



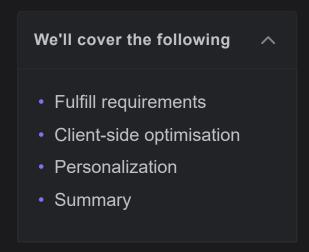






Evaluation of the Typeahead Suggestion System's Design

Evaluate the design of the typeahead suggestion system based on the non-functional requirements of the system.



Fulfill requirements

The non-functional requirements of the proposed typeahead suggestion system are low latency, fault tolerance, and scalability.

- Low latency: There are various levels at which we can minimize the system's latency. We can minimize the latency with the following options:
 - Reduce the depth of the tree, which reduces the overall traversal time.
 - Update the trie offline, which means that the time taken by the update operation isn't on the clients' critical path.
 - Use geographically distributed application and database servers. This
 way, the service is provided near the user, which also reduces any
 communication delays and aids in reducing latency.
 - Use Redis and Cassandra cache clusters on top of NoSQL databas
 clusters.



- Fault tolerance: Since the replication and partitioning of the trees are
 provided, the system operates with high resilience. If one server fails,
 others are on standby to deliver the services.
- Scalability: Since our proposed system is flexible, more servers can be
 added or removed as the load increases. For example, if the number of
 queries increases, the number of partitions or shards of the trees is
 increased accordingly.

Approaches to Fulfill Non-functional Requirements

Non-functional Requirements	Approaches
Low latency	traversal faster
Fault tolerance •	Replicating the tries and the NoSQL databases
Scalability	Adding or removing application servers based on the incoming traffic Increasing the trie partitions

Client-side optimisation

To improve the user's experience, we can implement the following client-side optimizations:

• The client should only attempt to contact the server if the user hasn't pressed any keys for some time—for example, any delay greater than 16 in the average delay between two keystrokes. This way, we can

- The client can initially wait till the user types a few characters.
- Clients can save a local copy of the recent history of suggestions. The rate of reuse of recent history in the suggestions list is relatively high.
- One of the most crucial elements is establishing a connection with the server as soon as possible. The client can establish a connection with the server as soon as the user visits the search page. As a result, the client doesn't waste time establishing the connection when the user inputs the first character. Usually, the connection is established with the server via a WebSocket protocol.
- For efficiency, the server can push a portion of its cache to CDNs and other edge caches at Internet exchange points (IXPs) or even inside a client's Internet service provider (ISP).

Personalization

Users receive typeahead suggestions based on their previous searches, location, language, and other factors. We can save each user's personal history on the server separately and cache it on the client. Before transmitting the final set to the user, the server might include these customized phrases. Personalized searches should always take precedence over other types of searches.

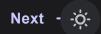
Summary

In this design problem, we learned how pushing resource-intensive processing to the offline infrastructure and using appropriate data structures enables us to serve our customers with low latency. Many optimizations lend themselves to specific use cases. We saw multiple optimizations on our trie data structures f condensed data storage and quick serving.

Тт







>





