Search Latency After Upgrade Troubleshootin

Opensearch upgrades from Elasticsearch 6.8 to OpenSearch +1.0 resulting in increased search latencies for ES clients.

SEARCH LATENCY AFTER UPGRADE TROUBLESHOOTIN (HTTPS://W.AMAZON.COM/BIN/VIEW/AMAZONWEBSERVICES/SALESSUPPORT/DEVELOPERSUPPORT/INTERNAL/AMAZONELASTICSEARCH/PLAYBOOKS/SEARCHLATENCY/)

OPENSEARCH SUPPORT OPS HAVE OBSERVED AN INCREASE IN ESCALATIONS FOR OPENSEARCH CAUSING HIGH LATENCY AFTER UPGRADING FROM ELASTICSEARCH 6.8. BASED ON THE VERSIONING HISTORY OF OPENSEARCH, OPENSEARCH 1.X IS BASED ON ELASTICSEARCH 7.10 OPEN SOURCE. DUE TO THIS, MOST OF THE DEFAULT SETTINGS AND BEHAVIORS OF ELASTICSEARCH 7.10 ARE GOING TO BE IN THE EARLIEST VERSIONS OF OPENSEARCH 1.X, 2.X. THIS UPGRADE ALSO MEANS CUSTOMER DID NOT DEPLOY THEIR DATA IN ANY VERSION OF ELASTICSEARCH 7.X, MEANING THEY DID NOT TAKE INTO ACCOUNT ANY BREAKING CHANGES FROM ELASTICSEARCH 6.8 TO 7.10 WHERE THE MOST SIGNIFICANT CHANGES OCCUR DURING MAJOR VERSION UPGRADE FROM 6.0 TO 7.0.

- Elasticsearch 7.0 Breaking Changes

LATENCY ISSUE:

ELASTICSEARCH 7.0 INTRODUCED A VARIETY OF KNOWN BREAKING CHANGES AND ONE OF THE MOST IMPORTANT CHANGES IS THE THE ADDITION OF THE FEATURE "SEARCH.IDLE.AFTER" WHICH AFFECTS THE DEFAULT "REFRESH_INTERVAL" OF 1 SECOND. THIS CHANGE IS IMPORTANT, ESPECIALLY TO INDEX HEAVY CLUSTERS THAT ARE CONSTANTLY ADDING AND UPDATING DOCUMENTS TO THE CLUSTER THAT NEED TO BE SEARCHED SHORTLY AFTER INGESTION. THIS MEANS DOCUMENTS WILL NOT ALWAYS BE AVAILABLE FOR SEARCH 1 SECOND AFTER BEING INDEXED TO THE CLUSTER IF SEARCHES GO IDLE FOR A CERTAIN PERIOD, 30 SECONDS BY DEFAULT.

- Specific Breaking Change Noted Here

In Elasticsearch, newly indexed documents need to be flushed from memory buffer to segments (disk) before they can be searched; this operation is the refresh operation. Reading freshly-written segments and getting them ready for search during refresh operations is expensive on the compute resources of the data nodes. The resulting behavior of this new feature happens when the first searches that come in after shards have been marked idle incur a latency for that data to be flushed from memory buffer to disk for search operations. When searches occur after a period of no refreshes occurring for documents to be committed to search, this will result in increased wait times for searches because the more documents that need to be flushed to disk. And when refreshes occur every second, the refreshes are allowed to run in the background while incoming search requests use whatever the latest “ready” searcher is. But when the shard goes idle, the last good searcher is considered too old to be used. So the first request(s) block to wait for the flush + reopen to happen synchronously.

- Building An Index
- Indexing Request Workflow

This latency issue is most commonly observed in major version upgrades from Amazon Elasticsearch 6.x to any version of OpenSearch 1.x. (Skipping ES 7.x). But it should also be considered in upgrades from ES 6.x to any version of Amazon Elasticsearch 7.x as the same behavior applies from 7.0 Breaking Changes.

Resolution:

When customer has upgraded to OpenSearch from Elasticsearch and they're seeing high latency, please consider checking the refresh intervals of their indices and advising them to explicitly set a refresh interval as none will be

there. And to also fine tune their refresh intervals; if search data is more important on cluster then 1s refresh interval should be set; if indexing is more important then 30s refresh interval will be a good start. It also important to tweak the "search.idle.after" setting as it will also be 30 seconds by default in ES 7.0 and above, in most cases you will want to increase this setting to ensure shards aren't marked idle.

COMMAND SYNTAX:

1) Check index settings first:

```
GET /my-index-000001/_settings
```

2) Apply "refresh_interval" settings to index:

```
PUT /my-index-000001/_settings
{
  "index" : {
    "refresh_interval" : "1s"
  }
}
```

3) Apply "search.idle.after" settings to index:

Note: Customer use-cases that benefit from the "search.idle.after" setting are:

a) Indexing logs that will only be searched when needed (low search, heavy index)

b) Having a clear separation between writing periods and searching periods (e.g. websites that do bulk indexing at night and serves more search traffic during the day)

c) Setting will more likely be increased for search heavy clusters and decreased for index heavy cluster. The default 30 seconds is too low in most cases.

```
PUT /my-index-000001/_settings
{
  "index" : {
    "search.idle.after" : "60s"
  }
}
```

Results After Applying Refresh Interval:

Other Causes of Search Latency

It's important to always keep in mind that Search latency can also be caused by compute resources. Ensure that you pay close attention to the CPU Utilization, JVMMemoryPressure and EBS Disk Metrics for Read IOPS and ReadThroughput.

CPU Utilization:

CPU Utilization can result in search latency when compute resources are under high load from traffic and threadpoolqueues are busy.
When a customer is experiencing search latency, please be sure to capture the hot threads dump as you would in any situation where high CPU Utilization is observed.
This will ensure you're able to capture the tasks and potential search tasks using CPU threadpools at that moment for your escalation to the Support Ops and Service Team.
The hot threads dump is useful and important because the tasks running at that time cannot be investigated and reviewed at a later time. The CPU hot thread dump is the snapshot of what the cluster was doing at that time to aid in RCA.

```
GET _nodes/hot_threads
```

High CPU Utilization Playbook

JVMMemoryPressure:

High JVMMemoryPressure can result in long GC Pauses that can make nodes drop in and out of the cluster.
Ensure you look for high JVM Memory Pressure and attempt to clear the cache or field data cache following the high JVM SOP's that we have.
High JVM is typically caused by expensive queries and high search traffic. It's also important to look at the shard allocation to ensure that clusters are following best practices for shard counts on instance types. High traffic clusters with too many shards per node are prone to high JVM Memory Pressure.

Troubleshooting High JVM

EBS Volumes:

Be sure to ensure that Read Metrics are not incurring IOPS Throttling and Throughput throttling at the disk level as this can result in performance bottlenecks that can also lead to search latency.
To catch this be sure to check the EBS microbursting metrics and calculate if the microbursting is exceeding the configured EBS IOPS and Throughput.

EBS Throttling Troubleshooting

Reference Cases and RCI tickets:

https://t.corp.amazon.com/V890712280/communication

https://t.corp.amazon.com/V1032231008/communication

https://command-center.support.aws.a2z.com/case-console#/cases/12533708551/correspondences

https://github.com/opensearch-project/OpenSearch/issues/9707