Evaluation of a Distributed Search's Design

Analyze how our design meets the requirements.

We'll cover the following

Availability

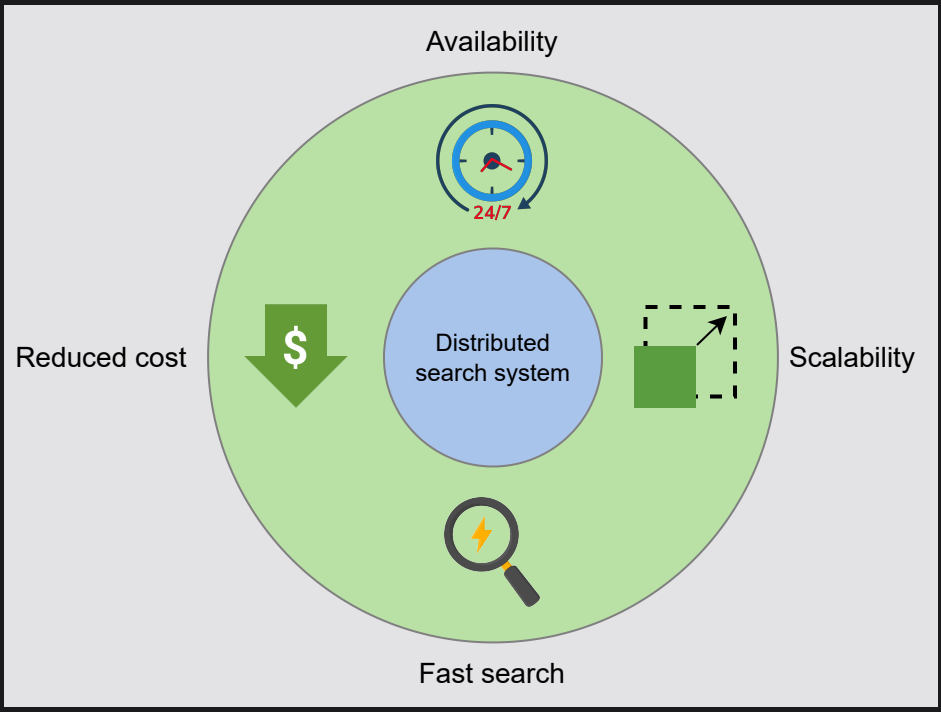
We utilized distributed storage to store these items:

- Documents crawled by the indexer.
- Inverted indexes generated by the indexing nodes.

Data is replicated across multiple regions in distributed storage, making cross-region deployment for indexing and search easier. The group of indexing and search nodes merely needs to be replicated in different availability zones. Therefore, we deploy the cluster of indexing and search nodes in different availability zones. So, if a failure occurs in one place, we can process the requests from another cluster. Multiple groups of indexing and search nodes help to achie high indexing and search availability. Moreover, in each cluster, if a node dies, another can take its place.

The indexing is performed offline, not on the user's critical path. We don't need to replicate the indexing operations synchronously. It is unnecessary to respond to the user search queries with the latest data that has just been added to the index. So, we don't have to wait for the replication of the new index to respond to the search queries. This makes the search available to the users.

> **Note:** Once we replicate the latest data in all groups of indexing nodes and the search nodes have downloaded it, then the search queries are performed on the latest data.



Guarantees provided by the distributed search

Scalability

Partitioning is an essential component of search systems to scale. When we increase the number of partitions and add more nodes to the indexing and search clusters, we can scale in terms of data indexing and querying.

The strong isolation of indexing and search processes help indexing and search scale independently and dynamically.

You're a software engineer for a rapidly growing e-commerce site aiming to redefine search with lightning-fast personalized results. Currently, user experience is suffering due to slow search response times, especially during peak loads.

Your task is to:

- Identify components of distributed search that can benefit from caching to improve performance.
- Identify the potential trade-offs to consider.

To get started, say "hello" to Ed in the widget below, and it will lead the way.

educative

Powered by AI

10 Prompts Remaining

Prompt AI Widget

Our tool is designed to help you to understand concepts and ask any follow up questions. Ask a question to get started.

Enter Prompt Here...

Fast search on big data

We utilized a number of nodes, each of which performs search queries in parallel on smaller inverted indices. The result from each search node is then merged and returned to the user.

Reduced cost

We used cheaper machines to compute indexes and perform searches. If one node fails, we don’t have to recompute the complete index. Instead, some of the documents need to be indexed again.

Conclusion

A search system is required for almost every application. We have seen that it isn’t possible to develop a search system that can run on a single node. We utilized a parallel computation framework and low-cost machines to build a search system that is available, scalable, and highly performant.

Back

Completed

Next

Scaling Search and Indexing

System Design: Distributed Logging