초격차 패키지 Online.

대규모 트래픽을 고려한 간단한 SNS 서비스

PART1 | 개요
강사 소개 및 프로젝트 소개

PART2 | 간단한 SNS 어플리케이션 만들기
회원가입 및 로그인, 포스트 CRUD API 개발하기

PART3 | SNS 어플리케이션 발전시키기
좋아요, 댓글, 알림기능 개발하기

PART4 | 대규모 트래픽 고려해보기
대규모 트래픽에 적합한 Architecture 설계 및 구현

대규모 트래픽 고려해보기

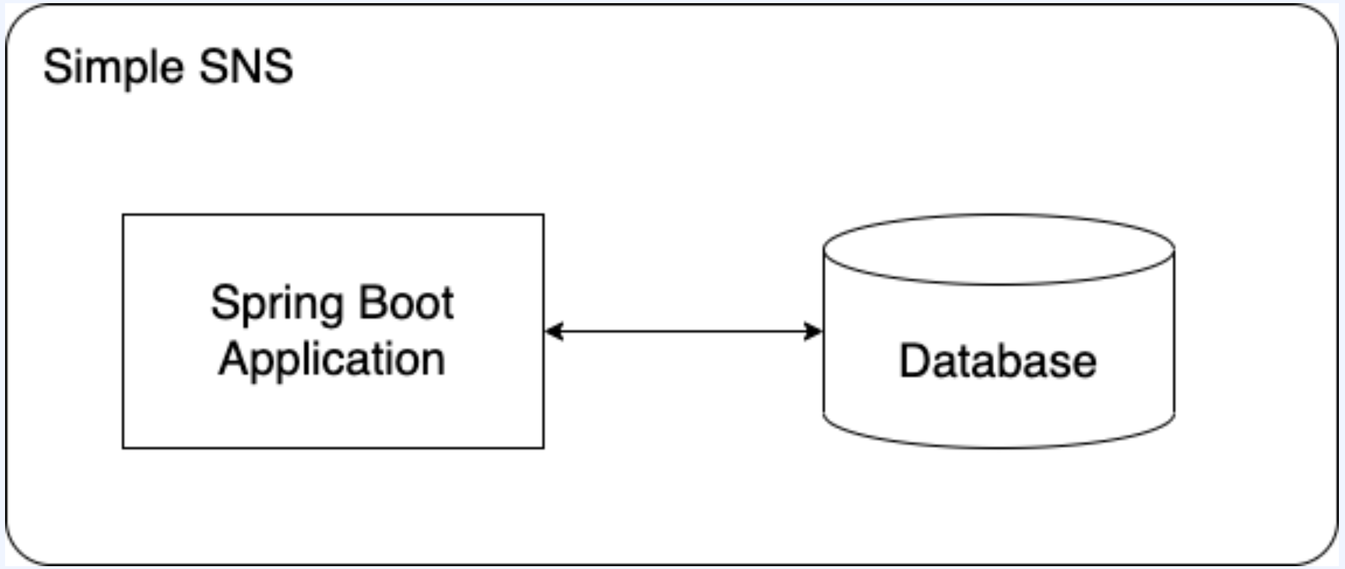
1 대규모 트래픽일시
현 architecture의 문제점 분석 및 해결법

지금까지 간단하게 SNS 서비스를 개발하여 보았습니다.

만약 이 SNS 서비스의 이용자들이 늘어나 트래픽이 증가한다고 가정해봅시다.

어떤 부분에서 문제가 생길 수 있을까요?

대규모 트래픽일시 현 architecture의 문제점 분석 및 해결법



문제점 1.

Token 인증시 user를 조회하고, 그 이후 또 user를 조회하는 구조

문제점 2.

매 API 요청시마다 조회하는 User

문제점 3.

Alarm까지 생성해야 응답을 하는 API

문제점 4.

