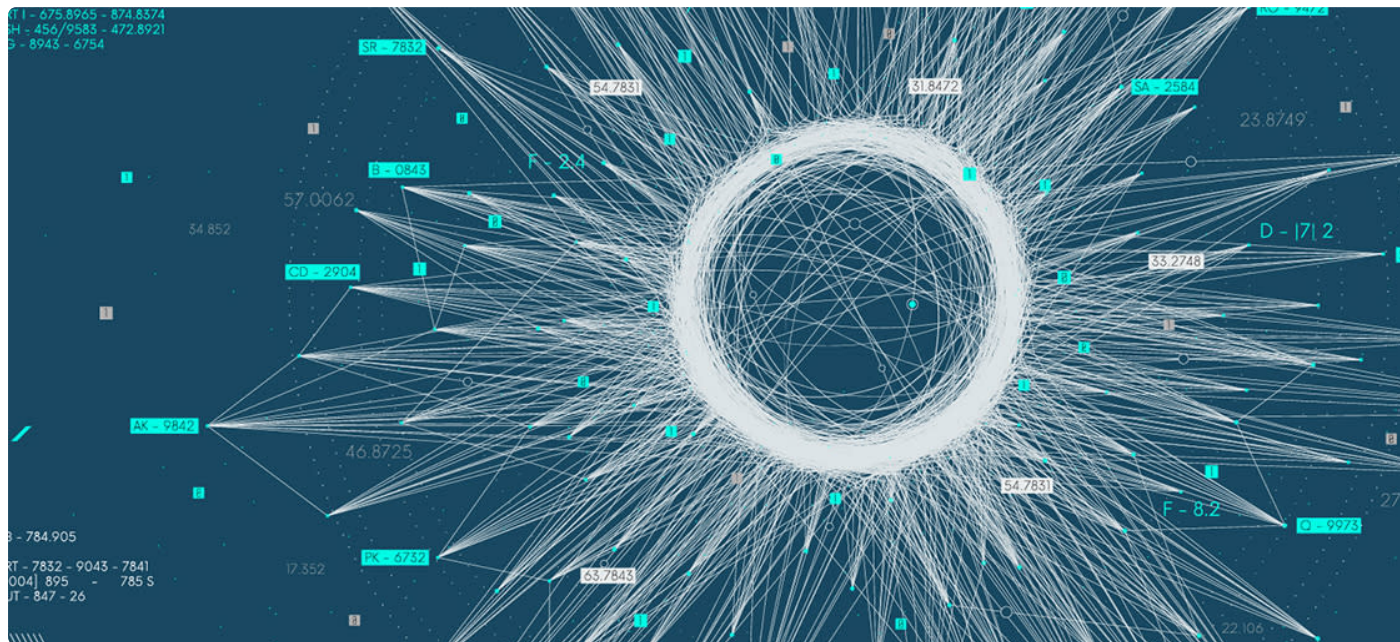


Reduce Elasticsearch Search Latency and Optimize Search Performance

Jochen Kressin · 2024-05-21



Elasticsearch is a highly popular and versatile search engine, known for its speed, scalability, and ability to handle large volumes of data. However, ensuring optimal search performance and minimizing latency is crucial for maintaining user satisfaction and operational efficiency. In this article, we'll explore practical tips on how to reduce search latency and optimize search performance in Elasticsearch.

1. Optimize Your Indexing Strategy

Proper Mapping: Define explicit mappings for your indices to control how data is indexed. Avoid using dynamic mapping as it can lead to inefficient data structures. Use appropriate field types and settings to reduce index size and improve search performance.

Avoid Oversharding: Increasing the number of shards can lead to better search performance, since the query execution can be distributed amongst more nodes in the cluster. However, having too many shards can also increase overhead and degrade performance during indexing. Calculate the optimal number of shards for your index based on your data volume and query patterns. Also, having many small shards could cause a lot of network I/O which can severely impact search performance.

Use Index Templates: Index templates ensure that your indices are created with predefined settings and mappings, leading to more consistent and efficient performance.

2. Refine Your Queries

Limit Query Scope: Use filters to narrow down the scope of your queries. Filters are cached, which can significantly improve performance for repeated queries.

Optimize Query Structure: Write efficient and well-structured queries. Avoid overly broad queries and use the minimum required query clauses.

Wildcard queries: Avoid wildcard queries, especially leading wildcard queries. They can cause the entire Elasticsearch index to be scanned. The same is true for Regex queries and parent-child relationships.

Term queries: Utilize term queries for exact matches, particularly on keyword fields. Elasticsearch, by default, creates text and keyword fields for each string value field, unless a specific mapping is provided. Often, users mistakenly apply the match query to keyword data types such as product ids, which incurs extra cost due to the analysis process involved in match queries. For optimal performance, it is advisable to always employ term queries when working with keyword data types and in scenarios where exact matching is required.

3. Manage Search Resources

Adjust Thread Pools: Fine-tune the size and queue length of search thread pools to handle your specific workload. Be cautious, as inappropriate settings can lead to resource contention.

Implement Query Throttling: Use query throttling to control the load on your Elasticsearch cluster, especially during peak times.

Deleted documents: A large number of deleted documents in an Elasticsearch index can cause search performance issues. Use the [Force merge API](#) to remove deleted documents and optimize the shards size.

4. Leverage Elasticsearch Features

Use Search Shards Preference: Specify shard preference in your queries to optimize the search. For instance, prefer local shards to reduce network latency.

Implement Source Filtering: When you only need certain fields, use source filtering to return only the relevant parts of the documents.

5. Optimize Hardware and Infrastructure

Use Faster Storage: Elasticsearch benefits significantly from SSDs over HDDs, especially in terms of I/O operations and data retrieval speeds.

Optimize Network Infrastructure: Ensure that your network infrastructure can handle the data transfer efficiently. Use faster network interfaces and connections where possible.

6. Use Caching Effectively

Leverage Field Data Cache: For fields that are frequently used in sorting, aggregations, or terms queries, ensure that they are loaded into field data cache.

Adjust Cache Sizes: Monitor and adjust the sizes of different caches like the request cache and field data cache based on your workload.

Summary

Optimizing search performance in Elasticsearch involves a combination of proper indexing, efficient query design, resource management, and hardware optimization. Regular monitoring and tuning based on performance metrics are crucial to maintaining low latency and high throughput. By implementing these tips, you can significantly enhance the search experience and efficiency of your Elasticsearch cluster.

----- Original Image Source: [Maxger](#) on Shutterstock

This article was published on the Search Guard blog: <https://search-guard.com/blog/reduce-elasticsearch-search-latency-optimize-search-performance/>

Search Guard is Security and Alerting for Elasticsearch and Kibana — search-guard.com · [Start a free trial](#)