.13	+86.76	+107.02

표 2. Partition3을 기준으로 지연 시간 변화
Table 2. Change in latency time based on partition3.

Latency Time(ms)	P3	P6	P9	P12	P15
Experiment1	0	-45.39	-61.31	-64.44	-69.55
Experiment2	0	-49.14	-64.45	-64.83	-65.13
Experiment3	0	-41.36	-53.81	-55.61	-63.51
Experiment4	0	-47.8	-61.47	-66.72	-71.05

표 3. Partition3을 기준으로 메모리 사용량 변화
Table 3. Change in memory usage based on partition3.

Memory Usage(MB)	P3	P6	P9	P12	P15
Experiment1	0	+26	+31	+51	+58b
Experiment2	0	+75	+85	+100	+127
Experiment3	0	+12	+22	+23	+34
Experiment4	0	+18	+19	+29	+42

표 4. Partition3을 기준으로 한 변화 증감율
Table 4. Change rate based on Partition3.

Change Rate	P3	P6	P9	P12	P15
Transfer Speed	100%	138%	154%	164%	169%
Latency Time	100%	72%	63.5%	60%	58%
Memory Usage	100%	120%	124%	129%	137%

표 1, 표 2 그리고 표 3은 서로 다른 세 개의 환경에서 네 번의 실험 결과를 바탕으로 변화량을 측정한 결과이다. Partition의 개수를 3개로 기준으로 하여 Partition이 증가함에 따라 전송 속도, 지연 시간 그리고 메모리 사용량 변화를 표로 정리한 결과이다. 전송 속도와 지연 시간은 Partition의 개수가 15개에 가까워질수록 변화량이 적어지는 것을 알 수 있고 메모리 사용량은 Partition이 증가함에 따라 계속 증가하는 것을 알 수 있다. 또한, 표 4는 Partition 3개를 기준으로 하여 Partition이 변화할 때 전송 속도, 지연 시간 그리고 메모리 사용량의 증감률을 구한 결과이다.

Partition의 개수가 증가함에 따라 전송 속도는 빨라지며 지연 시간은 감소하며 메모리 사용량은 증가한다는 사실을 알 수 있었다. 또한, Topic을 구성하는 Partition의 개수가 증가함에 따라 전송 속도가 가장 큰 변화 폭을 보여주며 지연 시간이 가장 작은 변화 폭을 보여주는 것도 알 수 있었다.

모든 실험 결과를 종합한 결과는 다음과 같다. Topic을 구성하는 partition의 수가 늘어날수록 전송 속도가 빨라지며 지연 시간은 감소한다. 하지만 15개 이상의 partition을 사용하는 경우에는 전송 속도는 감소하며 지연 시간도 증가하는 것을 알 수 있었다. 메모리 사용량은 Topic을 구성하는 Partition의 수가 늘어남에 관계없이 증가하는 것을 알 수 있었다. 사용자의 환경에서 최소한의 메모리를 사용하기 위해 최소한의 Partition 개수를 사용하여 Topic을 구성하면서

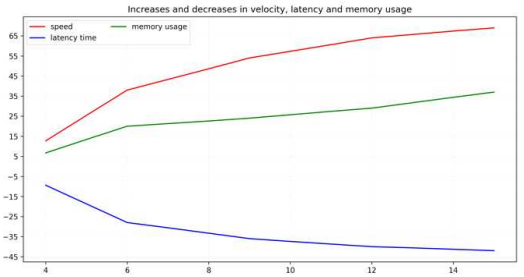


그림 12. 전송 속도, 지연 시간 그리고 메모리 사용량 변화량
Fig. 12. Changes in transfer speed, latency time and memory usage.

사용자가 목표로 하는 처리량을 만족하는 Partition의 개수로 Topic을 구성해야 한다.

4.2 제안하는 방법과의 비교

본 논문에서는 제안하는 방법을 통해 Partition 개수를 찾는 방법과 실제 사용자의 환경에서 목표 처리량에 맞는 Partition 개수를 찾는 방법을 비교하였다.

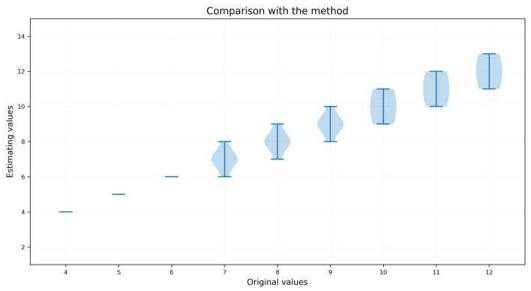


그림 13. 제안하는 방법과 실제 개수 비교
Fig. 13. Comparison of proposed method and actual number.

