Optimizations

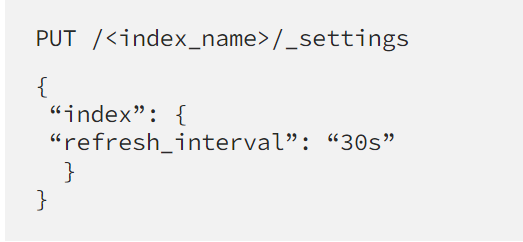
1. Refresh Interval – tokens are kept in in-memory buffer if the indexing process is heavy and later transferred to shard segment. This interval is called refresh.

Higher the better but cannot be too high depends on use case.

Default value – 1s

Change to – 30s

To change call following API



1. Cache –
   1. Node-level query cache –

Default value – 10,000 queries, 10% of heap memory

Change it to – Use case dependent, higher the better

To change – change the following parameter

Indices.queries.cache.size – accepts the value in percentage or memory.

Demerits – In some use cases, if you run same request running twice in a row, each request will go to different shard copies, preventing node-level from helping.

* 1. Shard-Level request cache –

Default value – 1%

Change it to – Use case dependent, higher the better

How to change – edit opensearch.yml and change the parameter indices.request.cache.size

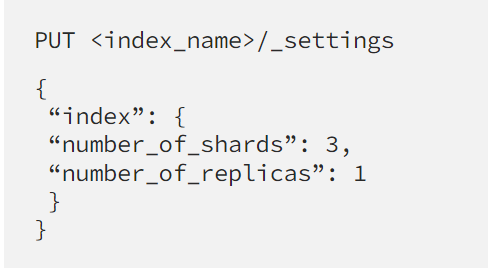
1. Sharding and replicas –

Too High impacts performance. (Merged results)

Recommended shard size – 30 to 50 GB

More replicas can improve the performance

How to change



1. curl -XPUT '[http://localhost:9200/\_all/\_settings?preserve\_existing=true'](http://localhost:9200/_all/_settings?preserve_existing=true%27) -d '{  
     "index.translog.flush\_threshold\_size" : "8192MB"  
   }' -H 'Content-Type: application/json'

and the settings are reflected by using this:

GET \_cluster/settings?include\_defaults=true

1. Indices –
   1. Break your data and indices – can use one or more than one indices.
   2. Avoid nested fields – making as flat as possible results better performance.
   3. Consider mapping identifiers as keywords – term level query works better with keyword fields. Consider mapping to keywords if no plan on searching them using range queries.
   4. Reindex your data occasionally – data is immutable, while updating new doc is created, it’s given a version number which helps OpenSearch track the latest document. However, as a result of including both new and old documents, the index expands significantly. Reindexing addresses this issue. After reindexing, your indices will contain only the most recent information, saving memory and speeding up subsequent searched.



1. Using Plugin – KNN optimization

<https://opensearch.org/docs/latest/search-plugins/knn/performance-tuning/>

1. Deleted Docs – Use force merge API

Query based

1. Regex – can cause latency, best to avoid when you can.
2. Time-out and terminate – When executing heavy searches or result data is vast.(\*\*)
3. Size parameter – huge value to size parameter leads to vast no of hits causing several performance issues, batch request in smaller size is better. (\*\*PAGINATE) <https://opensearch.org/docs/2.4/opensearch/search/paginate/>
4. Template – using templates can reduce the amount of data over the internet, provides abstraction. <https://opensearch.org/docs/1.0/opensearch/search-template/>
5. Multi Search API – msearch can execute queries in parallel in opensearch. <https://opensearch.org/docs/2.4/api-reference/multi-search/>

Slow query optimization

The **slow query analysis** feature provided by OpenSearch allows you to view the details of slow queries that consume heavy logical computing unit (LCU) resources within a specific period of time. The feature also provides optimization suggestions to help you reduce costs.

Free to use.

<https://www.alibabacloud.com/help/en/opensearch/latest/optimize-slow-queries>

References

1. <https://towardsdatascience.com/bolster-opensearch-performance-with-5-simple-steps-ca7d21234f6b>
2. <https://opster.com/blogs/improve-opensearch-search-performance/>
3. <https://www.alibabacloud.com/help/en/opensearch/latest/optimize-slow-queries>