# Elasticsearch Cost Optimization

Jochen Kressin · 2023-03-28


Elasticsearch has become the go-to platform for building highly scalable and responsive search and analytics solutions. However, running a large Elasticsearch cluster can be expensive, especially when it comes to infrastructure costs, maintenance, management, and subscription. In this article, we will discuss effective methods to lower costs and total cost of ownership (TCO) for running large Elasticsearch clusters, while maintaining optimal performance and reliability. Scale your cluster, not your cost!

## Data Management and Retention

Reducing the volume of indexed data can significantly lower storage costs and improve cluster performance. Here are some strategies to efficiently manage your data:

### Use ILM to Manage Your Indices

Automate the lifecycle of your indices by creating Index Lifecycle Management (ILM) policies that define actions like rollover, shrink, force merge, and delete. This will ensure that your data is moved to appropriate storage tiers (hot, warm, cold, or frozen) or gets deleted based on its age and usage patterns, reducing the costs associated with storage and compute resources.

### Use Data Tiers and Searchable Snapshots

Elasticsearch supports data tiers (hot, warm, cold, and frozen) that allow you to move data across different storage types based on its age and access frequency. Configure your ILM policies to automatically transition data between tiers, ensuring that your storage costs are minimized without sacrificing data availability.

For data that is rarely accessed, consider using searchable snapshots with low-cost object storage provides, like AWS S3, Microsoft Azure Storage, and Google Cloud Storage.

Data that you want to archive or need only for compliance purposes consider external archive solutions like Amazon Glacier.

These solutions offer long-term storage at a lower cost, allowing you to save on Elasticsearch cluster resources.

### Summarize Older Data Using Roll Ups

Elasticsearch provides a Rollup API which rollup "provide a means to summarize and store historical data so that it can still be used for analysis, but at a fraction of the storage cost of raw data".

This is especially helpful when working with huge amounts of time-series based data: It might be required to run analytics over second-by-second intervals for the most current data. However, for historical data is might be sufficient to look at larger intervals, like day-to-day trends.

Once this historical data has been summarized and rolled up, the original data can be deleted or archived.

## Replica Settings

Configuring the number of replicas for your indices can have a significant impact on both cost and performance. Consider the following approaches:

### Adjust Replica Count Based on Usage

Assess the read and write patterns of your indices and adjust the number of replicas accordingly. For read-heavy workloads, increasing the replica count can improve query performance, but it also increases storage costs. On the other hand, if an index is primarily used for write operations, reducing the replica count can save on storage costs without compromising write performance.

### Use Adaptive Replica Selection

Enable Elasticsearch's Adaptive Replica Selection (ARS) feature to dynamically route search requests to the most appropriate replica. This can help improve query performance and reduce the need for additional replicas, subsequently lowering storage and infrastructure costs.

## Shard Settings

Proper shard configuration can lead to better resource utilization and lower costs. Implement the following strategies to optimize shard management:

### Choose the Right Number of Primary Shards

Evaluate the expected size of your indices and determine the optimal number of primary shards accordingly. Allocating too many primary shards can lead to unnecessary overhead and increased costs, while too few shards can result in reduced performance. As a rule of thumb, aim for shards in the range of 10-50 GB.

### Implement Shard Filtering and Routing

Utilize Elasticsearch's shard filtering and routing capabilities to direct queries to specific shards, reducing the resources required for query execution. By efficiently routing queries to relevant shards, you can minimize the need for additional shards and reduce overall costs.

## Optimize Mappings and Templates

To reduce the amount of data and thus reducing storage costs, optimize index mappings and templates.

Design your index mapping to minimize the number of required fields, and use appropriate field types. Whenever possible, avoid the storage of the *_source* field altogether.

You can filter out unnecessary fields from your source data by setting up an index pipeline and a remove processor.

## Cache Optimization and Query Performance Tuning

Improving cache utilization and optimizing query performance can help reduce infrastructure costs by minimizing the resources required for processing queries.

### Use Query Caching

Elasticsearch automatically caches the results of frequently executed queries. You can fine-tune cache settings like cache size and expiration time to ensure that the most relevant data is cached, reducing query response times and decreasing the load on your cluster.

### Optimize Queries

Analyze slow queries and optimize them using techniques like pagination, query filtering, and result trimming. Additionally, use the "_explain" API to identify poorly performing queries and make necessary adjustments to improve their execution efficiency.

Whenever possible, avoid costly actions like parent-type queries, nested queries and nested aggregations, or running aggregations over irrelevant data. Keep in mind that Elasticsearch is a document-oriented data storage engine. Instead of trying to model complex relationships between documents and indices, in most cases, it is better to denormalize your data, storing redundant data in multiple documents.

## Lower your Subscription Cost

Elastic Stack subscriptions are costly. Moreover, the costs involved with a subscription are bound to the number of nodes in your cluster, and therefore directly connected with your data size and the usage of your cluster. When you scale your cluster, you also have to scale the cost. Running an Elasticsearch cluster with a subscription may seem acceptable in the beginning, but the cost can rise quickly.

> Scale your cluster, not your cost

While the commercial Elastic Stack offers a lot of features, most users will rarely need them all. Usually, Security and Alerting are enough.

Consider switching from a costly Elastic Stack Subscription to alternative plugins like Search Guard. Search Guard offers a free Community Edition and a paid Enterprise Edition which provide enterprise-level Security and Compliance features. Signals Alerting for Elasticsearch and Kibana detects data anomalies in your Elasticsearch cluster and sends notifications on various channels. It is fully integrated with Search Guard Security and the community version is free as well.

Image: shutterstock / NESS Design

---

This article was published on the Search Guard blog: https://search-guard.com/blog/elasticsearch-cost-optimization/