Search Auto-Complete

dinner

d ->
di ->
din ->....

Requirements and Clarifications

When do we start suggesting the recommendations? First letter onwards
What is the ranking system of these searches and should we design that? Frequency based and maybe let's make it flexible
How many results should we respond back with? 5
How many users use this product? 10 million DAU

Non-Functional Requirements

Data-consistency is on the low side
High availability

Scale Calculations

QPS: 10 million DAU = 5000 QPS (5 characters on average per query; 10 queries per user)

High-Level Abstract Design

Application Layer Design

1. People make search queries: on submitting a search query (pressed Search button or Submitted your query via Enter key), that goes into our database as a search query.
    a. submitSearch
2. People receiving auto-complete recommendations:
    On blur AJAX call that calls into the API:
    getAutoCompleteRecommendations(search_string): { List of the top five recommendations }

Data Layer Design

1. In memory: Tries
2. In database: search query (from step 1 of application layer design) and the number of times it has been searched.

Database, in an async process, builds the trie in memory.
Memory: Memory on a server.

Scaling the High-level Design

1. DB Sharding -> Shard off of the first letter; in some cases second letter because not all letters have the same frequency.
2. Trie Data Structure memory store also gets sharded.
3. Caching
4. Trending Topics

Future Enhancements

1. Rate limiting
2. Analytics