안녕하세요. 최원준입니다.
Backend Engineer

Phone. (+82) 10-4843-0882
Email. 724thomas@gmail.com
GitHub. github.com/724thomas
Blog. wonjoon.gitbook.io/joons-til
Portfolio. 노션 포트폴리오

Introduction

정량적 지표를 바탕으로 안정성을 보장하도록 노력하는 백엔드 개발자입니다.

- 다양한 도메인 프로젝트를 경험하며 정량적 데이터 기반으로 아키텍처 개선과 병목 분석 습관을 기르고 있습니다.
- 팔로잉 시스템 개선으로 알림 시스템 성능 개선과 서비스 신뢰성 하락 상황을 대비한 경험이 있습니다.
- 분산락과 보상 트랜잭션을 적용하여 데이터 정합성과 동시성 제어를 통해 안정적인 거래 처리를 하였습니다.
- 가설 수립과 성능 검증을 통해 **RDB** 샤딩에서 **Cassandra**로 전환하여 확장성과 고가용성을 확보하였습니다.

1등 서비스를 만들고 싶은 욕심이 강한 개발자입니다.

- 프로젝트의 성공을 위해 개발 외적인 부분까지 고민하고 직접 실행에 옮겼습니다.
- 그래픽 인턴십에서는 직접 서비스를 사용하며 기능적, 비즈니스적 개선사항들을 팀에 제안했습니다.
- 카카오브레인 인턴십에서는 서비스 검증을 위해 실제 강사와 대학생을 내돈내산 섭외해 성능 검증을 진행했습니다.

Skill

Backend: Java, Spring, JPA

DB: MySQL, Redis

Infra: Docker, AWS, RabbitMQ

Ai Tools: Cursor, MCP

Career

* 파란 글꼴은 블로그 포스트로 연결됩니다.

그래픽 개발팀 / Software Engineer 인턴 [2024.01 ~ 2024.05]

사용 기술 : **SpringBoot, JPA, S3, Redis, MySql, AWS, Whatapp**
API 개발과 최적화를 적극적으로 하며 개발, 출시, 운영 전과정을 경험했습니다.

Graphic: SNS 만화 서비스 백엔드 개발 5인 **(BE 1, Mobile 1, FE 1, UI 1, PM 1)**

1. 빠르게 증가하는 트래픽에 대응하기 위해 인덱스/쿼리 최적화 및 **MView** 스테이징 적용
    - 실시간 서비스 품질 개선: 유저 목록 조회 API의 평균 지연을 ~600ms → ~100ms로 단축(약 **70%** 개선)
    - 인덱스 설계와 쿼리 최적화: 쿼리 실행 계획 분석 및 인덱스 최적화(팔로워 정렬, 차단 유저 복합 인덱스)
    - 서빙 전용 테이블 구조 확립: 역정규화 MView 도입 + 3시간 주기 스테이징 스왑(atomic rename)
    - 운영 단순화: 관리 포인트 증가로 자동화 대신 문서화 채택, Redis 도입은 복잡도 대비 이득 낮아 보류

2. 알림 시스템 안정성 및 최종 일관성 확보
    - 구독/취소 요청 로직 재정립: 구독은 DB→FCM, 구독 취소는 FCM→DB 순서로 처리, 데이터 불일치 위험 최소화
    - 스케줄 기반 재시도 + 취소 전용 보강 처리: 실패 요청 테이블 기록. 단 구독 취소만 횟수 제한 적용
    - 구독 실패는 최소 영향, 취소 실패는 신뢰 타격 분석: 구독 취소 실패는 신뢰도 타격 → 반드시 원자성 + 재시도 보장
    - 성과: 실제 장애 발생 전 사전 예방 리팩토링으로 FCM/DB 불일치 가능성 제거, 서비스 신뢰성 강화

3. 서버리스 이미지 처리 최적화
    - **8K+** 이미지 처리 실패: Lambda 메모리 초과로 처리 실패 발생
    - **sharp** 메서드 체이닝으로 메모리 사용 최소화: 중간 버퍼 최소화, 고해상도(8K+) 처리 시 메모리 사용량 절감
    - 다중 파일 처리 레거시 제거 및 단일화: 단일 파일 처리 구조로 단순화, if문·max 중복 제거
    - **Lambda** 메모리 지표 실시간 모니터링: 메모리 사용 패턴 추적 및 이상 탐지
    - 성과: 이미지 리사이징·크롭·압축 성능 향상, Lambda 메모리 효율 최적화 및 비용 절감, 실패율 **0%** 달성