Alarm List API를 호출해야만 갱신되는 알람 목록

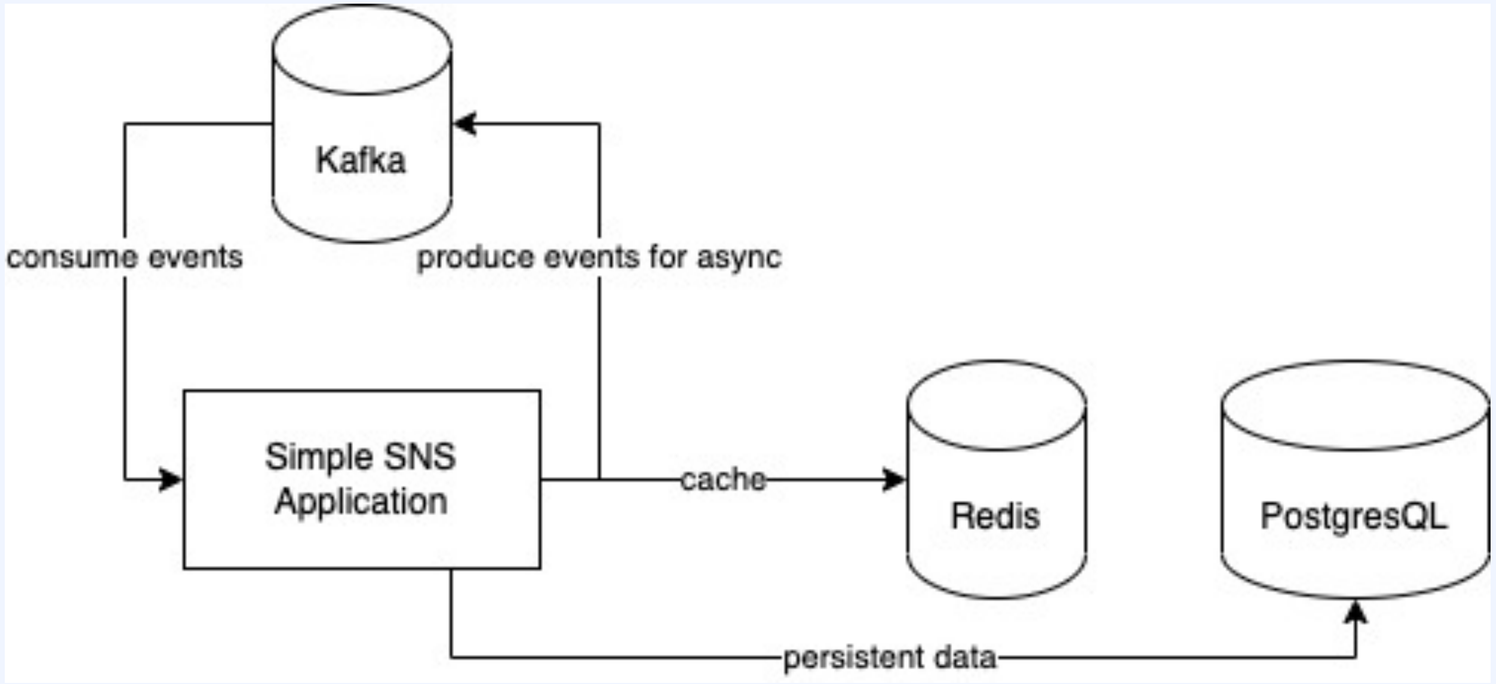
문제점 5.

데이터베이스로 날라가는 Query들은 최적화가 되었을까?

