Build Proximity System

Requirements clarification

- **Functional requirements**
  - Change places: Users should be able to add/delete/update places.
  - Search places: Given their location (longitude/latitude), users should be able to find all nearby places within a given distance.
- **Non-functional requirements**
  - High availability.
  - Low latency.
  - High availability is desirable (It should be ok for a user doesn't see a place for a while).

Estimation

- **Traffic estimation**
  - Our system will be read-heavy.

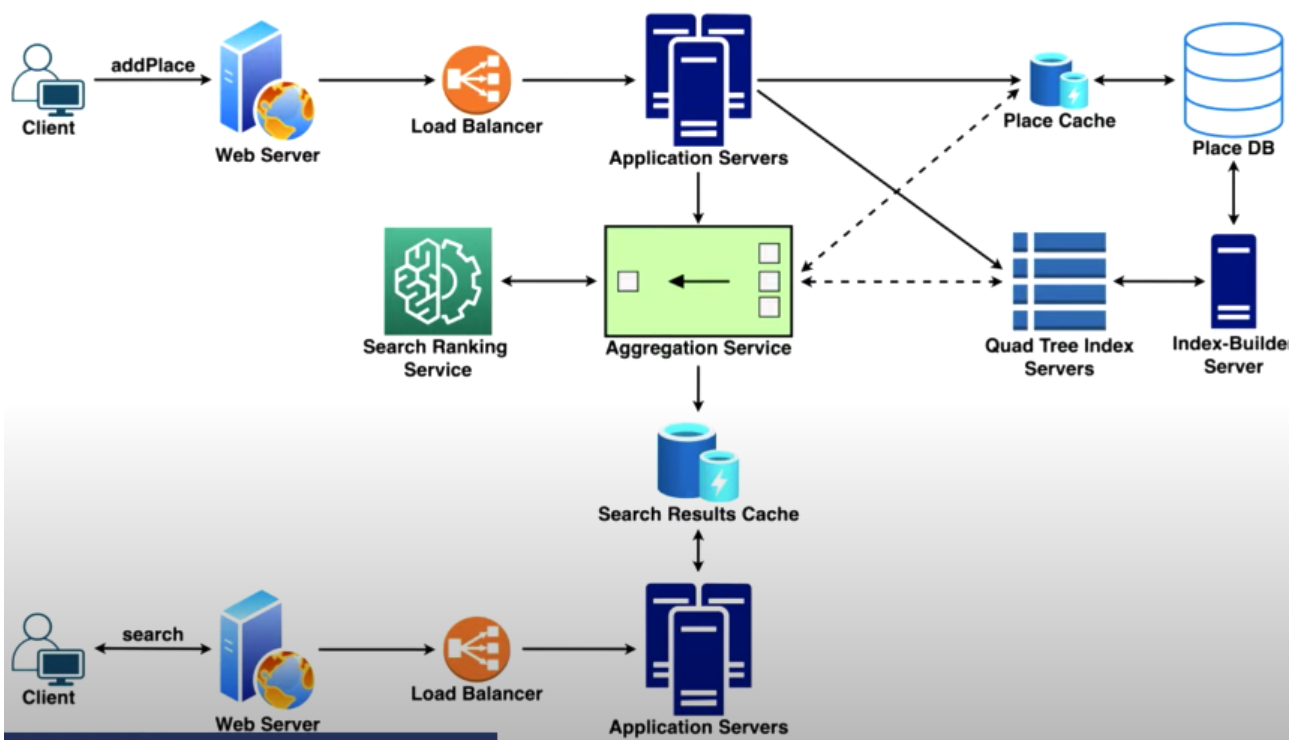
Data model definition

- **Schema**
  - Table 1: Place
    - LocationID (8 bytes): Uniquely identifies a location.
    - Name (256 bytes)
    - Latitude (8 bytes)
    - Longitude (8 bytes)
    - Description (512 bytes)
    - Category (1 byte): E.g., coffee shop, restaurant, theater, etc.

High-level design

- **Solutions**
  - SQL
    - Select * from Places where Latitude between X-D and X+D and Longitude between Y-D and Y+D
    - Not efficient
      - Need to use range to query 2 columns.
  - Grids (Static size grid)
    - Divide the whole map into smaller grids to group locations into smaller sets.
    - Select * from Places where Latitude between X-D and X+D and Longitude between Y-D and Y+D and GridID in (GridID, GridID1, GridID2, ..., GridID8)
    - It could be a problem there are a lot of places in a single grid.
  - QuadTree (Dynamic size grids)
    - Structure
      - Each node has four children.
      - If a node reaches our limit of 500 places, we will break it down to create four child nodes under it.
      - Leaf nodes represents the grids that cannot be further broken down.
      - Root node represents the whole world in one grid.
    - Traversal
      - If the current node has children, move to the child node that contains our desired location and repeat this process.
    - Find neighboring grids
      - Connect all leaf nodes with a doubly linked list.
      - Through parent nodes (each node has a pointer to access its parent).

Detailed Design


addPlace
Client
Web Server
Load Balancer
Application Servers
Place Cache
Place DB
Search Ranking Service
Aggregation Service
Quad Tree Index Servers
Index-Builde Server
Search Results Cache
search
Client
Web Server
Load Balancer
Application Servers