# Google Maps

System Design Interview Vol 2

Alex Xu & Sahn Lam

# Step 1: Understand the Problem and Establish Design Scope

- How many daily active users?
  - 1 billion DAU
- Which features should be focused here?
  - Location update
  - Navigation
  - Estimated Time of Arrival (ETA)
  - Map rendering
- How large is the road data?
  - TBs of raw data
- How about different travel models, e.g. driving, wlarking, transportation
  - All travel modes supported
- Should it support multi-stop directions?
  - Good to have, but not need to focus right now.
- How about business places and photos?
  - Do not need to design those.

# Step 1: Understand the Problem and Establish Design Scope

**Non-functional requirement and constraints:**

- Accuracy: correctness of the routes and directions to the users
- Smooth navigation: On the client-side, users should have smooth map rendering experience.
- Data and battery usage: The client should use as little data and battery as possible.
- General availability and scalability requirements.

# Map 101

- **Geocoding**

  Address → Latitude, Longitude

- **Geohashing**

  Geohashing encodes a geographic location into a short string of letters and digits.

  - First published by Gustavo Niemeye in 2008, base 32, http://geohash.org

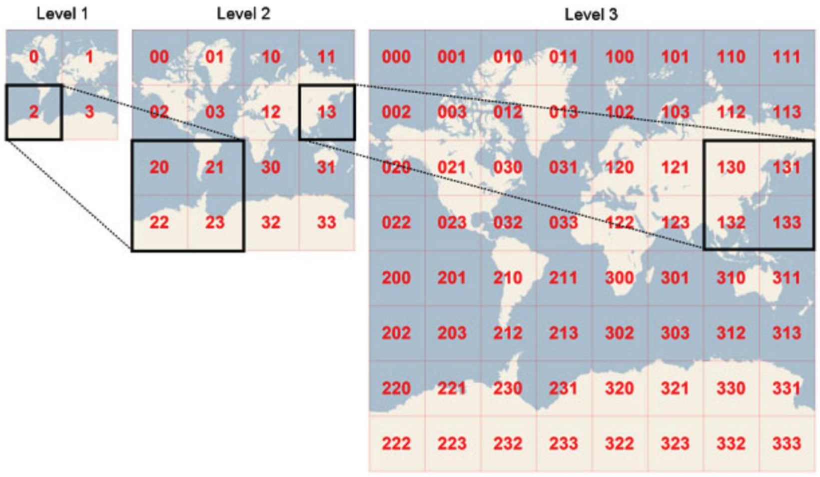
    For example, the coordinate pair 57.64911,10.40744 produces a slightly shorter hash of u4pruydqqvj.

  - Base64 geohash, 2009, 2014
  - Hilbert-Geohash, 2016

  The main usages of Geohashes are:

  - As a unique identifier.
  - To represent point data, e.g. in databases.

# Map 101




## Map Rendering and Road data processing

- Map tile: The world map is broken up into smaller tiles
- Routing tile: For each tile, we convert the road network into grap
  - Intersection -> node
  - Road -> edge
  - Each tile holds references to all the other tiles it connects to.
  - Typically three sets of routing tiles with different zoom level.

