박준식

qkrwns1478@gmail.com
010-4151-3797

https://github.com/qkrwns1478 (github)
https://munsik22.tistory.com/ (블로그)
https://qkrwns1478.vercel.app/ (포트폴리오)

익숙하지 않던 React와 Node.js를 처음 접했지만, 6주 만에 수백만 건의 데이터를 실시간 분석하는 대시보드를 구축하며
프론트부터 백엔드까지 전반을 주도했습니다.
크래프톤 정글 교육과정에서는 PintOS, Red Black Tree, malloc 등 난이도 높은 과제를 반복적으로 해결하며
시스템 구조에 대한 이해와 문제 해결력을 키웠습니다.
낯선 개념에 좌절하기보다는 동료들과 함께 구조를 분석하고 토론하며 끝까지 완수했고,
어떤 기술 환경에서도 빠르게 적응할 수 있다는 자신감을 갖게 되었습니다.
기능 구현을 넘어 데이터 흐름, API 설계, 사용자 경험까지 함께 고민할 수 있는 풀스택 개발자로 성장하고 있습니다.

[실전과 학습을 병행하며 꾸준히 성장하는 개발자입니다]

- 정글 교육과정 동안 프로젝트를 수행하면서 C++ 및 알고리즘 문제 풀이 스터디를 자발적으로 병행했습니다.
- 하루 한 문제 알고리즘 풀이, 블로그 정리를 통해 실전과 이론 사이의 간극을 줄이고자 꾸준히 반복 학습을 이어가고 있습니다.
- 프로젝트가 끝난 이후에도 기술 블로그 정리를 계속하며, 장기적인 성장 습관을 유지하는 개발자로 성장하고자 노력 중입니다.

[문제를 구조적으로 바라보고, 성능과 안정성을 함께 고려합니다]

- KlickLab 프로젝트에서 API 응답 속도가 10.67초까지 지연되던 병목 문제를 발견했습니다.
- 단순 튜닝에 그치지 않고 집계 테이블과 MV를 설계하고 기존 쿼리를 집계된 데이터 기반으로 교체하여 응답 속도를 개선했습니다.
- 1000만 건 기준 응답 시간을 60ms로 단축(99.43% 개선)하는 성능 개선을 이끌었습니다.

학력 및 병역

2019.03 ~ 2025.08 경북대학교 전자공학부 학사 졸업예정
2020.12 ~ 2022.06 카투사 병장 만기 전역

기술 스택

Languages : C, Python, JavaScript (TypeScript)
Frontend : React (Vite), Flask
Backend : Node.js (Express), PHP
Database : ClickHouse, PostgreSQL, MySQL, MongoDB

주요 경험

● 데이터 분석 플랫폼 KlickLab (2025.06 - 2025.07) (깃허브 레포지토리) (소개 영상)

클릭스트림 기반 실시간 데이터 수집 및 분석 플랫폼 개발 – 이벤트 수집 SDK, 쿼리 병목 개선, 고가용 시스템 설계

▶ 프로젝트 개요

- 사용자가 웹사이트에서 클릭, 페이지 이동, 폼 입력 등 행동을 수행할 때마다 이를 수집하고
  분석 가능한 형태로 가공하여 시각화해주는 데이터 분석 플랫폼
- 웹사이트 운영자들이 '전환율', '유입 경로', '이탈 지점' 등을 빠르게 파악할 수 있도록 대시보드를 제공하며,
  수집 → 분석 → 시각화 전 과정을 직접 설계 및 구현했습니다.

