전수민

홍익대학교 컴퓨터공학 학사

백엔드 개발자

wjsaos2081@gmail.com

https://dalcheonroadhead.tistory.com

https://github.com/dalcheonroadhead

(1) 고객이 없는 서비스는 무용지물이란 생각으로 능동적인 요구사항 수집, 말하지 못한 불편도 해결
(2) 기다려야 하는 답답함을 지양, 반복적인 성능 테스트와 고도화로 앞서나가는 개발자
(3) 회사의 상황과 환경에 적합한 기술을 선택해 구현

구인구직 매칭 서비스 개발 2025.02 - 2025.04

'당근 알바'와 같이 소일거리를 대신 해줄 사람을 찾아 일을 맡기는 매칭 서비스 전체를 개발

Java, Spring, queryDSL, JDBC, MySQL, Redis, Jmeter, Grafana, Cloud Watch

고객 현 위치 반경 N km 내 소일거리 찾기 서비스

부하 테스트 및 쿼리 실행 계획 분석

- DB에 소일거리 데이터가 5만 건씩 늘어날 때마다 응답시간이 크게 우상향 하였고, 데이터 20만 건기준, 1000 RPS를 7분 지속 했을 시, 평균 3102ms, 최대 40초를 기록
- 실행 계획 분석 시 위도, 경도 복합 인덱스가 존재함에도 table full scan을 실행하며 느려짐을 파악
- DB 내 데이터 분포도를 30회 밀집, 분산해본 결과, 전체 데이터 중 2.33% 조회 시만 인덱스가 활용됨을 확인

쿼리 튜닝

- 옵티마이저가 보조 인덱스를 통한 랜덤 I/O 접근 비용을 과하게 높게 잡고 있다고 판단, 인덱스 강제 힌트 적용
- 커버링 인덱스를 생성하여 랜덤 I/O 접근 비용 감축
- 서브 쿼리문을 활용하여 거리 계산 결과를 쿼리 내부에서 재활용
- R-tree, Spatial Index와 B-tree 위치 기반 인덱스 활용 시 실행 속도 비교
- JPQL, native-query, queryDSL, JDBC로 전송 전략 각각 구현 후 부하테스트 후 최적의 전략 선택
- 쿼리 실행 속도 (1.2초 → 0.0102초) 개선, API 응답 시간 3102ms → 170ms로 개선

증명 자료

- 기술 보고서: https://github.com/dalcheonroadhead/spot-be
- 발표 자료: https://cs-study-soomin.my.canva.site/job-searching
- 부하테스트 30회, 고도회 4회 진행

팀 통합 테스트 환경 구축

팀원 4인의 테스트 환경 분석 및 불편 사항 수집

- 요청 중 70%에서 TCP 연결 시간 초과 오류가 발생, 코드 외적 오류로 비즈니스 로직 개선에 집중할 수 없는 환경
- 오류 로그 수집 환경이 구성되지 않아, 한 번의 테스트 당 발생하는 5만 줄의 에러로그를 vim으로 디버깅

테스팅 환경 조성

- NIO connector 부터, OS TCP 백로그까지 코드 분석 후, TCP 요청 대기 큐 사이즈를 커널에서 확장
- DBCP 유휴 커넥션 수를 최대 커넥션 수까지 높여서 JDBC 연결 시간 초과 오류 감축
- Cloud Watch를 활용해, 간단한 쿼리로 오류를 수집할 수 있는 환경을 조성
- 평균 3시간 소요되는 오류 파악 시간 40분으로 단축, 팀원들이 비즈니스 로직에 집중할 수 있게하여 전체 성능 향상

소일거리 대행자 NO - SHOW 시 예약 철회 시스템

실 서비스 환경을 재현한 부하테스트 실시

- 실서비스와 유사한 환경을 재현하여, 숨겨진 병목 지점과 오류를 확인하고자 함.
- 일거리 생성부터 의뢰, 대행비 지급까지의 전체 서비스 주기를 Jmeter 코드로 시나리오 구성 및 테스트 실시
- 가상 유저 3000명을 서버와 미리 연결시키고 30초마다 300 RPS 씩 늘리며 계단식 부하 실시
  한 번의 요청 당 위의 모든 생애주기 순차 실행

문제 상황

- 첫 300 RPS 부하에서부터 오류율 25.74% 발생
- NO-SHOW 예약 철회 시스템에서 모든 쓰레드 및 대기큐가 가득차 비동기 요청 거부 예외가 발생.