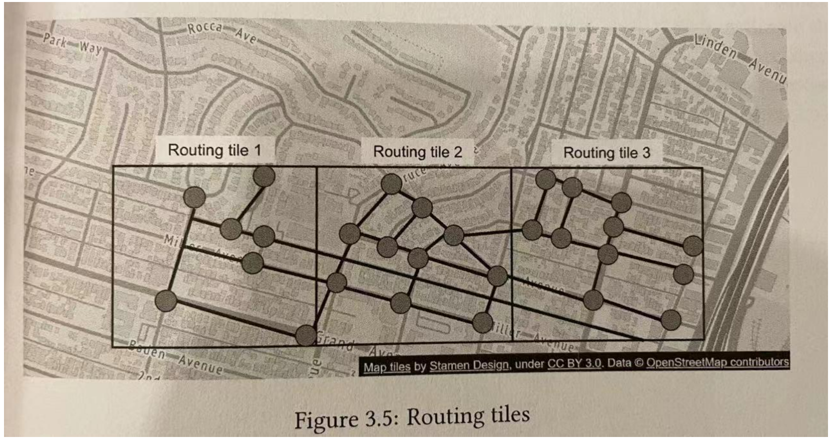


Figure 3.5: Routing tiles

## Navigation algorithms

Dijkstra or A* pathfinding algorithms

# Storage estimate

Map Tile Data: ~ 100 PB

| Zoom Level | Nr. of Tiles | Image total size | Storage needed |
| --- | --- | --- | --- |
| 0 | 1 = 4^0 | 100 KB | 50 / 4^21 |
| 1 | 4 = 4^1 | 4 * 100 KB | 50 / 4^20 |
| 2 | 16=4^2 | | 50 / 4^19 |
| …. | 4^n | | 50 / 4 ^ (21-n) |
| 21 | 4.4 trillion =4^21 | 440 PB | 440 * 90% ~ 50 PB |