Topic을 생성할 때 임의로 Partition 개수를 지정하면 사용자의 목표 처리량에 도달하는지 알 수 없을 뿐만 아니라 사용자의 환경에서 사용 가능한 메모리의 양이 충분하지 않다면 전송 과정에서 오류가 발생한다. 그러므로 초기 Topic 생성 시에 전송이 가능한 최소한의 Partition으로 Topic을 구성한 후 사용자의 목표 처리량과 처리량에 대한 증감률을 통해 대략적인 Partition 개수를 지정한 후 모니터링을 통해 동적으로 Partition 개수를 찾는 방법을 제안함으로써 사용자의 목표 처리량에 가장 비슷한 Partition 개수를 찾을 수 있고 메모리 부족으로 인한 전송 오류를 최소화할 수 있다. 하지만 Topic을 생성할 때 성능에 영향을 주는 사항인 Broker, Replication-factor 그리고 Partition 개수 등이 있는데 Partition 개수 변화만으로는 성능 향상에 제한이 있다. 본 논문에서 제안하는 방법을 통해서 처리량 증감률에 따른 Topic을 구성하는 Partition 개수를 찾을 수 있지만, Partition 개수 변화를 통한 성능 향상 범위를 초과하게 되면 사용자가 원하는 정확한 Partition 개수를 찾을 수 없다는 단점이 있다.

V. 결 론

본 논문에서는 Apache Kafka를 사용하여 메시지를 전송할 때 Topic을 구성하는 Partition 개수 증가에 따른 장단점을 설명하였다. Partition의 개수 변화에 따른 전송 속도, 지연 시간 그리고 메모리 사용량을 비교하여 사용자의 목표 처리량에 따른 최적화된 Partition 개수를 제안하기 위해 서로 다른 세 개의 환경에서 네 번의 동일한 크기의 메시지를 동일한 횟수만큼 전송하는 실험을 진행하였다. 진행한 실험 결과를 바탕으로 사용자의 환경에서의 성능과 성능 증감률을 통해 성능 모니터링을 하여 동적으로 Partition 개수를 조정하는 방법을 제안한다.

추후 연구로는 사용자가 Apache Kafka를 사용하여 데이터를 전송하는 경우 Topic을 구성하는 Partition의 개수 설정을 통해 얻을 수 있는 성능 향상은 제한이 있다. 그러므로 사용자가 Topic을 생성할 때 변화를 줄 수 있는 Broker 개수와 Replication-factor 개수에 따른 성능 변화에 관한 연구를 이어 나가고자 한다.

References

[1] V. M. Ionescu, "The analysis of the performance of RabbitMQ and ActiveMQ," *2015 14th RoEduNet NER*, pp. 132-137, 2015. (https://doi.org/10.1109/roedunet.2015.7311982)

[2] Z. Wang, W. Dai, F. Wang, H. Deng, S. Wei, X. Zhang, and B. Liang, "Kafka and its using in high - throughput and reliable message distribution," *2015 8th Int. Conf. Intell. Netw. and Intell. Syst.*, pp. 117-120, 2015. (https://doi.org/10.1109/icinis.2015.53)

[3] S. Vyas, R. K. Tyagi, C. Jain, and S. Sahu, "Performance evaluation of apache kafka - A modern platform for real time data streaming," *2022 2nd ICIPTM*, pp. 465-470, 2022. (https://doi.org/10.1109/iciptm54933.2022.9754154)

[4] R. Shree, T. Choudhury, S. C. Gupta, and P. Kumar, "KAFKA: The modern platform for data management and analysis in big data domain," *2017 2nd Int. Conf. TEL-NET*, pp. 1-5, 2017. (https://doi.org/10.1109/tel-net.2017.8343593)

[5] P. Le Noac'H, A. Costan, and L. Bouge, "A performance evaluation of apache kafka in support of big data streaming applications," *2017 IEEE Int. Conf. BIGDATA*, pp. 4803-4806, 2017.

(https://doi.org/10.1109/bigdata.2017.8258548)

[6] D. Kim, S. Son, M.-S. Gil, and Y-S. Moon, “Apache kafka benchmark test on HDD and SSD environment,” in *Proc. KCC 2017*, pp. 1691-1693, 2017.
(http://www.riss.kr/link?id= A1 03298423)

[7] T. P. Raptis and A. Passarella, “On efficiently partitioning a topic in apache kafka,” *2022 Int. Conf. CITS*, pp. 3-8, 2022.
(https://doi.org/10.1109/cits55221.2022.9832981)

[8] D. Kim, M.-S. Gil, M. C. Nguyen, H. Won, and Y-S. Moon, “Apache kafka benchmark test for an efficient distributed messaging environment,” in *Proc. KCC 2021*, pp. 87-89, 2021.
(http://www.riss.kr/link? id=A107796826)

[9] Y. Hong, D. Kim, and Y.-S. Moon, “Apache kafka benchmark test over various network and disk environments,” in *Proc. KSC 2021*, pp. 1187-1189, 2021.
(http://www.riss.kr/link? id=A108045568)

남 범 준 (Beomjun Nam)

2022년 2월 : 경일대학교 컴퓨터공학과 졸업
2022년 3월 : 경북대학교 컴퓨터학부 석사과정
<관심분야> 분산 시스템, 사물인터넷
[ORCID:0009-0000-7938-6257]

권 영 우 (Young-Woo Kwon)

2003년 2월 : 경북대학교 컴퓨터과학과 졸업
2005년 2월 : 광주과학기술원 정보통신공학과 석사
2014년 7월 : Virginia Tech, 컴퓨터과학과 박사
2014년 8월~2017년 6월 : Utah State University, 교수
2017년 8월~현재 : 경북대학교, 컴퓨터학부 교수
<관심분야> 분산시스템, 빅데이터, 인공지능, IoT, 지진조기경보, 재난 ICT
[ORCID:0000-0003-0625-8232]