개발자를 위한 Redis

6장. 레디스를 메시지 브로커로사용하기

메시지 브로커
서비스 간 통신이 불가능한 상황이 바로 장애로 이어지지 않게, 당장 메시지를 처리하지 못하더라도 보낸 메시지를 쌓아둔 뒤 나중에 처리할 수 있는 채널을 만들어 주는 것

메시지 브로커의 형태

1. 메시징 큐
- 데이터를 생성하는 쪽을 생산자, 데이터를 수신하는 쪽을 소비자
- 생산자는 소비자의 큐로 데이터 push.
- 소비자가 데이터를 읽어갈 때 큐에서 데이터 삭제
- 1:1 상황에서 한 서비스가 다른 서비스에게 동작을 지시할 때 유용

2. 이벤트 스트림
- 데이터 생성하는 쪽을 발행자, 데이터를 조회하는 쪽을 구독자
- 특정 저장소에 하나의 메시지를 보낼 수 있고 소비자들은 메시지를 pull
- 저장소 설정에 따라 특정 기간 동안 저장
- N:M 상황에서 유용

레디스를 메시지 브로커로 사용하기

레디스의 pub/sub에서 모든 데이터는 한번 채널 전체에 전파된 뒤 삭제되는 일회성의 특징을 가지며, 메시지가 잘 전달됐는지 등의 정보는 보장하지 않는다.
따라서 완벽하게 메시지가 전달되어야 하는 상황에는 적합하지 않을 수 있지만 **fire-and-forget** 패턴이 필요한 간단한 알림 서비스에서는 유용하게 사용 될 수 있다.

Fire-and-forget : 어떤 작업을 실행하고 그 결과에 대한 응답을 기다리지 않고 바로 다음 코드를 실행하는 디자인패턴

메시지 publish 하기
Publish hello world

메시지 구독하기
SUBSCRIBE event1 event2

클러스터 구조에서도 pub/sub을 사용할 수 있는데 메시지를 발행하면 해당 메시지는 클러스터에 속한 모든 노드에 자동으로 전달된다.
하나의 노드에 메시지를 발행하면 모든 노드에 전파되는 클러스터 환경의 핵심 목표와 부합하지 않는다.
레디스 7.0에서는 **sharded pub/sub** 기능이 도입되어 같은 슬롯을 가지고 있는 노드 간에만 pub/sub 메시지를 전파한다.

SPUBLISH apple a
SSUBSCRIBE apple a

레디스 list를 메시징 큐로 사용하기

큐의 tail과 head에서 데이터를 넣고 뺄 수 있는 LPUSH, LPOP, RPUSH, RPOP 커맨드가 존재하기 때문에 애플리케이션 특성에 맞는 메시징 큐를 직접 구현할 수 있다는 장점

RPUSHX는 데이터를 저장하고 하는 list가 이미 존재할 때에만 아이템을 추가하는 커맨드. 접속 빈도가 낮으면 굳이 캐시 데이터를 관리 할 필요 없음

List의 블로킹 기능
폴링 프로세스가 진행되는 동안 애플리케이션과 큐의 리소스가 불필요하게 소모될 수 있다.
BRPOP 과 BLPOP 커맨드를 사용하여 list에 데이터가 있으면 즉시 반환하고, 데이터가 없을 시 타임아웃 시간만큼 대기한 후에 nil 값을 반환한다.
타임아웃을 0으로 설정하면 데이터가 리스트에 들어올 때까지 제한없이 기다리고, 리스트에 데이터가 입력되면 가장 먼저 요청 보낸 클라이언트가 데이터를 가져간다.

BRPOP은 RPOP과 다르게 2개의 데이터(첫번째는 pop된 list의 키, 두번째는 데이터 값) 를 반환하는데 이는 여러 개의 리스트에서 대기할 수 있게 하기 위해서다.

List를 이용한 원형 큐
만약 특정 아이템을 계속해서 반복 접근해야 하는 클라이언트, 혹은 여러 개의 클라이언트가 병렬적으로 같은 아이템에 접근해야 하는 클라이언트에서는 원형 큐를 이용해 아이템을 처리하고 싶을 수 있는데 이때 RPOPLPUSH 커맨드를 사용하면 간편하게 원형 큐를 사용할 수 있다.

레디스의 Stream과 아파치 카프카
레디스 5.0에서 새로 추가된 자료 구조로 대용량, 대규모의 메시징 데이터를 빠르게 처리할 수 있도록 설계됐다.

사용목적에 따라 두가지 방식으로 활용될 수 있다.
1. 백엔드 개발자들이 stream을 대량의 데이터를 효율적으로 처리하는 플랫폼으로 활용
2. 데이터 엔지니어들이 stream을 여러 생산자가 생성한 데이터를 다양한 소비자가 처리할 수 있게 지원하는 데이터 저장소 및 중간 큐잉 시스템으로 사용

스트림이란?
1. 연속적인 데이터의 흐름, 일정한 데이터 조각의 연속을 의미
2. 끝이 정해지지 않고 계속되는 불규칙한 데이터를 연속으로 반복 처리할 때 이 또한 스트림 처리 한다고 부를 수 있다.