~90% of the world surface is un-inhabited area

Road info Data: several TB from external sources

# Server QPS estimate

1 Billion DAU

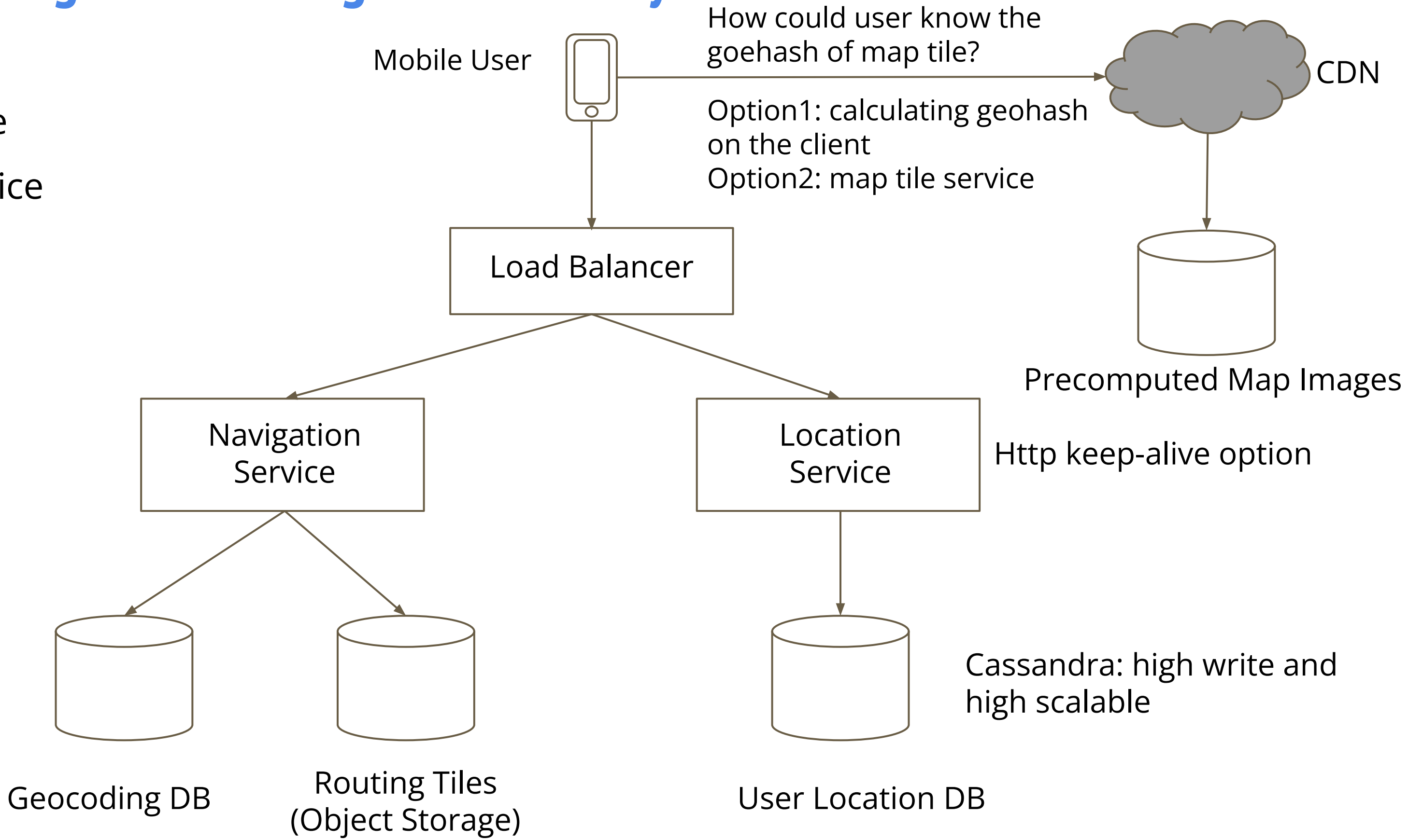
35 min/week per user

5 billion mins / day

| sending GPS coordinate to server | Requests for all DAU | QPS |
|---|---|---|
| 1 request / second | 5 billion * 60 / day | 3 million |
| 1 request / 15 second (in batch) | 5 billion * (60/15) / day | 0.2 million |

# Step 2: Propose High-level Design and Get Buy-in

1. Location service
2. Navigation service
3. Map rendering



# Step 3: Design Deep Dive

**Data Model:**

- Routing tiles

Offline processing pipeline: routing tile processing service.
Input : road info datasets (names, country, longitude, and latitude)
Output: routing tiles (graph data structure)
Storage: object storage like AWS S3, and cache on the routing service.

- User location data

It is user's location data at different timestamps.
It is used for many use cases: update routing tiles, build and update traffic data
It is write-heavy workload, should be horizontally scaled: Cassandra.

| User_id (key) | timestamp | user_mode | driving_mode | location |
|---|---|---|---|---|
| 101 | 1635740977 | active | driving | (20.0, 30.5) |