4. 서버 로그 아카이빙 자동화
    - **CloudWatch**: 애플리케이션 로그 수집 및 2주 경과 데이터 자동 마이그레이션.
    - **S3** 단기 보관: 2주 이상 로그는 CloudWatch -> S3로 이전 출시 초기 장애 분석과 대응에 적합. 비용 절감 **95%**
    - **Glacier** 장기 보관: 1개월 이상 장기 로그는 S3→Glacier로 이전, 검색은 제한적. 비용 절감 **99%**
    - **Lambda** 기반 로그 전송 자동화: Ephemeral Storage 활용 대용량 로그 안전 처리
    - **EventBridge** 스케줄링 자동 아카이빙: 수집·전송·보관까지 전체 파이프라인 무인화.
    - 성과: 로그 저장 비용 절감, Out-of-Memory 위험 제거 및 추후 대응책 문서화, 안정적 장기 보관 체계 확보.

카카오 브레인 언어모델사업팀 / Software Engineer 인턴 [2023.06 ~ 2023.08]

사용 기술 **: Spring Boot, FastAPI, JPA, GCP, GPT-3.5 Turbo, MySQL, PostgreSQL**
서술형 평가 첨삭 MVP에서 기획, 백엔드 개발, 검증을 담당하며 응답 속도와 정확성 개선에 집중했습니다.

Peterpen: LLM 기반 서술형 문제 첨삭 서비스 백엔드 개발 4인 (BE 2, Mobile 2)

1. **LLM** 호출 최적화 및 과채점 방지
    - **api**호출 병렬화로 응답 지연 단축: 순차→병렬시 정확도에 큰 영향이 없는 것을 확인
    - **PostgreSQL** 캐싱 레이어로 중복 호출 제거: A → B → A 상황 제거
    - 프롬프트 엔지니어링: 다양한 실험으로 openai API 응답 정확도·일관성 강화
    - 성과: API 응답 속도 최대 87% 단축, 할루시네이션 방지 및 신뢰도 향상

아이톡시 해외영업팀 / 해외영업 및 무역업 사원 [2021.07 ~ 2022.04]

사내 개인 프로젝트: 물류 관리, 의사결정 지원 프로그램 개발

삼성 엔지니어링 발전사업부 / 프로젝트 매니징 인턴 [2012.01 ~ 2012.06]

Shaybah #4 프로젝트 관리 지원 업무를 담당, 일정·문서 관리 및 협력사(Saudi Aramco) 커뮤니케이션을 수행

Project

Ecom 상품 등록, 주문 결제까지의 도메인 중심 Ecommerce 플랫폼 개발 - Github Repo

주요 사용 기술 : **Spring Boot, RBAC, Redisson Lock, JPA Pessimistic Lock**

1. 재고 동시성 제어 문제 해결: 이중 락 기반 동시성 제어 시스템 설계 및 구현
   - 고객 불만 및 비즈니스 리스크 초래: 동시 주문 시 재고 중복 차감 발생 → 오버셀링 및 신뢰성 저하4
   - 핵심 접근법: Redis 분산락(인스턴스 간 제어) + JPA 비관적 락(DB 레벨 보호) 조합으로 이중 락 메커니즘 설계
   - 예약-확정 단계 분리 및 **MultiLock** 적용: 데드락 방지, TTL 기반 예약 만료 관리
   - 단위/통합 테스트 및 실제 부하 검증: 재고 중복 차감 0건·락 해제 100% 보장
   - 성과: 대규모 동시 주문 시 시스템 신뢰성 강화 및 데이터 일관성 확보 + 안정적 응답 시간 유지

Twitter Clone Tweeter 핵심 API 구현 및 기술 비교 분석 - Github Repo

주요 사용 기술 : **Spring Boot, Cassandra, Rate Limiter, Redis Cluster**

1. 트윗 생성 성능 향상: **RDB** 샤딩과 **Cassandra**의 성능 차이 분석을 위한 시뮬레이션
   - 시뮬레이션 환경 구성: Fan-out-on-write 패턴 적용 (MySQL 4-Shard vs Cassandra 3-Node)
   - 팔로워 규모별 단계적 부하 테스트: 팔로워 1 → 100,000명 시나리오, LogTrace AOP 기반 정밀 측정
   - **Cassandra** 성능 우위 입증: RDB 대비 **3~24**배 빠른 성능 달성 (100,000명: RDB 13.8s vs Cassandra 3.9s)
   - 배치·병렬 처리 **vs** 순차 처리 방식 비교: LSM Tree 기반 쓰기 최적화 + 배치·병렬 처리 vs RDB 샤딩·순차 처리
   - 성과: 100K 팔로워 환경에서 안정적 성능 확보 및 대규모 소셜 미디어 워크로드에서 NoSQL 확장성·성능 우수성 실증

