Issue: Design a system that can support million users send latitude and longitude to server per minute.

Submit by: Yuhao Wang

My exploration:

1.Preparation:

Before talking the design idea of this system, there are several situations where need to discuss.
Firstly, whether there would be such situation where two different areas have the different number of sending requests. For instance, people who live in metropolis such as New York City and San Francisco would send much more location information than the people who live in small city like Salt Lake City or Honolulu.
Secondly, whether there would be some persons that send location information with a high frequency than other persons.
Thirdly, latitude and longitude are the most important elements that must be sent and stored in our databases. However, whether we also need to add other attributes to our database at the same time. If there is one person who send same location message as past, what action should we take?

2.Assumption:

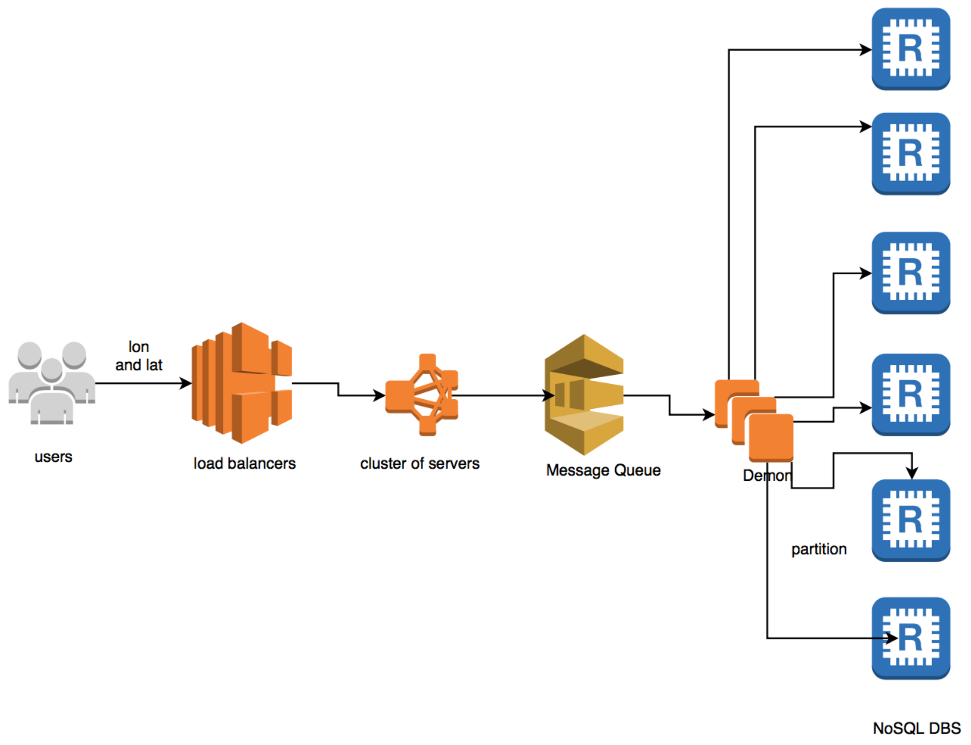
That system should support million users send geolocation message to our server on every minute. In this situation, NoSQL databases such as Redis, MongoDB and HBase would be a better choice than relational databases. Because such large information usually needs to apply for distributed system, and NoSQL databases can support distributed system much well. And we also want to get user's latitude and longitude by user id, key-value pairs are good way to implement it.

3.Performance:

1). Storing Data
Considering the concerns on preparation section, our system should have some features that can address that matters.

Step 1: Message Queue (Asynchronization)
If we got the same location information a lot from one area or one person in one minute, we can take some delay function or lazy function to decrease the message pressure from same geolocation or same person. For example, we can let that same location information stay in queue for 10-15 waiting seconds so that it will spare the channels to other different location information. Because our server has already received that same location information, 10-15 seconds delay would affect less.

Step 2: Sharding and Partition
In order to guarantee on receiving the geolocation information from small city or inactive users, we also can implement some function to normalize the chances of receiving geolocation. Because there may be some important information from that area or people, there could be some loss or misconduct if we cannot assure the equal chances of receiving. In this case, we can hash the geolocation information into different partition, but we must assure that the hash function cannot change the location accuracy. According the outcomes of hash step, we can let the similar geolocation information to its own partition parallelly.



draft for the system

2) Accessing Data
We have stored data, so how to make us access data fast would be our next main concern. Let us imagine this situation where we want to get all the data from a point in New York City extending to 10km, how should we do?

Choice 1: Indexing the database by K-d tree
K-d tree is a very useful way to search by range or nearest neighbor. Given a center of O and a parameter K, we can get K data points nearest to the center O. By this way, we can improve the speed to get the nearest data when being given a center.

Choice 2: Splitting the map into S2 cell
We can split the map to the different rectangles, every couple latitude and longitude would go into that corresponding cell. Then we just need to run binary search, we can use O(logn) time to get the result.