# Step 3: Design Deep Dive

**Data Model:**

- Geocoding database

  Store address -> geo coordinate (lag/lng pair)
  Key-value database: Redis

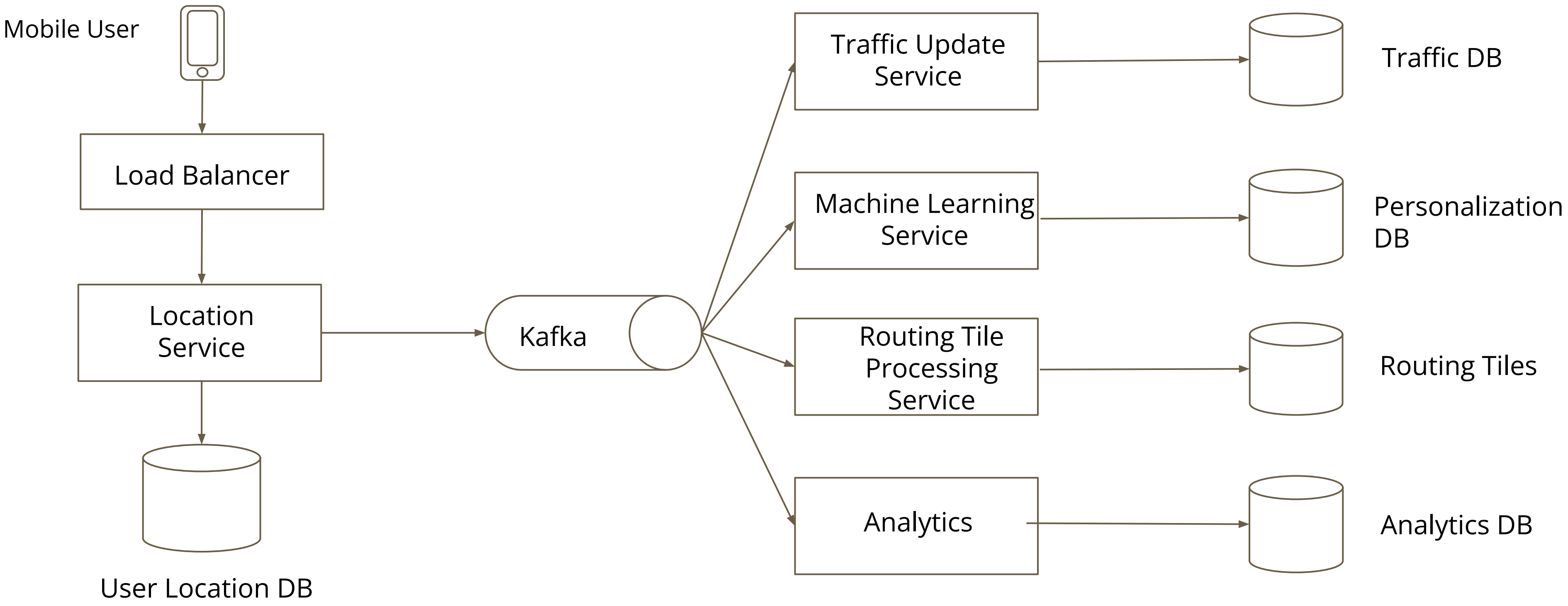
- Precomputed images of the world map

  Precompute images at different zoom levels
  Store them on CDN, backed by cloud storage like AWS S3.

# Step 3: Design Deep Dive

**Services:**

- Location service



# Step 3: Design Deep Dive

**Services:**

- Map Rendering

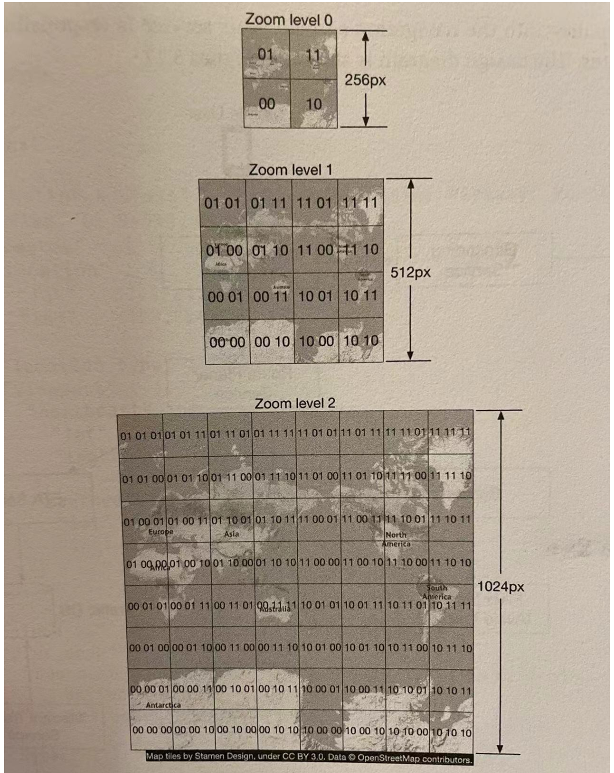


Figure 3.16: Zoom levels

Google Maps uses 21 zoom levels.
The higher zoom level, the bigger the size of files.

