ES

题目难度：★★★

知识点标签：Elastic Stack

学习时长：15分钟

题目描述

es 在数据量很大的情况下（数十亿级别）如何提高查询效率啊?

解题思路

> 需要从ES搜索优化作答
>
> es的性能优化，主要是围绕着**fileSystem cache**也可以叫做OS cache来进行；es写入数据实际上数据最终都会写入到磁盘中去，当我们搜索读取的时候，系统会将数据放入到os cache中，而es严重依赖于这个os cache，如果我们给机器的内存足够多，在es里存的书库里昂小于内存容量，那么搜索的效率是非常高的，

性能优化的杀手锏——filesystem cache

你往 es 里写的数据，实际上都写到磁盘文件里去了，查询的时候，操作系统会将磁盘文件里的数据自动缓存到 filesystem cache 里面去。

es 的搜索引擎严重依赖于底层的 filesystem cache，你如果给 filesystem cache 更多的内存，尽量让内存可以容纳所有的 idx segment file 索引数据文件，那么你搜索的时候就基本都是走内存的，性能会非常高。

减少字段

如果我们的表里有很多的字段，而我们只需要往es库里写入我们需要检索的那几个字段就可以了，对于其他的字段我们可以存到mysql或者说其他的比如Hbase中，hbase的特点是适用于海量数据的在线存储，就是对hbase可以写入海量数据，不要做复杂的搜索，就是做很简单的一些根据id或者范围进行查询的这么一个操作就可以了，从es中根据检索的字段去搜索，拿到的结果可能就十几个doc id，然后根据doc id到hbase里去查询每个doc id对应的完整的数据，给查出来，再返回给前端。简单地说就是：elastcisearch减少数据量仅仅放要用于搜索的几个关键字段即可，尽量写入es的数据量跟es机器的filesystem cache是差不多的就可以了；其他不用来检索的数据放hbase里，或者mysql。

数据预热

如果说我们按照方案一的方法做了之后，效率还是不行，存的数据量还是超过os cache的空间，那么我们就可以吧一些比较热门的数据，比如在电商系统中，像一些热门的商品，我们可以在后台单独的写一个子系统，每隔一段时间，我们就访问一下，然数据进入到os cache中，这样用户来访问的时候就访问到的是os cache中的数据，就比较快。

冷热分离

elasticsearch 可以做类似于 mysql 的水平拆分，就是说将大量的访问很少、频率很低的数据，单独写一个索引，然后将访问很频繁的热数据单独写一个索引。最好是将冷数据写入一个索引中，然后热数据写入另外一个索引中，这样可以确保热数据在被预热之后，尽量都让他们留在 filesystem os cache 里，别让冷数据给冲刷掉。

假设你有 6 台机器，2 个索引，一个放冷数据，一个放热数据，每个索引 3 个 shard。3 台机器放热数据 index，另外 3 台机器放冷数据 index。然后这样的话，你大量的时间是在访问热数据 index，热数据可能就占总数据量的 10%，此时数据量很少，几乎全都保留在 filesystem cache 里面了，就可以确保热数据的访问性能是很高的。但是对于冷数据而言，是在别的 index 里的，跟热数据 index 不在相同的机器上，大家互相之间都没什么联系了。如果有人访问冷数据，可能大量数据是在磁盘上的，此时性能差点，就 10% 的人去访问冷数据，90% 的人在访问热数据，也无所谓了。

document 模型设计

对于 MySQL，我们经常有一些复杂的关联查询。在 es 里该怎么玩儿，es 里面的复杂的关联查询尽量别用，一旦用了性能一般都不太好。

最好是先在 Java 系统里就完成关联，将关联好的数据直接写入 es 中。搜索的时候，就不需要利用 es 的搜索语法来完成 join 之类的关联搜索了。

document 模型设计是非常重要的，很多操作，不要在搜索的时候才想去执行各种复杂的乱七八糟的操作。es 能支持的操作就那么多，不要考虑用 es 做一些它不好操作的事情。如果真的有那种操作，尽量在 document 模型设计的时候，写入的时候就完成。另外对于一些太复杂的操作，比如 join/nested/parent-child 搜索都要尽量避免，性能都很差的。

分页性能优化

不允许深度分页（默认深度分页性能很差）

类似于 app 里的推荐商品不断下拉出来一页一页的，你可以用 `scroll api` scroll 会一次性给你生成所有数据的一个快照然后每次滑动向后翻页就是通过游标scroll_id `移动，获取下一页下一页这样子，性能会比上面说的那种分页性能要高很多很多，基本上都是毫秒级的。初始化时必须指定` scroll` 参数，告诉 es 要保存此次搜索的上下文多长时间。你需要确保用户不会持续不断翻页翻几个小时，否则可能因为超时而失败。

除了用 `scroll api` ，你也可以用 `search_after` 来做， `search_after` 的思想是使用前一页的结果来帮助检索下一页的数据，显然，这种方式也不允许你随意翻页，你只能一页页往后翻。初始化时，需要使用一个唯一值的字段作为 sort 字段。