카프카와 레디스 비교

1. 카프카
- 스트림 데이터는 **토픽**이라는 개념에 저장된다. 토픽은 각각의 분리된 스트림을 뜻하며, 같은 데이터를 관리하는 하나의 그룹을 의미한다.
- 각 메시지는 0부터 시작해 증가하는 시퀀스 넘버로 식별할 수 있는데, 이때 시퀀스 넘버는 토픽 내의 파티션 안에서만 유니크 하게 증가하기 때문에 토픽이 1개 이상의 파티션을 갖는다면 유니크 하지 않다.
- 생성자는 데이터를 토픽에 푸시하며, 소비자는 토픽에서 데이터를 읽어간다. 데이터를 저장하기 위해 토픽을 먼저 생성한 뒤, 프로듀서를 이용해 메시지를 보낸다
- 데이터 조회시 소비자는 특정 토픽을 실시간으로 리스닝하며, 새롭게 토픽에 저장되는 메시지를 전달받을 수 있다.

2. 레디스
- stream에서 각 메시지는 시간과 관련된 **유니크한 ID**를 가지며, 이 값은 중복되지 않는다.
- 따로 stream을 생성하는 과정은 필요하지 않으며 **XADD** 커맨드를 이용해 새로운 이름의 stream에 데이터를 저장하면 저장과 동시에 자료구조가 생성된다.
- 데이터 조회시 카프카처럼 실시간으로 처리되는 데이터 리스닝하는 방법과 ID를 이용해 필요한 데이터를 검색하는 방법이 있다.

XREAD : 실시간 조회
BLOCK 0 : 더 이상 가져올 데이터가 없더라도 연결 끊지 말고 계속 리스닝 하라는 의미
STREAMS Email 0 : email이라는 stream에 저장된 데이터 중 ID가 0보다 큰 값을 읽기 $를 입력하면 커맨드를 실행한 이후의 메시지만을 가져올 수 있다.
XRANGE : ID를 이용해 원하는 시간대의 데이터를 조회

소비자와 소비자 그룹
같은 데이터를 여러 소비자에게 전달하는 것을 **팬아웃** 이라 한다.
레디스 Stream에서는 데이터가 저장될 때마다 고유한 ID를 부여받아 순서대로 저장되므로 소비자에게 데이터를 전달될 때 순서가 보장되지만 카프카의 유니크 키는 파티션 내에서만 보장되기 때문에 소비자가 여러 파티션에서 읽으면 데이터 순서를 보장할 수 없다. 메시지가 토픽에 저장될 때 해시함수에 의해 파티션에 랜덤으로 분배되기 때문이를 위해서 카프카는 소비자 그룹을 사용해야 한다.

ACK와 보류 리스트
레디스 stream에서는 소비자 그룹에 속한 소비자가 메시지를 읽어가면 각 소비자별로 읽어간 메시지에 대한 보류리스트를 새로 생성하며, 마지막으로 읽어간 데이터의 ID로 last_delivered_id값을 업데이트 하여 해당 소비자 그룹에 마지막으로 전달한 ID가 무엇인지 파악해 중복 전달을 막는다.

보류리스트를 이용해 소비자가 처리한 데이터를 파악할 수 있고 데이터가 처리됐다는 뜻의 ACK를 보내면 보류리스트에서 삭제한다. 이를 통해 장애가 발생해도 메시지를 놓치지 않고 처리할 수 있다.

at most once
메시지를 최소 한번 보내는 것을 의미한다. 소비자는 메시지를 받자마자 실제 처리하기 전 먼저 ACK를 보내는데 속도는 향상되지만 실제 처리되지 못한 데이터는 잃을 수 있다. 메시지가 일부 손실되더라도 빠른 응답이 필요한 경우 선택하는 전략이다.

at least once
소비자는 받은 메시지를 모두 처리한 뒤 ACK를 보낸다.

exactly once
모든 메시지가 무조건 한번씩 전송되는 것을 보장한다는 의미이다.

메시지의 재할당
소비자별 보류 리스트를 유지하지만 소비자 서버에 장애가 발생해 복구되지 않는다면 해당 소비자가 처리하던 보류중인 메세지들은 다른 소비자가 대신 처리해야 한다.
XCLAIM 커맨드를 이용하면 메시지의 소유권을 다른 소비자에게 할당할 수 있다. XCLAIM 커맨드를 사용할 때에는 최소 대기시간을 지정해야 한다.
XAUTOCLAIM 커맨드를 이용하면 보류했던 메시지 중 하나를 자동으로 가져와서 처리할 수 있도록 해서 반복적 호출이 가능해진다.

Stream 내의 각 메시지는 counter라는 값을 각각 가지고 있는데 XREADGROUP을 이용해 소비자에게 할당하거나 XCLAIM 을 이용해 재할당할 경우 1씩 증가한다.
만약 메시지가 문제가 있어 counter가 특정 값에 도달하면 특수한 다른 stream에 보내 관리자가 추후 처리할 수 있도록 하는 것이 현명하다. 이런 메시지를 dead letter라 부른다.

Stream 상태 확인
XINFO 커맨드를 이용해 stream의 여러 상태를 확인할 수 있으며, 이때 사용할 수 있는 기능은 help 커맨드로 확인할 수 있다.