장아령

Backend Engineer | ahryeong.jang@gmail.com | 010-3073-0252 | github.com/aristo0922

- 변화가 많은 스타트업 환경에서도 빠르게 학습하고 적응하며 아키텍처 최적화와 성능 개선을 주도했습니다.
- 모니터링과 부하테스트를 기반으로 문제를 탐색하고 최적화를 통해 API 응답 속도를 -46%, 처리량은 137배 향상시킨 경험이 있습니다.

경력

Java-SpringBoot, Python, MySQL, PostgreSQL, Docker, Redis

Contacto Link: 아티스트 랜덤 매칭 플랫폼 | wakawaka 인턴 25.02~ 재직중 Backend Engineer

[Redis Stream] 를 사용해 알람 서버를 비동기 서버로 전환하기 위해 이벤트 리스너를 구현하고 MicroService 의 결합도를 낮춤
- Spring Eureka **기반의 동기 통신 구조**로 user-service 와 alarm-service 가 FeignClient 로 통신하여 알림 전송 마무리까지 대기하여 응답 지연
  - success 처리할 수 있는 요청이 **알림 서비스에서 일어난 장애가 전파**되어 실패 응답 반환 및 **매칭 데이터 유실**되는 상황 확인
- 메시지 중복 전송 및 유실 가능성이 있는 Redis Pub/Sub 대신, **at-least-once 동작을 보장**하는 Redis Stream 기반 이벤트 큐로 전환
- 알림 서버에 Redis Stream **메시지를 주기적으로 polling** 하여 처리하는 **Stream Listener 를 구축**하고 **실패한 요청에 대한 재시도 로직** 구현
  - **알림 서비스를 완전한 비동기 구조로 개선하고 서비스 간 결합도를 낮춰 안정성 확보**

[Redisson **분산락**] 을 이용해 서로 다른 두 사용자 매칭 결과가 되는 채팅방이 중복 생성되는 동시성 이슈 해결
- **'좋아요' 요청이 매칭으로 이어져 채팅방을 생성하는 과정에서 연속적인 클릭으로 '좋아요' 중복 요청으로 채팅방도 중복 생성되는 문제**
- Swift 에서 중복 클릭 방지 로직을 적용했지만 여전히 상대방에 대한 '좋아요' 기록과 매칭된 이용자 간 '채팅방'이 중복 생성되는 상황 반복
  - 두 사용자가 동시에 '좋아요' 요청을 보내는 상황을 테스트 한 결과 **신규 매칭 여부를 파악하는 과정에서 동시성 이슈 발견**
  - 좋아요 데이터 생성 후, **매칭 여부를 파악하는 과정**에서 동시에 매칭 여부를 판별하고 동시에 채팅방을 생성하는 것이 원인으로 확인
- **사용자 pk 를 이용한 Lock-key 를 생성하고** '좋아요' 데이터 상태 변화 및 매칭 판별 로직 전체에 **분산락을 걸어 원자적으로 처리**
  - **동일 사용자 간 동일한 Lock-key 를** 공유하기 위해 더 작은 사용자 pk 를 *'lock:like:{userId1}:{userId2}'* 에서 *{userId1}*에 위치하게 설계

Link: Redisson 분산락으로 사용자 매칭 중복 생성 문제 해결기

[Redis Cache] 에서 로그인 토큰을 관리하도록 하여 낙관적 락 문제 해결 및 성능 개선
- Datadog **모니터링 중 SignIn API 처리 속도가 평균 4.6 초,** 에러 발생 시에는 요청 처리시간 5~10 초까지 지연되는 현상 발견
- K6 테스트(50 VUs×30s) 결과, JPA 에서 token 처리 중 **낙관적 락** 반복으로 **처리량 4 회(0.13 iter/s), 평균 응답 4.57s, 오류율 100% 기록**
- token 조회/삭제/생성으로 DB 에 다수 접근하는 로그인 로직을 **Redis 단일 키 갱신 방식으로 개선하여 동시성 충돌 및 트랜잭션 병목 제거**
- 동일 환경 부하테스트 진행 결과 **낙관적 락 예외율 100%→0%, 평균 응답 4.57s→2.47s(-46%), 처리량 0.13→17.9 iter/s 대폭 개선**
  - **동시성을 안정적으로 제어**하고 **처리량을 향상**시켜 주 1,2 회 발생하던 로그인 관련 사용자 문의가 개선 후 0 건 유지중

Link: Redis 의 원자성(Atomicity)으로 로그인 API 의 동시성과 성능을 한 번에 잡다

[DataDog **과 슬랙**] 기반 실시간 알림 시스템을 도입하여 대응시간 단축
- 서비스에 전반적인 성능 개선이 필요하나 팀원 간 **최적화 진행 우선순위에 대한 견해가 분분**했고, **개선 정도에 대한 지표** 필요성 절감
- 배포 서버에서도 **오류 대응 시스템 부재**로 인해 운영팀으로 문의가 들어온 후 비로소 문제를 인식하는 상황 다수 발생
  - 외부 공격이나 예상치 못한 에러로 서버가 다운된 경우 즉각 알릴 수 있는 **자체적인 서비스 오류 알림 체계 구축 필요성 절감**
- Slack **웹훅** 기반 실시간 장애 알림 시스템을 구축하여 **개발·운영팀의** MTTD **를 단축**하고, **문제 인지 속도 및 협업 효율 개선**

[그 외] 서비를 관리 및 운영하는 관리자 페이지 구축

프로젝트

MyTube Link: 영상 스트리밍 서비스 | F-lab 멘토링 24.05~24.07 Backend Engineer

[불필요한 테이블 스캔 감소]
- 사용자 엔티티와 다대일 관계인 파티 엔티티 조회 시 추가적인 쿼리가 실행되는 N+1 문제 발생
- 직접 JPQL 을 작성하여 Fetch Join 을 명시적으로 사용하여 테이블 조회 최적화

[단위 테스트를 통한 개발 효율성 증대]
- Mockito 와 JUnit5 를 활용하여 GWT pattern 으로 작성한 단위테스트 기반으로 개발 효율성 증대

대외 활동

- **F-lab/** MyTube Spring 백엔드 코스 멘토링 | 24.02~ 24.07
- **Techeer/** SilliconValley 문화 기반 국내외 개발자 커뮤니티 | 23.09~ (활동중)
- **SilliconValley Bootcamp/** T4Y: This is for you | 23.06~23.08

한국공학대학교 컴퓨터공학부 소프트웨어 전공 | 19.03~ 24.12