## ElasticSearch生产集群配置优化

### 集群主机（节点）配置相近

集群中主机最好配置相近，集群的性能符合木桶定律，即集群的查询性能是由集群中性能最差的主机决定的。

### 建议使用SSD硬盘替换传统的机械硬盘

ES对磁盘IO的性能要求比较高。

### ES内存配置不要超过32G

指定的堆内存可以是节点总内存的一半。

[https://www.elastic.co/guide/en/elasticsearch/guide/current/heap-sizing.html#compressed\_oops](https://www.elastic.co/guide/en/elasticsearch/guide/current/heap-sizing.html" \l "compressed_oops)

ES\_HEAP\_SIZE

### 集群名称和节点名称必须要配置

ES根据集群的名称组成一个集群，在生产中可以指定一个具有实际意义的集群名称。

节点名称一般指定主机名，便于管理和监控，出现故障也易于定位。

### \*使用独占的主节点和数据节点。（主要避免脑裂或负载过高）

一个集群里面出现了两个或以上的master节点，就称为脑裂。

发生脑裂的原因主要由两个：网络质量不好或master节点负载过高。

因此，在前面的优化中，使用独占的主节点和数据节点可以避免负载过高。

缓解脑裂还可以设置一个参数： discovery.zen.minimum\_master\_node。

这个参数决定了至少有多少个存活的节点，才选举产生新的master节点。

discovery.zen.minimum\_master\_node: (master\_eligible\_nodes / 2) + 1

换句话说，如果有三个主节点，那么最小主节点应该设置为（3/2）+ 1或2

discovery.zen.minimum\_master\_nodes：2

如果集群网络环境不太好， 建议将网络故障的超时时间设置的长一点。

为了保证网络，可以将es和hadoop或spark存在于同一个机架。

discovery.zen.ping\_timeout（默认3秒）

设置集群中自动发现其它节点时ping连接超时时间，默认为3秒，对于比较差的网络环境可以以高点的值来防止自动发现时出错。

### 减少swap内存交换

bootstrap.memory\_lock: true

### \*副本、分片数量的规划

分片的数量（建议3~4个）需要根据实际数据量测试，确定最合适的分片数量。

分片数过少，并发少。分片数过多，消耗的系统资源多（打开文件句柄、内存），影响性能。

一个分片最多能存储20亿条记录，根据实际性能规划。

副本一般推荐设置2~3个（大量数据情况）。

副本数量能提升数据检索的性能。

副本需要从主分片同步数据，为了避免服务器压力，副本数量不能过多。

### 集群启动数据恢复相关参数配置

集群维护后重启。比如一共5个节点，其中3个节点启动很快，2个节点启动很慢。

当3个节点启动后，另外两个节点还没启动，这个时候集群找不到没启动服务的节点上的shard，这个时候集群就会尝试做分片的重新分配。等另外两个节点启动后，集群再次对集群的分片重新分配。这样就导致了不必要的IO操作，对性能影响非常大。

gateway.recover\_after\_nodes: 3 当集群中至少启动3个节点，才开始做分片的分配。

gateway.recover\_after\_nodes

设置集群中N个节点启动时进行数据恢复，默认为1。

gateway.recover\_after\_time

设置初始化数据恢复进程的超时时间，默认是5分钟。

gateway.expected\_nodes

设置这个集群中节点的数量，默认为2，一旦这N个节点启动，就会立即进行数据恢复。

### 数据导入优化

两种方式提升性能。

（1）、在数据导入前，将副本数设置为0。

数据导入完成后，将副本数设置为正常数值。

在数据导入，如果存在副本，需要从主分片同步数据，增加ES服务器的压力。

（2）、设置index.refresh\_interval参数

index.refresh\_interval默认是1秒，刷新后文档就能被检索到。

可以将index.refresh\_interval设置为-1，不刷新。

或者将index.refresh\_interval增大，比如30秒，避免频繁刷新。

### 关闭不必要的index

open的索引的shard会加载到内存中，close不必要的index可以减少内存占用。

curl -XPOST http://node01:1 9200/myshard/\_close

### 事务刷新阈值

Elasticsearch在写入时是先写日志，即事务日志。事务日志在满足一定的阈值大小就会进行延迟提交，可以确保插入和删除操作的原子性。

可以将事务日志的刷新阈值提高，index.translog.flush\_threshold\_size默认是512M，比如设置成1g：index.translog.flush\_threshold\_size=1GB

### 索引缓存大小

增加索引缓存，可以在内存中缓存更多的数据。

indices.memory.index\_buffer\_size 默认是10%，即为堆内存的10%。一般不需要设置，如果需要设置，一定要进行性能测试。

## \*Spark整合Elasticsearch优化

### 读取性能

增加shard数量是否可以增加读取性能？

增加节点， 数据更好的跨越多个节点， 理想情况下数据跨越多个节点。

如果一个index的多个shard在同一个机器上，即仅仅是数据虚拟分区，硬件保持不变。虽然可以增加客户端读取的并行度，因为更多的shard意味着更多的任务可以同时从elasticsearch读取数据，但是从elasticsearch角度，没有实际效益，因此性能可能保持不变。

### 写性能

写入Elasticsearch是由spark的分区数量驱动。elasticsearch-hadoop检测要写入主分片的数量，并在这些shard之间分配写入。因此，可用的shard越多，写入Elasticsearch的并行度越高。

Spark跨越所有的task并行写数据。提升写性能的一个关键方面是Elasticsearch能无压力摄入数据的最大速率。这取决于很多因素（数据大小、硬件、当前负载等），因此Elasticsearch集群自身的优化也非常重要。

### 减小bulk的大小

假设有T个任务，配置为B字节和N个文档（其中d是平均文档大小），则在给定时间点批量写入请求的最大数量可以是T \* B个字节或 T \* N个文档（T \* N \* d以字节为单位）。

对于具有5个任务的工作，使用默认值（1mb或1000文档）意味着最多5mb /5000文档批量大小（散布在shard中）。如果处理时间超过1-2秒，则无需减少bulk的大小。Spark允许为每个任务配置批量写入Elasticsearch的条目数量和大小。

如果有Kafka参与，可以配置Kafka限速参数等。

### 限制写入Elasticsearch的任务数(基于Spark)

使用Spark写数据到Elasticsearch，如果是RDD或者表的join会生成多个任务，这可能导致产生大量的Task, 即导致用户计划用于Elasticsearch的任务数与实际的任务数之间的数量不成比例，有时可能高出1-2个数量级。

如果ES的集群只有几个节点，同时处理这么多的task将导致写数据过程非常缓慢。

解决：减少Task数量，减小源端的小文件，使用coalesce或者repartition参数等。