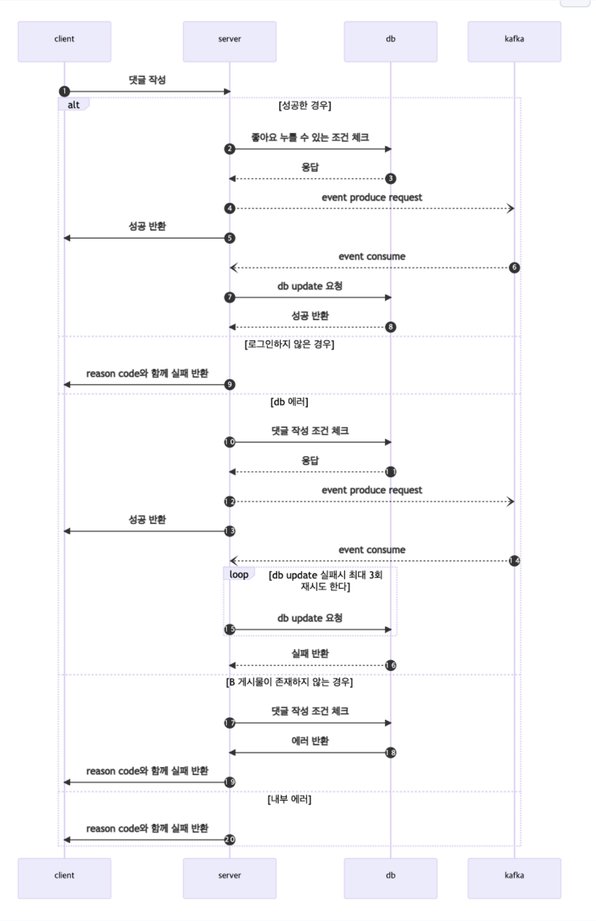
대규모 트래픽일시 현 architecture의 문제점 분석 및 해결법

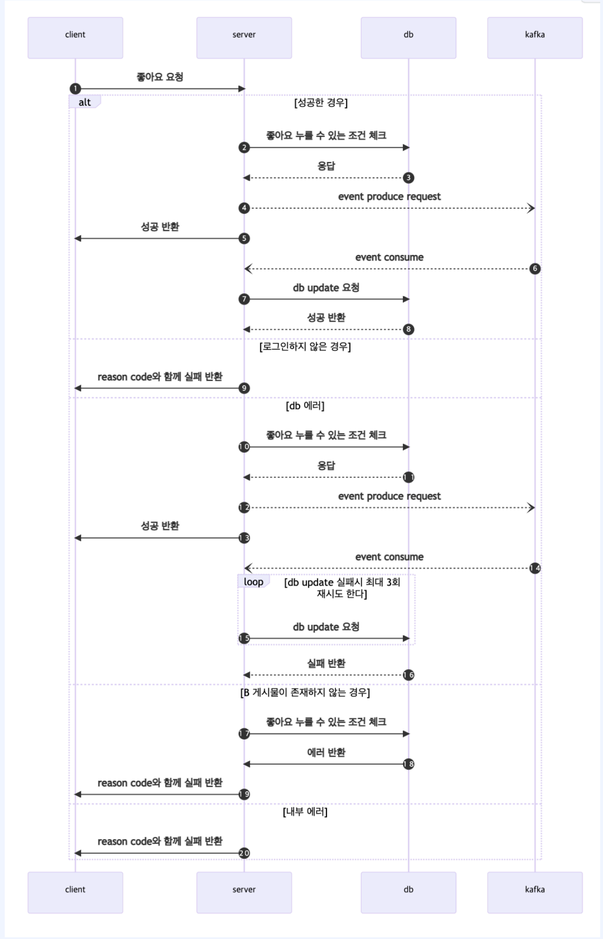
1. 코드의 비 최적화
2. 수많은 DB IO
3. 기능간의 강한 결합성

