一次 ES 集群崩溃及恢复过程

一、背景介绍

生产环境上搭建了一套由 Filebeat + Kafka + Logstash + Elasticsearch + Kibana 实现的实时日志分析系统。每一台生产服务器上利用 Filebeat 收集指定的应用日志，作为生产者写入到 Kafka 中；另一端 Logstash 作为消费者消费 Kafka 中的消息，并上报到 ES 集群中，最终通过 Kibana 展示日志消息。

我们的生产集群一共七台机器专用于运行 Elasticsearch 集群，其中六台内存大小为 62G, 一台内存大小为 128G。ES 启动的 JVM 堆的大小设置为 Xmx=42G、Xms=42G。

二、问题发生

2024 年 7 月 18 日，生产环境的 Elasticsearch 集群突然无法访问，查看 ES 集群日志，发现如图 1 报错：

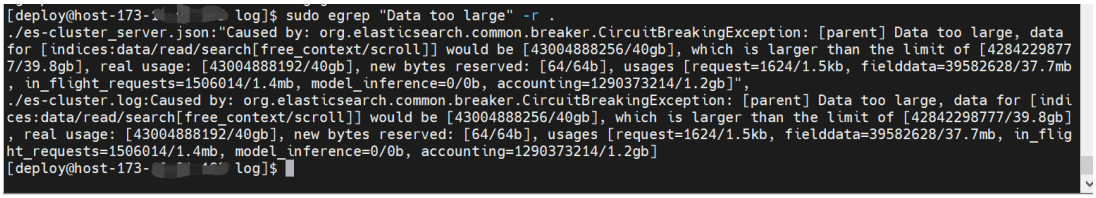

图 1

从日志信息中可以看出，这是 ES 熔断器的抛出的异常，异常信息告诉我们此时 ES 的数据量过大，超过了熔断器设置的阈值（【4300488256/40gb】），ES 无法响应请求。ES 启动参数中涉及索引缓存的两个参数的值分别如下：

indices. memory. index_buffer_size：30%

indices. fielddata. cache. size：70%

以上两个参数告诉我们，我们允许索引使用 30%的内存作为缓存，70%作为 fielddata 缓存。ES 中存在一个 fieldata 熔断器，当缓存在内存中的 field data 内存 40%后会触发熔断机制。根据这些参数值以及报错信息我们判断，ES 抛出该异常的原因是缓存到内存中的数据量过大，触发了 ES 集群的熔断机制，导致

ES 集群无法正常工作。至于缓存数据量过大的原因，我们分析有以下两点：

1. 没有对 ES 集群进行生命周期管理。我们一共配置了 23 条不同的流水线，用来收集系统日志、应用日志、Nginx 日志等信息，这些流水线每天会创建一个新的索引用来写入当天的消息。最早的索引能够追溯到 2022 年 5 月。这些索引虽然很少被检索，但由于还是 open 状态，且相关的分片也是 started 状态，因此也会占用大量内存；
2. 在崩溃发生前两天，我们发现了部分 filebeat 报错显示，由于数据过大导致无法采集的问题（如图 2），因此 ES 集群上缺失了部分日志信息，当时我们调大了 Kafka 消息队列和 Filebeat 允许的数据抓取大小。这使得原来有部分数据不会上传到 ES 集群，调整后，更多的数据能够上传，从而 ES 需要处理的请求增加，增加了集群的负载。

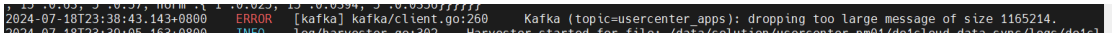

图 2

三、恢复过程

定位到问题后，首要措施是重启 ES 集群，但是重启前我们考虑了以下几个问题：

1. 此时 ES 集群无法访问，需要依次重启每个节点成功后，ES 才能恢复正常。生产环境的 ES 集群有七个节点，滚动重启每个节点至少需要 20 分钟，此时是下午上班时间，应用日志和 Nginx 日志产生的速度非常快，分分钟达到几百万条，在此期间产生的日志消息怎么办？
2. 由于 Kafka 已经在调整了 message. max. bytes 和 replica. fetch. max. bytes 两个参数后运行过一段时间，意味着如果不停止从 Kafka 中消费消息，ES 重启后可能依然会存在 Data too large 报错。
3. ES 重启集群后，需要重新分配分片，这套 ES 集群中每个索引模式每天都会根据日期创建新的索引，例如：

   system_command_log. 2024. 07. 01

   system_command_log. 2024. 07. 02

   …

这些索引并没有实施任何生命周期策略。在排查问题的过程中我们发现，这套 ES 集群保存着从 2022 年 5 月以来的所有索引。共计 30 个索引模式（system_command_log*等），17602 个索引，45580 个分片，其中包括主分片和副本分片。ES 对这些索引重新进行分配需要时间，如果在集群恢复的过程中用户报障，我们可能没有办法快速地利用 ES 排查故障。

