Scalability as User Base gets larger

Assuming the anagrams application is used by millions of users, and most of the users would like to do lookups for anagrams. We can look into following aspects to scale the system efficiently.

When the input file gets too large so will the database. As the database gets larger, reading the data from database, it will eventually get very slow. By having only single database we will also introduce a single point of failure. Hence the next step should be sharding the database. The database can become huge especially if there will be more words coming in future. By sharding the database, we can split the data across multiple machines while ensuring we have a way of figuring out which data is stored on which machine. One way to split this data efficiently is by maintaining a lookup table. Such that when a request is received to find all anagrams of a word, the word can then be passed in lookup table, lookup table should return a machine ID where database containing this word is stored. Once we have machine ID we can sort the word and look for it as key of database. The tricky part is when we have sharded the database and need to add more words in it then it should be done carefully making sure that one database should not get too big while others stay free.

We can also consider caching results by analyzing common requests received. This can be done using memcache or redis. Based on the requests it can be checked what kind of requests are high in demand or we can cache most recent requests thus whenever a new request comes in we can look it up in cache if found then return the result. Using redis instead of memcache will be better in anagrams application because redis support key value pairs thus the sorted word of anagrams application can act as key and hashSet of anagrams will be values. One thing to take care of in using any of caching methods is doing cache invalidation. Which is making sure that if some data is updated in database and we have the data in cache then it should also be either update or remove from cache. Otherwise cache will return old data.

Sharded databases can get corrupted and we may end up losing big chunk of data. Looks like sharded databases are our single point of failure. To prevent this, we can introduce database replicas. The replicas can also be used for reading the data plus if one of the two i.e database or its replica gets corrupted we still have the data. We then can quickly fix the issue which has caused loss of data and make sure data is again present in both database and its replica.

Lastly the data center where the data is stored can be our single point of failure. There is a possibility that the power of the data center goes out for some reason or some other unfortunate thing might happen. In order to make sure we are not dependent on one data center we can distribute the data or keep replicas of data in data centers located in different locations. This can also help in better accessibility to data by users. Based on user's location they will be directed to the data center which is closer to them hence reduced response time more happy customers.

High level design of Anagram application