구글 맵 (240114)

🔥 목적지와 경로를 찾을 수 있는 서비스를 설계하자

▼ 1️⃣ 설계 범위 확정

- DAU 10억
- 위치 갱신, 경로 안내, ETA, 지도 표시에 초점
- 도로 데이터는 수 TB 수준의 가공되지 않은 데이터
- 교통 상황 고려
- 운전, 대중교통 등 다양한 이동 방법 지원
- 경유지 고려하지 않음
- 사업장 위치 및 사진 표시하지 않음

▼ 2️⃣ 요구사항 정리

주 단말은 스마트폰

기능 요구사항

- 사용자 위치 갱신
- 경로 안내 서비스
- 지도 표시

비기능 요구사항

- 정확도
- 부드러운 경로 표시
- 데이터 및 배터리 사용량
- 가용성
- 규모 확장성

▼ 3️⃣ 기본 개념 짚고가기

측위 시스템

구 표면 상의 위치를 표현하는 체계

- 위도: 주어진 위치가 얼마나 북/남쪽인가
- 경도: 주어진 위치가 얼마나 동/서쪽인가

지도 투영법 / 도법

3차원 구의 위치를 2차원 평면에 대응시키는 절차

- 메르카토르 도법 (구글 Pick!)
- 퍼스 퀸쿤셜 도법
- 갈-페터스 도법
- 윈켈 트리펠 도법

지오코딩

주소를 지리적 측위 시스템의 좌표로 변환하는 프로세스

- 인터폴레이션 : GIS와 같은 다양한 시스템이 제공하는 데이터를 결합
- GIS : 도로망을 지리적 좌표 공간에 대응시키는 방법을 제공하는 시스템

지오해싱

지도 위 특정 영역을 짧은 문자열에 대응시키는 인코딩 체계

- 2차원의 평면 공간으로 표현된 지리적 영역 위 격자를 더 작은 격자로 재귀적 분할
- 맵 타일 관리에 적용 예정

지도 표시

- **타일**: 지도를 화면에 표시하는 데 가장 기본이 되는 개념
  - 지도 확대/축소를 지원하기 위해 여러 종류의 타일을 준비해야 함

경로 안내를 위한 도로 데이터 처리

모든 경로 탐색 알고리즘은 교차로를 노드, 도로는 선으로 표현하는 그래프 자료 구조를 가정한다.

경로 탐색 알고리즘의 성능은 그래프 크기에 좌우되므로 그래프를 관리 가능 단위로 분할해야 한다.

따라서 …

1. 지오해싱과 같은 분할 기술로 세계를 작은 격자로 나누고
2. 격자 안의 도로망을 노드와 선으로 구성된 그래프 자료 구조로 변환한다.
3. 이 때 각 격자는 `경로 안내 타일`이라 하고 각 타일은 도로로 연결된 다른 타일에 대한 참조를 유지한다.

> 🤦🏻 **주의 !**
>
> 경로 안내 타일은 지도 표시에 이용되는 타일과 유사해보인다.
> 그러나
> **지도 타일은 PNG** 이미지, **경로 안내 타일은 이진 파일**이다.

계층적 경로 안내 타일

효과적인 경로 안내를 위해

- 구체성 정도를 상, 중, 하로 구분하여 3가지 종류의 경로 안내 타일을 준비한다.
  - 구체성이 높을수록 크기는 작다.
  - 구체성이 낮을수록 큰 도로 데이터를 둔다.
- 각 타일에는 다른 정밀도 타일로 연결되는 선이 있을 수 있다.

▼ 4️⃣ 개략적 설계

▼ 규모 추정

저장소 사용량

세계 지도

- 지도 확대마다 1 타일을 4 타일로 펼친다고 가정
- 최대 확대 수준: 4.4조 개 타일 필요
- 타일 1장 : 256 * 256픽셀 압축 PNG 파일(100KB)

⇒ 최대 확대 시 4.4조 * 100KB = `440PB`

단, 지구 표면 90%는 인간이 살지 않는 영역이므로 10% 수준으로 줄어듦.

대략 100PB로 가정했다.

메타데이터

각 지도 타일의 메타데이터는 아주 작기 때문에 추정하지 않는다.