▶ 주요 기능과 문제 해결 흐름

1. SDK 이벤트 수집 제외 기능

- SDK가 페이지 내의 모든 클릭 이벤트를 수집하는 구조였기 때문에,
  사용자 행동과 무관한 의미 없는 클릭 이벤트(예: 빈 영역 클릭, 그림자/오버레이 클릭 등)까지
  필터링 없이 마구잡이로 수집되는 문제가 발생했습니다.
- 의미 없는 요소에 대해 이벤트 수집을 차단할 수 있도록,
  kl-disabled 클래스를 포함한 DOM 요소 또는 부모 요소에서 발생한 이벤트는 수집하지 않도록 SDK 이벤트 리스너를 수정했습니다.
  이벤트 타겟이 없는 경우를 대비한 방어 로직과,
  상위 요소 하나에만 클래스를 부여해도 전체 하위 요소 이벤트가 차단되도록 DOM 트리 탐색 구조를 도입했습니다.
- 시각적·기능적으로 의미 없는 클릭이 수집되지 않게 되어 지표의 정확도와 정합성이 향상되었고,
  클릭 우선순위, 전환 분석 등에서 실제 사용자 행동만 기반으로 한 분석이 가능해졌습니다.

2. 대시보드 쿼리 병목 개선 – 집계 테이블 및 Materialized View 구조 설계

- 프로젝트 초기에는 모든 대시보드 지표(전환율, 이탈률, 유입 분석 등)를 ClickHouse의 events 원본 테이블에서 직접 조회하는 구조였습니다.
  특히 GROUP BY, JOIN, WHERE 조건이 복합적으로 걸리는 쿼리에서 응답 지연 및 타임아웃 현상이 발생했고,
  1,000만 건 이상의 데이터에서 응답 시간이 10초 이상 소요되는 병목이 발생했습니다.
- 대시보드에서 자주 사용하는 지표에 대해 일/시/분 단위의 집계 테이블(flat_, agg_)을 별도로 생성하고,
  ReplacingMergeTree, AggregatingMergeTree 등 테이블 특성에 맞는 엔진을 선택해 쿼리 최적화와 TTL 관리까지 고려한 설계를 진행했습니다.
  ClickHouse의 Materialized View(MV)를 통해 이벤트가 적재될 때마다 자동으로 집계 테이블로 가공되도록 처리했습니다.
  백엔드 API는 모두 MV 대상 집계 테이블만 조회하도록 쿼리 구조를 변경했습니다.
- 주요 쿼리의 응답 시간이 최대 10.67초 → 최소 60ms로 단축(약 99.43% 개선)되었고,
  분석 중 타임아웃 문제 없이 대시보드의 실시간 인터랙션 가능 수준까지 도달했습니다.
  API 측에서도 분석 대상 테이블을 원본에서 집계 테이블로 전환함으로써 CPU 사용량과 I/O 부담이 감소했고,
  데이터 증가량이 수천만 건으로 늘어났음에도 성능 저하 없이 분석 기능이 유지되었습니다.

3. 읽기/쓰기 분리 구조 적용 – ClickHouse 기반 CQRS 설계

- SDK에서 수집한 클릭 이벤트가 실시간으로 events 테이블에 INSERT되는 동시에,
  대시보드 API에서 수많은 SELECT 쿼리가 실행되며 같은 ClickHouse 노드에 읽기/쓰기 부하가 집중되면서
  일시적으로 메모리 부족(OOM) 현상이 발생하였고 데이터 수집 실패, 서버 중단, ClickHouse 노드 장애까지 이어졌습니다.
- 읽기/쓰기 분리를 위한 CQRS 구조를 ClickHouse 레벨에서 직접 설계했습니다.
  - A 노드: Kafka를 통해 수집된 로그를 INSERT하는 쓰기 전용 노드(Producer)
  - B 노드: 집계 테이블(flat_, agg_)만 조회하는 읽기 전용 노드(Consumer)
- 이벤트 수집과 분석 요청이 완전히 물리적으로 분리되면서 OOM 현상이 재발하지 않았고,
  ClickHouse 장애나 로그 수집 중단 없이 수집과 분석이 안정적으로 병행 가능해졌습니다.
  실시간 분석 요청이 많은 상황에서도 대시보드 응답 속도와 시스템 안정성이 유지되었습니다.