대규모 트래픽일시 현 architecture의 문제점 분석 및 해결법



대규모 트래픽일시 현 architecture의 문제점 분석 및 해결법





대규모 트래픽 고려해보기

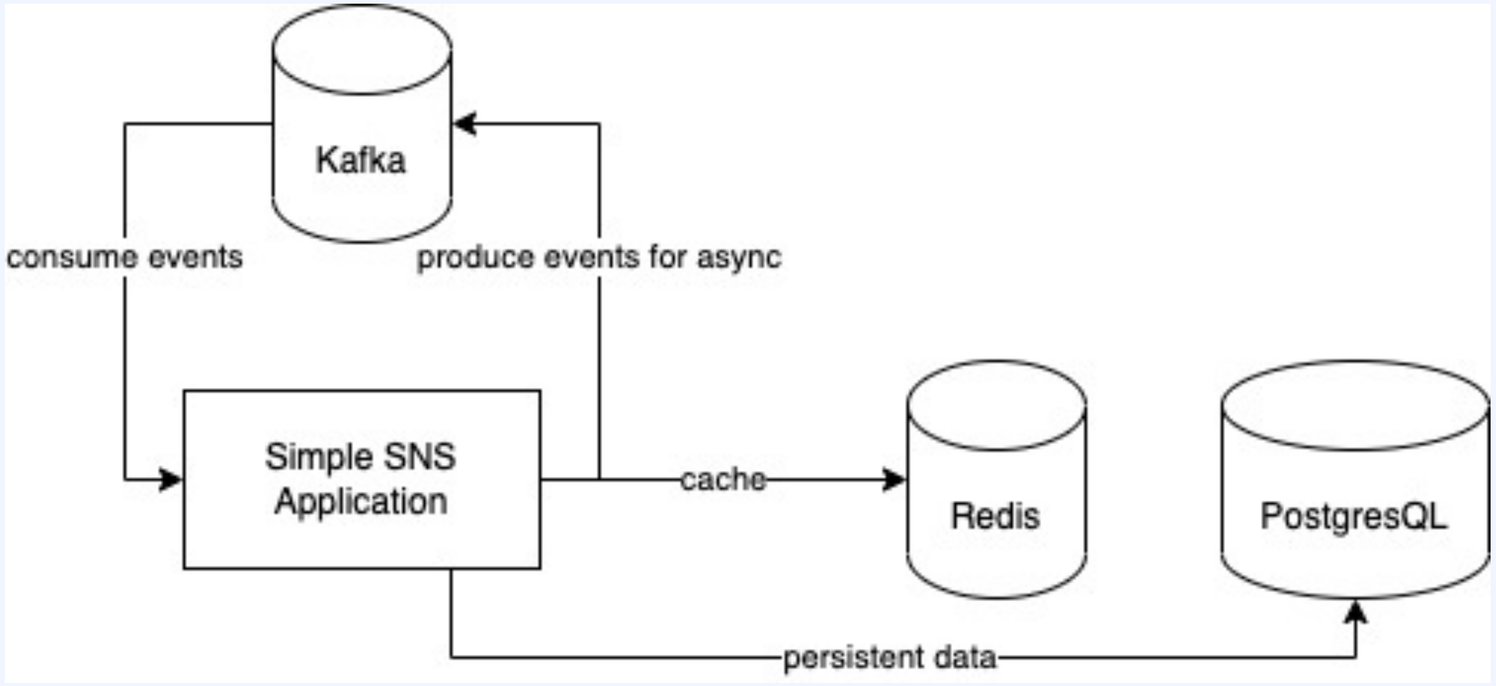
2 코드 최적화하기

함께 진행해 봅시다!

대규모 트래픽 고려해보기

3 Redis 소개하기

수정된 architecture



Redis VS Local Caching

	Redis	Local Caching
특징	In-memory 데이터베이스로 key-value 형태의 데이터베이스. 다양한 command를 제공하며 single thread이다.	서버 내에 caching 하는 방법. 서버와 라이프사이클을 함께할 수 있다.
장점	여러 instance가 하나의 데이터를 공유할 수 있다. 다양한 command를 지원한다.	네트워크를 타지 않기 때문에 Redis에 비해 비교적 빠르다.
단점	Local caching에 비해서는 느리다.	여러 instance로 구성된 서버의 경우 캐시를 공유할 수 없다.

Redis

- In-memory 데이터베이스로 key-value 형태의 데이터베이스이다.
- 다양한 command를 제공한다
- Single thread이다
- 높은 availability를 위한 몇가지 옵션이 존재한다.
  - Sentinel
  - Redis Cluster

https://redis.io/

대규모 트래픽 고려해보기

4 caching을 통해 DB 호출수 줄여보기

caching을 통해 DB호출수 줄여보기

4.
caching을 통해
DB호출수 줄여보기

함께 진행해 봅시다!

