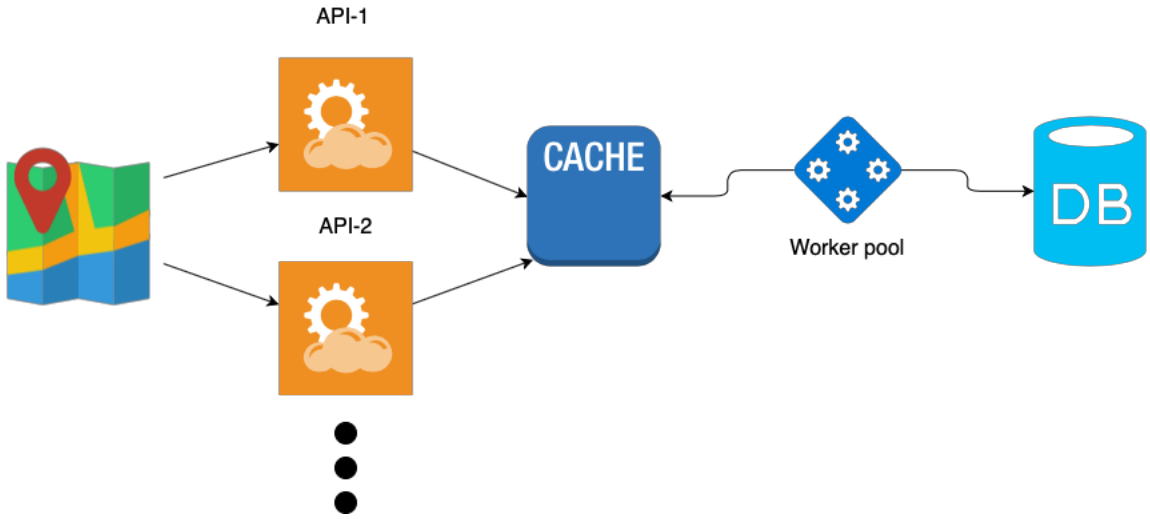
PLOTING/READING 100K SCOOTERS ON A MAP

BY LIM CHOOI GUAN, AWS-CSA

Let's say we have 100,000 scooters, how can this be displayed on a map?

Architecture



BACK END STORAGE

With regards to back-end storage, the scooter data should be sharded and stored in different partitions. For example, we can store scooters 1-1000 in partition 1, 1001 to 2000 in partition 2, etc. Storage of scooters should be as random and equally distributed as possible. This is to avoid any "hot" partitions. Distribution is typically done by a hashing function, i.e. f(x) where x is the scooter number/identifier which will return the partition id. Since I primarily work with AWS I can recommend that scooter data be stored in DynamoDB which is a NoSQL database. All data should be stored in only one table, unlike the RDMS model. Using NoSQL makes it easier to scale horizontally. Since the problem only involves reading, another way to make this scale is to create many replicas of the same database. This also allows for high availability when one database goes down.

CACHING TIER

Now that all scooter data is sharded, you can create N number of workers (running node.js or whichever language you prefer) which can query X number of scooters and post them to a cache like Redis/Memcached. The cache can be periodically refreshed if required through a pub sub mechanism (i.e. workers subscribe to change events on the database layer) causing the positions of the scooters to invalidate, thereby updating the

scooter's position in the cache. The cache can also serve as a failover mechanism (i.e. provide graceful degradation, you can see an example of how I've done this for this other project) if/when the database fails. That way the last location of all scooters will still be present even if stale. An API layer can be built upon to expose the cache. Multiple copies of the same API can then be deployed to allow for high availability and can also be used to allow parallelization of queries. For example one API could be queried to retrieve locations of scooters 1-1000, and another copy of the same API could be queried to retrieve locations of scooters 1001-2000, and so. Ideally the caching layer should also be distributed to allow for high availability. It is possible to run Redis/Memcached as clusters.

PLOTTING THE DATA ON A MAP

With regards to plotting the data, with the data readily available on different partitions this should be quite straightforward. Create N asynchronous worker threads on the front end that connect to the APIs which poll directly from the cache. This can be done via web sockets for example. Then simply plot the positions on the map. Now to plot 100K scooters could be too resource intensive on the client side so there should be some intelligence where you only plot scooters which are within the map's viewable regions and only plot those. Again this can be done in an asynchronous manner, and it's probably a good idea to do a merge of the positions when the user zooms out and only show separate positions upon zooming in on the map.