4. RPS 급락 문제 해결 – Node.js GC 분산 구조 전환

- 초기에는 c6in.xlarge 고사양 인스턴스 17대를 구성해 로그 수집 서버를 운영했지만,
  Node.js의 단일 스레드 구조와 주기적인 Garbage Collection(GC)으로 인해 약 30초 주기로 Stop-the-World 현상이 발생했고,
  RPS가 순간적으로 급락하며 일부 요청이 504 Gateway Timeout으로 실패하는 문제가 지속되었습니다.
- 고사양 인스턴스를 추가할수록 비용이 급격히 상승하고, vCPU 총량 할당 한계에도 부딪혀 수직 확장 전략이 더 이상 유효하지 않았습니다.
- vCPU 1개짜리 t2.small 인스턴스 74대를 활용해 수평 확장 구조로 전환했습니다.
  각 인스턴스는 독립적인 Node.js 프로세스를 실행하며, GC는 개별 인스턴스 수준에서만 발생하므로
  Stop-the-World의 영향이 전체 트래픽에 파급되지 않도록 구조적으로 분산시켰습니다.
- GC에 의한 RPS 급락 현상이 사라졌고, 평균 RPS는 기존 3,938에서 10,682로 약 2.7배 증가했습니다.
  동시에 인프라 비용은 약 56% 절감되었으며, 더 이상 수집 중단이나 타임아웃 없이 안정적인 고빈도 로그 수집이 가능해졌습니다.

5. **전환 이벤트 정의 구조 개선 – 사용자 목적에 맞는 전환 지표 분석 지원**

- 기존 전환 이벤트 로직은 event_name IN ('purchase', 'signup', 'conversion')처럼
  쇼핑몰 운영자 관점에서 정해진 전환 이벤트 목록이 하드코딩된 구조였습니다.
  하지만 사용자 맞춤 전환 조건을 반영할 수 없어 대시보드에서 제공하는 전환율 지표가 사용자의 실제 관심사와 불일치하는 문제가 발생했습니다.
- 전환 정의 방식을 고정된 목록이 아닌 사용자 정의 기반으로 전환했습니다.
  사용자는 클릭랩 대시보드에서 특정 이벤트명, 페이지 경로, 버튼 요소 등 원하는 조건을 직접 등록할 수 있도록 개선했고,
  SDK는 수집 시점에 이 설정값을 기반으로 해당 이벤트가 전환 조건에 해당하는지를 판별하게 했습니다.
- 사용자는 이제 자신의 서비스 목적에 따라 다양한 행동을 전환으로 자유롭게 지정할 수 있게 되었고,
  대시보드에서 제공하는 전환율과 퍼널 분석 지표가 실제 관심 있는 지점에 정확히 대응하게 되었습니다.

6. 렌더링 최적화를 위한 API 요청 구조 개선

- 대시보드 내 일부 컴포넌트가 실제로는 화면에 표시되지 않음에도 불구하고 관련 API가 모두 호출되어
  불필요한 네트워크 요청이 다수 발생했고 전체 렌더링 시간이 길어지며 사용자 경험 저하로 이어졌습니다.
- 조건부 렌더링 로직을 적용해 보이지 않는 컴포넌트의 API 요청을 차단하고,
  요청 병렬화 및 응답 캐싱 로직을 추가해 중복 요청을 제거했습니다.
- 초기 렌더링 시점의 API 요청 수가 감소하면서 로딩 시간이 단축되었고, 클라이언트 측 부하와 서버 요청량이 모두 감소했습니다.
  반복되는 지표 조회에서도 불필요한 네트워크 요청 없이 지속적으로 빠른 응답을 유지할 수 있는 기반을 마련했습니다.

● **크래프톤 정글 8기 교육과정** (2025.03 – 2025.07)
운영체제 및 시스템 프로그래밍 기반 실습 – KAIST PintOS

- **Project 1 – Thread**
  라운드로빈 스케줄러 구현 및 priority donation 기능을 도입하여 커널 수준의 동시성 문제를 해결하는 구조를 설계했습니다.
- **Project 2 – System Call**
  시스템 콜을 직접 구현하면서 프로세스 생성과 파일 I/O 처리에서 사용자 공간과 커널 공간 간 인터페이스 작동 방식을 학습했습니다.
- **Project 3 – Virtual Memory**
  Lazy loading, Demand paging, Stack growth를 통해 프로세스 메모리 요청 시점에 물리 프레임을 동적으로 할당하는 구조를 구현하고
  실제 페이지 폴트 상황에서의 처리 흐름을 코드 레벨로 디버깅했습니다.