대규모 트래픽 고려해보기

5 SSE 소개하기

주기적으로 Client에서 데이터를 가지고 와야하는 경우

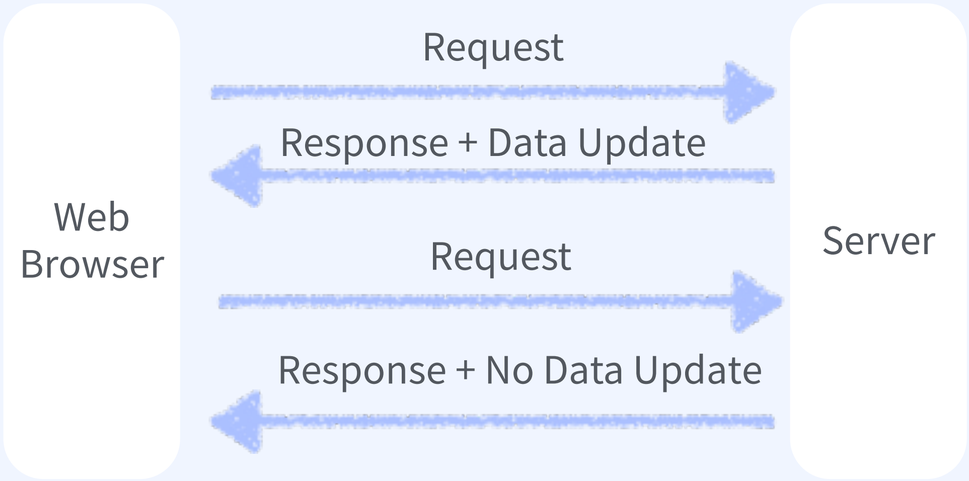
Polling

일정 주기를 가지고 서버의 API를 호출하는 방법.

실시간으로 데이터가 업데이트 되지 않는다는 단점이 있다.

불필요한 요청이 발생하며 따라서 불필요한 서버 부하가 발생한다.

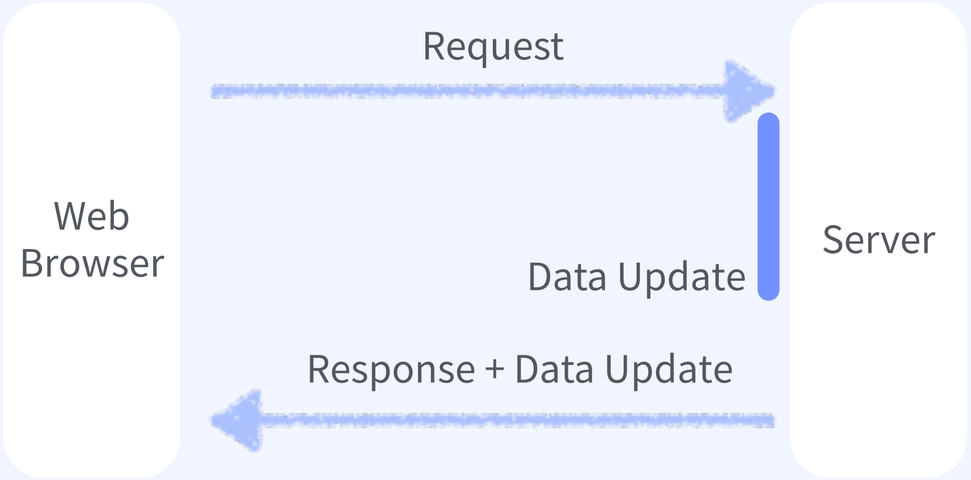
다만 호환성이 좋다는 장점이 있다.



주기적으로 Client에서 데이터를 가지고 와야하는 경우

Long-Polling

서버로 요청이 들어올 경우 일정 시간동안 대기 하였다가 요청한 데이터가 업데이트 된 경우
서버에서 웹브라우저로 응답을 보낸다.
연결이 된 경우 실시간으로 데이터가 들어올 수 있다는 장점이 있다.
따라서 Polling보단 개선된 형태이지만 데이터 업데이트가
빈번한 경우 Polling과 유사하다.