요청 재시도 전략 구축

- 지수 백오프 전략을 구현하여 처리 거부 당한 요청이 지수 시간 후 실행을 재시도 하도록 조치
- 위 방법은 트래픽을 분산하는 게 아니라 뒤로 미루는 것에 불가해 오히려 성능이 저하됨
- 지연 변이를 적용한 지수 백오프 전략으로 시스템 재구축
- 초당 요청 수 300 에서 오류율 25.74%가 나던 서버를 3000 RPS에서도 오류율 0%로 개선

증명 자료

- 통합 테스트 보고서: https://github.com/6-SPOT/spot-be/wiki
- 시나리오 테스트 16회, 10회 고도화 진행

시니어를 위한 러닝앱 서버 개발 2024.02 - 2024.04

'런데이'와 같은 러닝 앱, 써드 파티 앱에서 사용자의 운동 정보 대규모 수집 및 동기화 시스템 구축
운동 중인 시니어의 소통을 돕기위한 말로하는 채팅 서비스 개발 (음성 → 채팅 기록(STT), 채팅 → 음성 변환 (TTS))

Java, Spring Batch, AWS Lambda, AWS Transcribe, S3, STOMP Socket, HTTP polling

운동 데이터 동기화 시스템 구축

써드 파티앱에서 운동 데이터를 수집하여 서버에 통합 동기화, 실패 지점 대응을 통한 개발자 수동 개입 최소화가 관건

문제상황

- 인프라 팀원의 시행 착오로 서버가 비정상 종료될 때마다 스케줄러가 데이터 수집을 처음부터 다시 실행
  - 이로 인해, 중복된 데이터 삽입 → 관련 API 마비
- 장애 대응 매커니즘의 부재로 스케줄링 도중 작업 실패 시 원인 파악이 어렵고 해결에 오래 걸림 (평균 3시간)
- 동기적인 스케줄러로 구현하여, 스케줄링 실행 시 타 API에 성능 저하 발생

일간 운동 데이터 수집 배치 작업 설계

- Job, Step, Processor 등 Spring Batch 컴포넌트를 활용해 작업 계층을 나눔.
- JpaPagingItemReader를 통해 chunk 단위로 회원 데이터를 메모리에 적재 후 작업 → I/O bound 비용 절감
- 6개의 예외별 트랜잭션 처리를 달리하여, 하나의 작업 실패가 전체 배치에 영향을 주지 않도록 개선
- Job Instance로 실패 지점 특정 및 재실행 시 해당 부분부터 재실행하도록 시스템 구축
- Job 메타테이블을 활용해, 디버깅 시간 단축 (평균 3시간에서 40분으로 단축)
- 데이터 중복 삽입 오류율을 0%로 개선

러닝 중에 '말로 하는' 채팅 서비스 개발

비동기 아키텍처 설계로 채팅 서비스 안정성 강화 및 사용자 경험 개선

문제상황

- 음성 파일 변환 요청이 많아질수록 STT 변환 API의 처리 속도 저하 →전체 80%의 요청에서 시간 초과 예외 발생
- 같은 회원이 여러 번 음성 파일 변환 요청을 하면, 이전 작업 이전에 새 작업이 먼저 완료되어 덮어쓰기되는 오류 발생

비동기 아키텍처 설계 및 구현

- 먼저 음성 파일 원본을 채팅방에 전송한 뒤, 음성 파일의 텍스트 변환이 끝나면 교체하여 사용자 경험 개선
- 서버와 STT 변환 서버 둘로 나누어졌던 아키텍처를 S3와 AWS Lambda 기반으로 재설계
  - 음성 파일 요청 시, S3에 회원 ID + TIME STAMP로 적재
  - AWS Lambda 자동 트리거를 활용해 Aws Transcirbe에 변환을 요청하고 끝나면 S3에 완성된 json 파일을 적재하도록 이벤트 기반 설계
  - 서버는 최초 요청 후 HTTP polling을 통해 s3에 텍스트로 변환된 음성파일이 있는지 확인, 있으면 해당 텍스트로 음성 파일을 교체
- 요청 시간 초과 오류, 덮어쓰기 오류 등 100% 해결,
- S3가 요청 대기 역할을 함으로써, STT 변환 API 성능 향상 (평균 반환 속도 20초 → 7초 개선)