도로 정보

수 TB

서버 대역폭

1. **경로 안내 요청**
   - DAU 10억, 기능 사용시간 평균 주당 35분이라고 가정
     - 사용 시간 환산 시 주당 350억분, **하루 50억분**

- GPS 좌표 매초 전송 시 하루 3000억 건 요청 발생 ⇒ 300만 QPS
- 15초마다 전송 시 **QPS 20만**
- **최대 QPS** = 20만 * 5 = **100만**

2. **위치 갱신 요청**

   상세 설계에서 다룹니다 …

위치 서비스

> 💡 사용자의 위치 기록. 클라이언트가 t초마다 자기 위치를 전송한다고 가정

Why?

1. 데이터 스트림을 활용하여 시스템 개선
   a. 실시간 교통 상황 모니터링
   b. 새로 만들어진 도로나 폐쇄된 도로 탐지 가능
   c. 사용자 행동 패턴을 분석하여 개인화 경험 제공
2. ETA의 정확한 산출과 교통 상황에 따라 다른 경로 안내

⇒ 😡 **15초마다 위치를 보내도록 해도 여전히 높은 쓰기 부하 발생 !**

- 높은 쓰기 요청 빈도에 최적화, 규모 확장 용이한 카산드라 사용
- 카프카와 같은 스트림 처리 엔진 활용하여 위치 데이터 로깅
- HTTP `with` `keep-alive` 사용하여 효율 높임

경로 안내 서비스

> 💡 A → B로 가는 합리적이고 빠른 경로 탐색. 높은 정확도 보장

GET /v1/nav?origin=1355+market+street,SF&destination=Disneyland

거리, 소요시간, geocoded waypoints 등 반환

상세 설계에서 경로 재탐색, 교통 상황 변화 문제를 고려해보자.

지도 표시

💡 확대 수준별로 모든 지도 타일을 가지고 있기 버거우니까 대안을 찾아보자.

클라이언트가 보는 지도 확대 수준에 따라 즉석에서 필요한 타일 만들기

- 위치 + 확대 수준 조합은 무한함
- 모든 지도 타일을 동적으로 만들어야 하는 서버 클러스터에 심각한 부하 발생
- 캐시 활용 어려움

미리 만들어 둔 지도 타일을 클라이언트에 전달하기

1. 지오해싱과 같은 분할법을 사용해 고정된 사각형 격자로 표현
2. 클라이언트가 지도 타일이 필요한 경우 현재 확대 수준에 근거하여 필요한 집합 결정
3. 각 위치를 지오해시 URL로 변환
4. 미리 만들어 둔 정적 이미지는 CDN을 통해 반환

모바일 데이터 사용량을 줄이기 위해 클라이언트 측 캐시를 고려할 수 있다.

🔎 **데이터 사용량**

사용자는 **30km/h** 속도로 이동 중, 한 이미지가 **200m * 200m** 영역 표현

1km * 1km 영역 표현 시 이미지 25장 필요 (2.5MB)

→ **시간 당** 30 * 2.5MB = `75MB` **데이터 소진**

🔎 **CDN을 통해 서비스되는 트래픽 규모**

매일 50억 분의 경로 안내 처리, 즉 50억 * 1.25MB = 6.25PB/일

→ 초당 전송해야 하는 지도 데이터 양 = `62500MB`

전 세계 200개 POP가 있다고 가정

→ **각 POP 처리 트래픽** = 62500 / 200 = `수백 MB`

지도 표시 흐름

1. 모바일 사용자가 지도 타일 서비스 호출
2. 로드밸런서로 요청 전달
3. 로드밸런서는 해당 요청을 지도 타일 서비스로 전달
4. 지도 타일 서비스는 클라이언트 위치, 확대 수준을 입력으로 9개 타일 URL 계산
5. 클라이언트에 반환
6. 해당 타일을 CDN을 통해 다운로드

▼ 5️⃣ 상세 설계

🗂️ 데이터 모델

(1) 경로 안내 타일

외부에서 받은 데이터, 방대한 양의 도로와 메타데이터로 구성된 그래프 자료 구조 형태 데이터

경로 안내 타일 처리 서비스(오프라인 데이터 가공 파이프라인)를 주기적으로 실행해서 **경로 안내 타일로 변환**해야 한다.

어디에 저장할까?