주기적으로 Client에서 데이터를 가지고 와야하는 경우

SSE (Server-Sent Event)

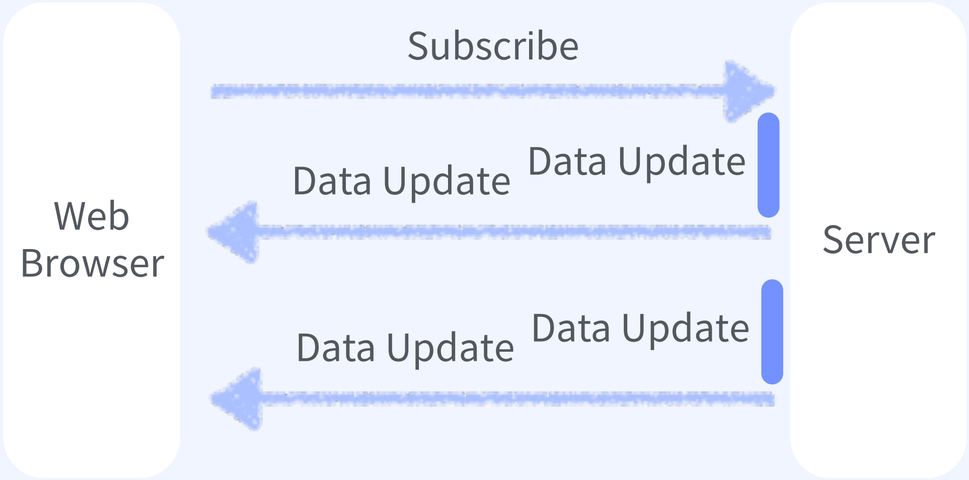
서버에서 웹브라우저로 데이터를 보내줄 수 있다.

웹브라우저에서 서버쪽으로 특정 이벤트를 구독함을 알려준다.

서버에서는 해당 이벤트가 발생하면 웹브라우저쪽으로 이벤트를 보내준다.

다만 서버에서 웹브라우저로만 데이터 전송이 가능하고 그 반대는 불가능하다.

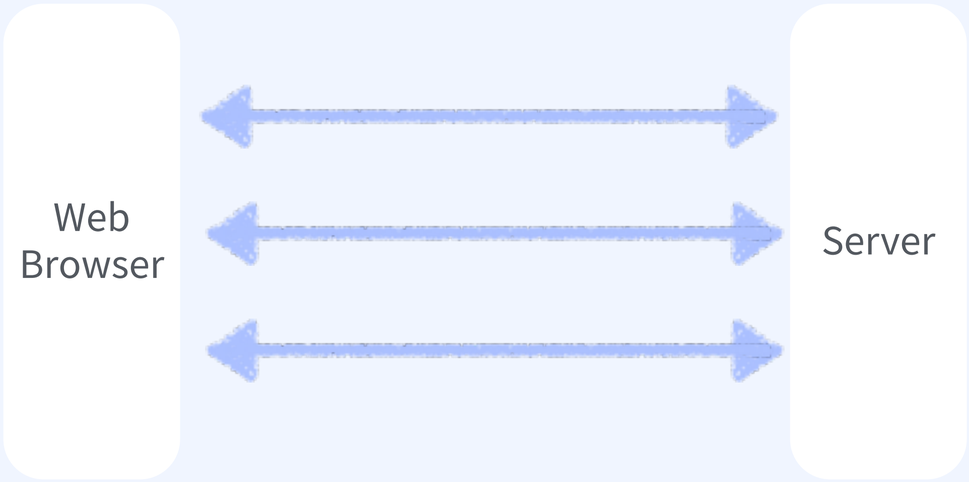
또한 최대 동시 접속 횟수가 제한되어 있다.



주기적으로 Client에서 데이터를 가지고 와야하는 경우

WebSocket

서버에서 웹브라우저사이 양방향 통신이 가능한 방법이다.



