High-level Design of the Typeahead Suggestion System

Get an overview of the high-level design of the typeahead suggestion system.

We'll cover the following

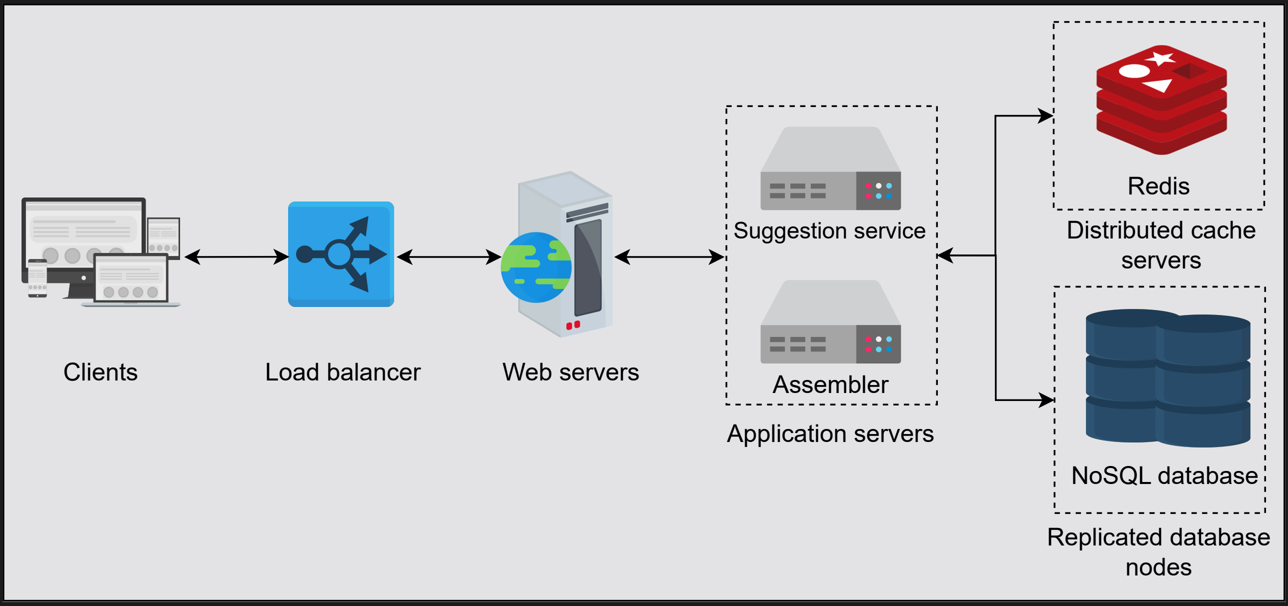
High-level design

According to our requirements, the system shouldn't just suggest queries in real time with minimum latency but should also store the new search queries in the database. This way, the user gets suggestions based on popular and recent searches.

Our proposed system should do the following:

- Provide suggestions based on the search history of the user.
- Store all the new and trending queries in the database to include them in the list of suggestions.

When a user starts typing a query, every typed character hits one of the application servers. Let's assume that we have a **suggestions service** that obtains the top ten suggestions from the cache, Redis, and returns them as a response to the client. In addition to this, suppose we have another service known as an **assembler**. An assembler collects the user searches, applies some analytics to rank the searches, and stores them in a NoSQL database that's distributed across several nodes.



The high-level design of the typeahead suggestion system

Furthermore, we also need load balancers to distribute the incoming requests evenly. We also add application servers as entry points for clients so that they can forward requests to the appropriate microservices. These web servers encapsulate the internal system architecture and provide other services, such as authentication, monitoring, request shaping, management, and more.

API design

Since the system provides suggestions to the user and adds trending queries to the databases, we need the following APIs.

Get suggestion

```
getSuggestions(prefix)
```

This API call retrieves suggestions from the servers. This call is made via the suggestion service and returns the response to the client.

The following table explains the parameter that's passed to the API call:

Parameter	Description
`prefix`	This refers to whatever the user has typed in the search bar.

Add trending queries to the database

```
addToDatabase(query)
```

This API call adds a trending `query` to the database via an assembler if the query has already been searched and has crossed a certain threshold.

Parameter	Description
`query`	This represents a frequently searched query that crosses the predefine limit.

Point to Ponder

Question

Instead of updating the whole page, we just need to update the suggested query in the search box in real time. What technique can we use for this purpose?

Show Answer

In the next lesson, we'll learn about an efficient data structure that's used to store the suggestions or prefixes.