针对以上问题，我们决定通过以下一系列的措施实施 ES 集群重启过程：

1. 停止 Logstash 向 ES 集群写入数据，先暂时让所有采集到的日志消息积压在 Kafka 消息队列中，一方面减小 ES 集群的压力，另一方面确保重启期间产生的日志消息不丢失。
2. 删除自调整了 Kafka 参数后创建的索引及数据，在 ES 集群恢复后，重新采集这部分数据。
3. 减小 ES 的 indices. memory. index_buffer_size 和 indices. fielddata. cache. size 参数的数值，防止缓存的 index 数据量过多，导致内存使用率达到熔断机制触发的阈值，并且调高熔断器的触发值。
4. 滚动重启 ES 集群节点，重启后手动优先对使用频率高的索引分配进行赋值，分配成功后，再开启对应的流水线。

通过以上操作，我们暂时顺利地恢复了 ES 集群以及部分索引的写入，此时研发人员已经可以通过 ES 查询应用日志和 Nginx 日志了。

四、优化措施

恢复了 ES 集群后，剩下的工作是优化。我们采取了以下几点优化措施：

1. 关闭 2022 年 5 月开始到 2023 年 12 月的索引，减少索引占用的内存。
2. 增加生命周期策略，设置冷、热、温阶段。索引自创建日期 30 天后自动进入冷阶段并关闭索引；设置冷热节点，将 128g 内存的机器设置为热节点；
3. 缩小 Kafka 消息队列数据抓取大小限制；
4. 降低 Logstash 流水线数据写入速度，减轻 Elasticsearch 处理请求及读写索引的压力。

经过以上一系列的措施，ES 集群各个内存节点的内存使用情况有明显改观。

图 3 为调整前的各节点内存使用情况，图 4 为调整后的内存使用情况吗，可以看到内存使用情况有明显的下降。

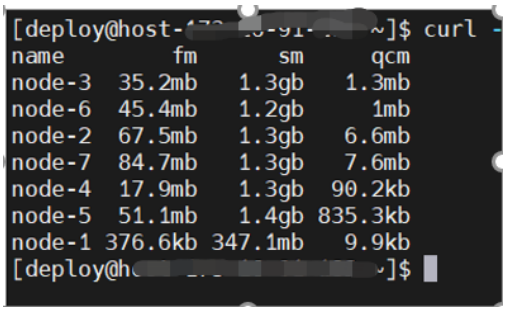

图 3

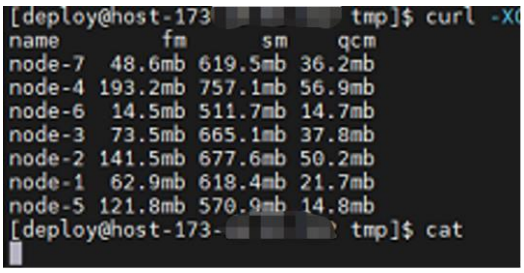

图 4

自此，我们的 ES 集群已经恢复完毕，并且运行速度也有一定的提高。

以上便是一次 ES 集群因数据量过大触发熔断机制导致 ES 集群无法访问及恢复的过程。因为这一套 ES 集群仅供运维及研发人员使用，因此没有对业务造成任何影响。通过这一次事件，我们总结了以下经验：

1. ES 集群崩溃时，首先减轻 ES 集群负载，即，减少 ES 读写请求，然后再重启 ES 集群；
2. 重启集群时滚动重启；
3. 保证集群能够使用的内存大小限制小于熔断机制触发的阈值。
4. 及时地对索引进行管理，设置合理的索引生命周期，关闭不需要的索引，减少不必要的内存使用量。

当然，ES 中需要学习的内容一定还有更多，在未来的工作中，如果遇到别的问题，将持续地更新。