대규모 트래픽 고려해보기

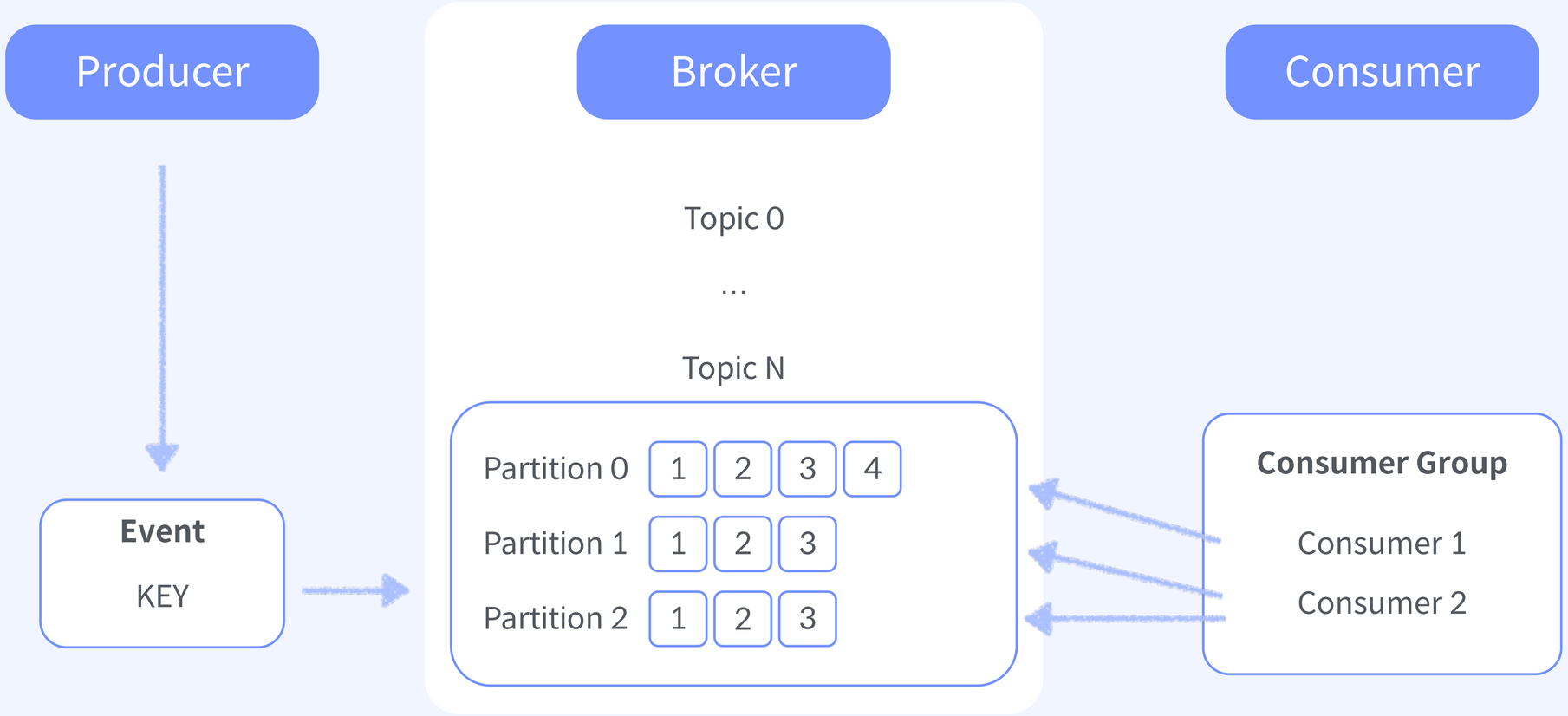
6 sse를 통한 알람 개선하기

함께 진행해 봅시다!

대규모 트래픽 고려해보기

7 Kafka 소개하기

Kafka 소개하기



https://kafka.apache.org/documentation/

대규모 트래픽 고려해보기

8 비동기적으로 데이터 처리하기

함께 진행해 봅시다!