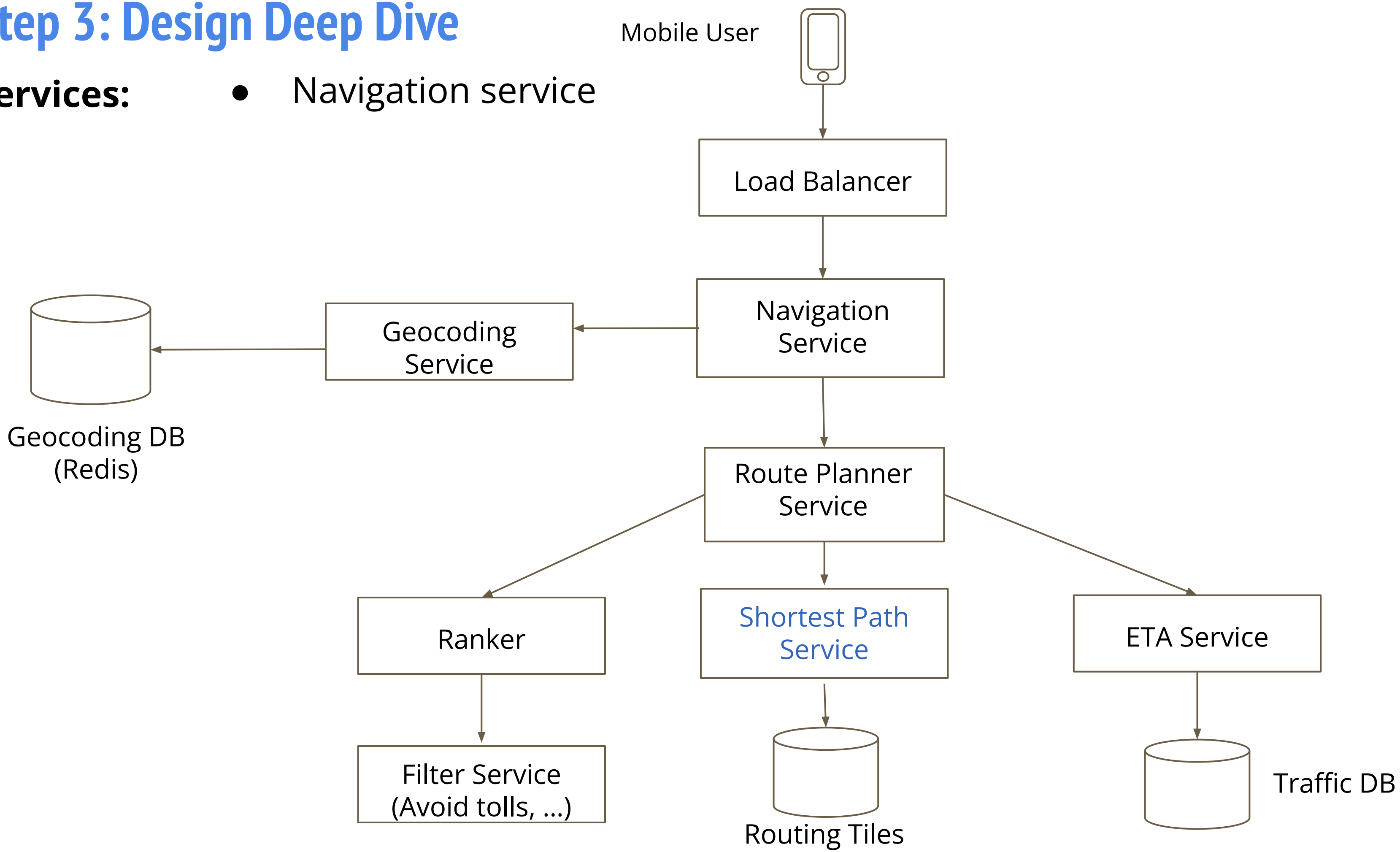
Optimization: use vectors

Sending vector information (paths and polygons), instead of images over the network.

# Step 3: Design Deep Dive

**Services:**

- Navigation service



# Step 3: Design Deep Dive

**Services:**

Shortest-path service: A* pathfinding algorithms

- It receives the origin and destination in lat/lng pair, and returns the top-k shortest paths without considering traffic or current filter.
- The lat/lng pairs are converted to geohashes, and load to the start and end-points of routing tiles
- The algorithm starts from origin routing tile, traverse the graph, and hydrates additional neighboring tiles. The algorithm could enter the bigger tiles containing only highways, until a set of best routs is found.

ETA service: calculate estimate time for each routes

- It use machine learning to predict the ETAs based on the current traffic and historical data.

Ranker service: after ETA prediction, apply possible filters as defined by the user.

- E.g. avoid toll, avoid freeways
- Returns top-k results to the navigation service.

Delivery protocols:

- Mobile push notification
- Long polling
- **WebSocket**
- Server-Sent Events (SSE)

# Step 4: Wrap Up

Key features:

- Location update
- ETAs
- Routing planning
- Map rendering

Further expanding the systems:

- Multi-stop navigation