System Design - Ride **Sharing System** Uber Eng.: https://eng.uber.com/ Req. - Functional Rider https://www.youtube.com/wat ~ See available riders in vicinity **GPS** ch?v=umWABit-wbk **GPS** ~ Hail a ride ~ Notification of ETA/Location LB ~ Pay for a ride ~ Ride Pricing' ~ ETA Calc. Driver Location Driver Tracking Service Drivers/ App ~ Notification of Book ~ Location of User ~ Driving Directions Rider Driver Regionally Sharded Location Reg. - Non-Functional (Write Heavy) Map/ State Cache Local Digest Regionally Sharded ~ Real Time Data (Refreshed every 5s) Cache ~ Location Accuracy LB Rider Query ~ 10 M Drivers active daily ~ 100 M Riders active daily Service ~ 1 Ride daily ~ 99.99% Availability (1 hour/year outage) ~ Identifies Rider Cell LB (Backup Data Center) Queries and returns Drivers in that WebSocket Scale Calculations Connection ~ Location Data 110M+ \sim TPS 110M/5s = 22M TPS ~ 700 B Rides/day Key Design Issues ~ Location Data Refresh/Accuracy ~ Scale of incoming location data ~ Consistency of location writes/reads Rider APIs ell Based Queues Notifier ~ getDrivers(riderid, loc_lat, loc_long) Service ~ sendLoc(dirverId, loc_lat, loc_long, timestamp) Sharded by Geo-Cell Datastore ~ Consistent

Queues

Worker

Worker

Hashed

Rides

Rides

Service

Rider (NoSQL)

["riderId", "name", "location_lattitude", "location_longituge",

"last_update_timestamp", "geo_cell"]

Driver (NoSQL)

["driverId", "location_lattitude", "location_longitude"

"last_update_timestamp", "geo_cell"]

Rides (SQL)

["rideId", "riderId", "driverId", "rider_lat_long", "driver_lat_long",

"dest_lat_long", "timestamp"]