- 후보 1) 메모리
  - 그래프 데이터는 메모리에 인접 리스트 형태로 보관하는게 일반적
  - 우리 데이터는 양이 너무 많다.
- 후보 2) 데이터베이스
  - 그래프의 노드와 선을 데이터베이스 레코드로 저장
  - 비용이 많이 들 것이다.
  - 경로 안내 타일은 데이터베이스가 제공하는 기능이 필요 없다.
- 후보 3) S3
  - 객체 저장소에 파일을 보관하고 그 파일을 이용할 서비스에서 적극적으로 캐싱
  - **지오해시 기준으로 분류**해두면 위도, 경도가 주어졌을 때 빠르게 찾을 수 있다.

(2) 사용자 위치

엄청난 양의 쓰기 연산 처리, 수평적 규모 확장이 가능해야 함 → **카산드라**

user_id	timestamp	user_mode	driving_mode	location
101	1635740977	active	driving	(20.0, 30.5)

🤔 Why 카산드라?

- 초당 백만 건 업데이트 발생 → 쓰기 연산에 강해야 함
- 사용자 위치는 계속 변화고 이전 정보는 필요없음 → 가용성 > 일관성

→ 가용성(C), 분할 내성(P)을 만족하는 데 집중하여 카산드라를 채택한다.

(3) 지오코딩 데이터

`주소 → 위도/경도 쌍`으로 변환한 정보 보관

빠른 읽기 연산을 제공하는 **키-값 저장소가 적합** (ex: Redis)

출발/목적지 주소는 경로 계획 서비스에 전달되기 전 이 데이터베이스를 통해 위도/경도 쌍으로 변환돼야 함

(4) 미리 계산해 둔 지도 타일 이미지

특정 영역의 지도를 요청할 경우 인근 도로 정보를 취합한 이미지를 만들어내야 함

중복 요청이 많으므로 **이미지 계산은 한 번만, 결과는 캐싱**해두는 전략

확대 수준별로 이미지를 미리 만들어 두고 CDN(원본 서버는 S3와 같은 클라우드 저장소)을 통해 전송

🗺️ 위치 서비스

사용자 위치 데이터

- key: (**user_id**, **timestamp**) / value: **location**
- user_id는 파티션 키(특정 사용자의 최근 위치를 빠르게 읽기 위함), timestamp는 클러스터링 키
    - 같은 파티션 키를 갖는 데이터는 함께 저장, 클러스터링 키 값에 따라 정렬됨

> 🤔 **어떻게 쓰이는가?**
>
> 지도 데이터의 정확성 개선, 실시간 교통 현황 파악, …
>
> 카프카 메시지 큐에 로깅하면 개별 서비스가 용도에 맞게 활용할 수 있다!

🔎 지도 표시

구글의 설계를 참고해서 지도 타일을 미리 만들고 지도 표시를 최적화하자

지도 타일 미리 만들기

확대 수준별 지도 타일을 미리 만든다. **확대 수준 0은 세계를 `256 * 256픽셀` 타일 1개로 표현**한다.

확대 수준이 올라갈 때마다 …

- 전체 타일 수는 두 배씩 늘어난다.

- 해당 수준 전부를 합친 해상도는 이전 대비 네 배씩 늘어난다.

→ **화면에 한 번에 표시 가능한 지도 타일 개수는 달라지지 않으므로** 네트워크 대역폭을 소진하지 않고도 최적의 지도를 표시할 수 있다 !

지도 표시 최적화 기법

WebGL 기술로 지도를 표시하자 !

네트워크를 통해 이미지를 전송하는 대신 경로와 다각형 등의 벡터 정보를 보낸다.

클라이언트는 수신된 경로와 다각형 정보를 통해 지도를 그린다.

장점

- 이미지에 비해 월등한 압축률, 네트워크 대역폭을 아낀다.
- 매끄러운 지도 확대 경험

🧭 경로 안내 서비스

가장 빠른 경로를 안내하자

컴포넌트

- **지오코딩 서비스**: 주소 → 위도, 경도 쌍 변환
- **경로 계획 서비스**: 최적화된 경로 제안
- **최단 경로 서비스**: 출발지, 목적지 위도/경도를 입력 받아 k개 최단 경로 반환
  - 교통, 도로 상황 등 실시간 정보는 고려하지 않음 → 정적임 캐싱 대상!
  - A*(에이스타) 경로 탐색 알고리즘 실행
    - 1) 출발지, 목적지 위도/경도를 지오해시로 변환하여 경로 안내 타일을 얻음
    - 2) 출발지 타일부터 그래프 자료구조 탐색
    - 3) 탐색 범위를 넓히는 과정에서 필요한 주변 타일은 객체 저장소(or 캐시)에서 가져옴
    - 4) 경로가 충분히 확보될 때까지 검색 범위를 계속 확대하며 필요한 만큼 타일을 가져옴
- **예상 도착 시간 서비스**: 기계 학습을 활용해 현재 교통 상황, 과거 이력에 근거하여 예상 도착 시간을 계산
  - 앞으로 10분, 20분 뒤 어떻게 달라질지도 예측해야 함 알고리즘 차원에서 풀어야 한다네요