증명자료

- 프로젝트: https://github.com/dalcheonroadhead/walk-walk

채팅 서비스 개발 2024.02 - 2024.04

이미지, 동영상, 음성 파일의 실시간 전송이 가능한 채팅 서비스 프론트 엔드 - 백엔드 전체 구현

Java, Spring, STOMP socket, Base 64, AWS S3, react, vite

채팅 서비스 아키텍처 통합을 통한 개발 효율성 및 사용자 경험 개선

문제상황

- 기존 채팅 서비스는 텍스트는 Socket으로 실시간 전송하면서도, 멀티미디어 전송은 REST API로 구현하여 실시간을 흉내만 내고 있는 이원화 구조
- 코드 복잡성이 증가, 유지 보수 범위 확대, 멀티미디어 전송 속도 저하 (평균 1200ms) 문제 발생

텍스트와 이미지 전송 모두 STOMP socket으로 통합

- STOMP socket의 "텍스트 전용" 제약을 Base64 인코딩/디코딩으로 극복
- 개인 프로젝트에서 미리 검증 후 chunking(65536자 단위 분할) 매커니즘 구현하여 대용량 파일도 전송 지원
- 이미지 업로드 시간 50% 단축 (1200ms → 603ms)
- 코드 복잡성 및 유지 보수 범위 축소로 개발 편의성 높임

증명 자료

- 프로젝트: https://github.com/dalcheonroadhead/kkakka
- 검증용 토이프로젝트:
  - 서버: https://github.com/dalcheonroadhead/real-time-chat-server
  - 프론트: https://github.com/dalcheonroadhead/real-time-chat-client

교육 이수

프로펙트
(2025.02 ~ 2025.04)
(주) goorm 주관
- Back-end 개발자 단기 심화 과정
- 성능 테스트 툴 활용법 학습
- 지표에 따른 성능 개선 역량 구축

삼성 소프트웨어 아카데미 (SSAFY)
(2023.07 ~ 2024.06)
(주) 삼성 전자 주관
- 전공자 백엔드 개발자 과정 우수 수료 (상위 30%)
- 4회 프로젝트, 4회 수상
- 이 달의 멤버 2회 선정

홍익대학교 컴퓨터 공학과 학사
(2017.03 ~ 2023.08)
- SW / 데이터 활용 역량 인증 취득

대외 활동

구름톤
2024.10 ~ 2024.11
주최: kakao x groom
https://github.com/dalcheonroadhead/groomthon-11th-server

개요: 백엔드 개발자 5인 중 한 명으로 선발 및 3박 4일 과정 수료
역량:
- 인프라 역량 증진 필요성을 느끼고 참여
- 크램폴린과 kakao 내부 Docker 서버를 활용해 서버 배포 경험
- Nat-gateway와 private subnet 이론과 활용법 학습

새싹톤
2024.8 ~ 2024.9
주최: 서울 경제 진흥원 x 구글
https://github.com/sesac-dev/eazione-back

개요: 50팀 중 6등으로 본선 진출 및 2주 과정 수료
역량:
- 기한 내 MVP 개발 역량을 쌓기 위해 참여
- 처음 조우한 디자이너 및 PM과 협업 역량 증진

신한 해커톤
2023.9 ~ 2023.9
주최: 서울 경제 진흥원 x 구글
https://github.com/dalcheonroadhead/solsol-pokect

개요: 100팀 중 25팀에 선발 및 1박 2일 연수원 본선 과정 수료
역량:
- 신한은행 내부 계좌 이체 API 활용을 경험할 수 있어 참여
- 현직자 멘토링을 통해, 현직 수준의 성능 테스트와 개발 역량을 확인

수상 이력

- 삼성 소프트웨어 아카데미 자율 프로젝트 우수상 (2024.05.20)
- 삼성 소프트웨어 아카데미 특화 프로젝트 우수상 (2024.04.05)
- 삼성 소프트웨어 아카데미 공통 프로젝트 우수상(2024.02.16)
- 삼성 소프트웨어 아카데미 1학기 관통 프로젝트 최우수상 (2023.11.24)