Scaling ideas for Search Engine (Super high level since it is a one pager)

Current Problem

The solution implemented serves the need of a single user with a finite set of seed words. However this solution would not be able to scale if there are a million users and the seed words grow beyond a particular size as a single machine cannot store a big Trie or serve the requests of a million users

Some estimates

If we have a million users and a single user queries the search engine at a minimum of 10 times a day
Total requests = 10^6 * 10 = 10^7 per day ~= 100 QPS. Peak traffic ~= 200QPS
Lets say we get 100 characters in every query and we store every query in our global trie
Size ~= 10^6 * 10 * 100 = 10^9 B = 1GB/day = 365 GB/yr
Our system needs to be able to handle the scale and have low latency.
High level ideas

Horizontal scaling

- Have multiple servers which can listen for the incoming requests. We can have a Load balancer delegate requests to the appropriate server (based on the load / server closest to the user)

Database

- Its clear that we need to be able to store the trie in either a relational database or a NoSQL database. Since our system does not have strong consistency reqs (its probably fine if we miss a particular result in our suggestions) we can use a NoSQL.
- Each node in the trie will have a unique id. We can probably use a key value store with the following data model.
    - Key: NodeId
    - Value: {
        - Children : List of NodeIds which are its children
        - Words: List of complete words in the NodeId subtree
          }
- We can have a redis cache infront of our DB which takes a daily snapshot of the trie, puts a TTL so that it automatically gets evicted. This would make our reads much faster
- We can implement a clever sharding strategy. We can have a shard manager maintain clusters for each country so that the traffic gets distributed accordingly. We can scale the clusters according to the traffic.

Aggregators

If we make write operations to the global trie at the same time reads are coming in, it might lead to synchronization issues. We have to follow the PACELC theorem here and make a tradeoff

between consistency and latency. One approach might be to put a lock only on the subtree of the node being impacted.
Another solution which I prefer is to have another database which aggregates all the queries for a single day and we would have dedicated servers which would be responsible to perform the updates to the trie database during down time (and invalidate cache).