- **순위 결정 서비스**: 사용자가 정의한 필터링 조건(유로 도로 제외, 고속도로 제외 등) 적용
  - 필터링 이후 경로를 소요 시간 순으로 정렬하여 최단 시간 경로 k개 반환
- **중요 정보 갱신 서비스들**
  - 카프카 위치 데이터 스트림을 구독하고 있다가 중요 데이터를 비동기적으로 업데이트
    - **실시간 교통 정보 데이터베이스**: 예상 도착 시간 서비스 정확도 상승
    - **경로 안내 타일**: 새로 발견된 도로 or 폐쇄된 도로
- **적응형 ETA와 경로 변경**
  - 현재 경로 안내를 받고 있는 모든 사용자를 추적해야 함
  - 교통 상황이 달라질 때마다 각 사용자의 ETA를 변경해야 함

> 🤔 **그렇다면?**
>
> 현재 경로 안내를 받고 있는 사용자를 어떻게 추적할까?
>
> 교통 상황 변화에 영향을 받는 경로와 사용자를 어떻게 효율적으로 가려내는가?
>
> 1) 어떤 타일에 대해 사용자마다 안내 받은 경로 정보 레코드를 전수 조사해서 찾는다.
>
> 2) 각각의 경로 안내 타일, 그 타일을 포함하는 상위 타일, 그 상위, 그 상위, … (출발지와 목적지가 한 타일에 있을 때까지) 재귀적으로 더하여 보관한다.
>
> - 어떤 타일에 대해 해당 사용자의 데이터베이스 레코드 마지막 타일에 그 타일이 속하면 ETA가 달라지는 사용자다.
>
> 3) 경로 안내를 받는 사용자가 이용 가능한 경로의 ETA를 주기적으로 재계산한다.

- **전송 프로토콜**
  - 경로 안내 중 변경된 경로 상황을 모바일 클라이언트에 전송
  - **모바일 푸시 알림**: 메시지 크기가 제한적, 웹앱 지원 X
  - **웹소켓**: 서버 부담이 크지 않아 롱 폴링보다 좋은 방안임
  - 서버 전송 이벤트

▼ 6️⃣ 마무리

위치 갱신하기

- 엄청난 양의 쓰기 연산 ~ 카산드라
- 파티션 키, 클러스터링 키
- 카프카에도 로깅하기

경로 계획하기

- 출발지/목적지 → 위도, 경도 쌍으로 변환해서 `최단 경로`, `ETA`, `순위 결정` 서비스 호출하기
- 적응형 ETA
- 경로 변경
- 어떻게 알릴 것인가 ~

지도 표시하기

- 확대 수준별로 타일 만들기
- 캐싱 전략, CDN
- WebGL로 최적화

ETA 계산하기

- 기계 학습
- 현재 교통 상황
- 과거 이력

추가로…

`중간 경유지 설정하기` 같은 것도 얘기해 볼 수 있겠다~

토론 주제

사용자 위치 데이터가 지도 데이터 개선(신규/폐쇄 도로 확인)에 쓰인다고 했다. 어떤 기준으로 신규/폐쇄 도로인지를 확인할 수 있을까?

설계에 앞서 도로 데이터는 외부에서 정기적으로 받아온다고 했는데 이걸로 지도 데이터 개선을 하는 게 더 효율적이지 않을까?

- 새로 만들어진 도로나 폐쇄된 도로 데이터를 확인하기 위해 도로가 아닌 타일에 일정량 이상의 사용자가 위치하는지, 도로인 타일에 어느 기간동안 사용자가 위치하지 않는지 등등.. 을 추적해야할 것 같은데 거기에 쓰는 리소스보다 외부 데이터를 받아오는 게 더 효율적이지 않을까?
- 만약 둘 다 사용하면 이 두 가지가 충돌할텐데? 그건 어떻게 해결할 수 있을까?
    - 외부 데이터 갱신 때 다시 엎어치고
    - 사용자 데이터 분석하면서 다시 또 엎어치고 .. 무한 반복