2. 인플루언서를 고려한 하이브리드 **Fan out** 전략 적용
   - 배경: Fan-out-on-write 전략을 사용, 팔로워 수가 수백만에 달하는 계정에서 지연과 시스템 부하 발생
   - 기본 전략(**Fan-out-on-write**): Cassandra BatchStatement + ThreadPool(8) 기반 병렬 처리 적용
   - 보완 전략(**Fan-out-on-read**): 인플루언서 계정 트윗은 타임라인 조회 시점에 계산하도록 설계
   - **Consistency Level** 조정으로 응답 최적화: `QUORUM` → `ONE` 변경, 응답 시간 **50%** 단축 (최종 일관성 허용)
   - **RabbitMQ** 기반 재시도/**Dead Letter** 처리: 원본 트윗 저장 보장, Fan-out 실패 시 재시도 및 DLQ 처리
   - 성과: 일반 사용자 → 즉시 반영되는 실시간 타임라인, 인플루언서 → 트윗 생성 API 지연 급증 해소

Leetcode Clone Leetcode 핵심 API 구현 애플리케이션, 아키텍처 레벨 최적화 - Github Repo

주요 사용 기술 : **Spring Boot, AWS, Redis, RDS, K6, Grafana, Prometheus, RabbitMq**

1. 대규모 동시 제출 부하테스트: **Timeout** 문제 해결과 **5**배 성능 개선 사례
   - 대규모 동시 요청 부하 시험: 비기능 요구사항(최대 **10,000**명 동시 제출 처리) 충족 여부 검증

- 동시 연결 과부하로 인한 장애 요인 파악: I/O Time-out·Request Time-out → 톰캣 커넥션 한계 및 응답 지연
- 톰캣 리소스 튜닝으로 **Dial I/O** 타임아웃 해결: max-connections 10,000으로 상향
- **RabbitMQ + Sandbox** 분리로 **API** 병목 완화: 코드 실행을 Sandbox 서버로 위임, 병목 해결
- 결과 처리 서버 전용 분리로 안정성 확보: Consumer 스레드 확장(최대 250), 결과 처리 서버 분리, 부하분산
- 성과: 10,000 요청 전량 성공 처리 달성, 최종 요청 응답 시간 64분 52초 → 1분 7초 단축 (약 **58**배 개선)

2. 정량적 데이터 기반 애플리케이션 레벨 최적화
   - 문제 목록 조회 비교 분석: Pagination vs Cursor
   - 문제 상세 정보 조회 비교 분석: Cache-aside 전략
   - 풀이 제출 비교 분석: 캐싱 & 멱등키 도입
   - 리더보드 조회 비교 분석: Redis 도입

Open Source

apache/gravitino [2024.01] PR#1359
- 윈도우 환경에서 발생할 수 있는 문제점 식별
- 클래스 동작 기반 equals, hashcode 재정의로 명확성↑

eclipse/jkube [2024.01] PR#2547
- 메서드에서 불필요한 `continue/goto` 문을 제거.
- 중첩 로직을 분리하여 코드 가독성과 유지보수성을 개선

Education

세종대학교 / 경영학 전공 [2011.03 ~ 2019.02 / 졸업]
패스트캠퍼스 / **KDT** 백엔드 개발자 국비교육 [2022.09 ~ 2023.04]
F-Lab / 백엔드 심화 프로젝트 **1:1** 멘토링 [2025.01 ~ 2025.05]

Activity

국제 대학생 연합 봉사활동 동아리
[2014-2015] BoonTour 부회장

Model UN 토론 대회
[2013.07 ~ 2013.07] Cuba 대표

Certificate

TOEIC 960 [2024.06]
990 [2018.08]

SQLD [2019.03]

경영 빅데이터 분석사 **2**급
[2015.09]

국제무역사 **1**급
[2013.08]