▶ 수만 라인 규모의 베이스 코드를 해석하고, 커널 내부 데이터 구조와 실행 흐름을 깊이 있게 이해하며 시스템 수준 디버깅 능력을 향상시켰습니다.
▶ 코드 리뷰와 팀 내 토론을 통해, 단순한 기능 구현을 넘어 “왜 이 구조로 구현해야 하는가”를 설명하고 설득하는 협업 방식에 익숙해졌습니다.

컴퓨터공학 기초 역량 강화

- C언어 기반 자료구조 및 시스템 구현
  Red-Black Tree 구현, next-fit malloc 기반 사용자 메모리 할당기 작성, 소켓 통신 기반 웹서버 및 프록시 서버 개발 등을 통해
  포인터, 메모리 모델, 시스템 호출 흐름을 체계적으로 익혔습니다
- Python 알고리즘 풀이 및 문제 해결 능력 강화
  DFS, BFS, DP, 우선순위 큐 등 핵심 알고리즘을 실전 환경에서 반복 학습하며, 풀이 시간 단축과 코드 최적화를 함께 훈련했습니다.

▶ 알고리즘/시스템 구현 팀 활동 및 스터디를 통해 서로의 접근 방식을 비교하고, 설계 이유를 설명하며
문제 해결 방식 자체를 개선해 나가는 공동학습 경험을 쌓았습니다.
▶ 단순 구현이 아닌 왜 이 방식이어야 하는지 설명할 수 있는 코드를 목표로 학습하였고,
이를 통해 협업 기반 개발 문화에 필요한 커뮤니케이션 역량과 사고 정제 능력을 길렀습니다.

● 키오스크 버전 관리 웹 애플리케이션 (2024.02)

키오스크 앱 배포 및 버전 현황 관리를 위한 운영자용 백오피스 시스템

1. 대용량 APK 업로드 실패 문제

- 네트워크가 불안정한 환경에서 .apk 파일 업로드가 중단되면 전체 파일이 유실되거나 다시 처음부터 업로드해야 하는 상황이 반복되었습니다.
- 분할 업로드 및 서버 병합 구조를 도입하여, 클라이언트에서 .apk 파일을 여러 조각으로 나눠 전송하고
  서버에서는 이를 순차적으로 병합하도록 구현했습니다.
  각 조각의 업로드 상태를 저장하고, 중단 지점부터 이어서 업로드할 수 있도록 재요청 가능 구조로 설계했습니다.
- 대용량 파일도 안정적으로 업로드할 수 있게 되었으며, 네트워크가 끊기더라도 부분 복구가 가능해 사용자 경험과 안정성이 향상되었습니다.

2. 버전 정보 및 메타데이터 통합 관리 기능

- 각 지점 키오스크의 설치 버전 정보를 관리할 방법이 없어 APK 파일을 언제 누가 어떤 버전으로 업로드했는지 파악할 수 없었고,
  잘못된 버전이 배포되어도 이력을 추적하기 어려운 상태였습니다.
- APK 파일 등록 시 자동으로 생성되는 버전 코드, 파일명, 용량, 등록 일시 등을 버전 관리 테이블에 함께 저장하도록 설계하고,
  각 단말기별 현재 설치 버전과 연결할 수 있도록 구조화했습니다.
  운영자가 등록된 버전 목록을 UI 상에서 테이블 형태로 쉽게 확인할 수 있도록 구성했습니다.
- 운영자는 이제 각 단말기 또는 전체 사업장의 버전 업로드 이력과 상태를 한눈에 확인할 수 있게 되었고,
  잘못된 버전 배포나 누락을 사전에 방지할 수 있게 되었습니다.

▶ 백엔드와 프론트엔드를 모두 구현하며, 실사용자의 관점에서 '어디서 막힐 수 있는가'를 중심으로 UI/로직을 설계하는 감각을 키울 수 있었습니다.
▶ 기능 구현을 넘어서, 운영자 입장에서 시스템을 바라보며 사용자 중심 설계의 중요성을 체감했습니다.