

返回列表

[总结型] 如何提高ElasticSearch在数据量很大的情况下（数十亿级别）的查询效率？ [复制链接]

📄 ↺ ↻

头像

BGnv5 发表于 2019-8-3 14:32:11 | 只看该作者 | 只看大图 ▶

楼主 电梯直达

本帖最后由 BGnv5 于 2019-8-3 14:49 编辑

问题导读：

- 1.ElasticSearch在数十亿级别数据下，如何提高查询效率？
2. 一些复杂的关联查询是直接写入 es 中还是在Java 系统里完成更好？
- 3.为什么说es 的分页是比较坑的呢以及如何解决？

面试题

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？

面试官心理分析

这个问题是肯定要问的，说白了，就是看你有没有实际干过 es，因为啥？其实 es 性能并没有你想象中那么好的。很多时候数据量大了，特别是有几亿条数据的时候，可能你会懵逼的发现，跑个搜索怎么一下 5~10s，坑爹了。第一次搜索的时候，是5~10s，后面反而就快了，可能就几百毫秒。

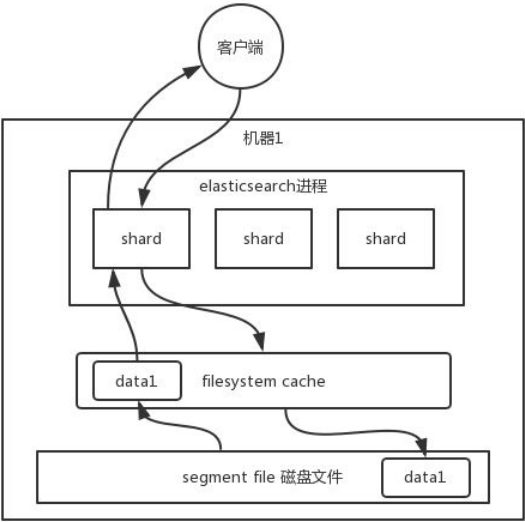
你就很懵，每个用户第一次访问都会比较慢，比较卡么？所以你要是没玩儿过 es，或者就是自己玩玩儿 demo，被问到这个问题容易懵逼，显示出你对 es 确实玩儿的不怎么样？

面试题剖析

说实话，es 性能优化是没有什么银弹的，啥意思呢？就是不要期待着随手调一个参数，就可以万能的应对所有的性能慢的场景。也许有的场景是你换个参数，或者调整一下语法，就可以搞定，但是绝对不是所有场景都可以这样。

性能优化的杀手锏——filesystem cache

你往 es 里写的的数据，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。



性能差距究竟可以有多大？我们之前很多的测试和压测，如果走磁盘一般肯定上秒，搜索性能绝对是秒级别的，1秒、5秒、10秒。但如果是走 **filesystem cache**，是走纯内存的，那么一般来说性能比走磁盘要高一个数量级，基本上就是毫秒级的，从几毫秒到几百毫秒不等。

这里有个真实的案例。某个公司 **es** 节点有 **3** 台机器，每台机器看起来内存很多，**64G**，总内存就是 **64 * 3 = 192G**。每台机器给 **es jvm heap** 是 **32G**，那么剩下来留给 **filesystem cache** 的就是每台机器才 **32G**，总共集群里给 **filesystem cache** 的就是 **32 * 3 = 96G** 内存。而此时，整个磁盘上索引数据文件，在 **3** 台机器上一共占用了 **1T** 的磁盘容量，**es** 数据量是 **1T**，那么每台机器的数据量是 **300G**。这样性能好吗？**filesystem cache** 的内存才 **100G**，十分之一的数据可以放内存，其他的都在磁盘，然后你执行搜索操作，大部分操作都是走磁盘，性能肯定差。

归根结底，你要让 **es** 性能要好，最佳的情况下，就是你的机器的内存，至少可以容纳你的总数据量的一半。

根据我们自己的生产环境实践经验，最佳的情况下，是仅仅在 **es** 中就存少量的数据，就是你要用来搜索的那些索引，如果内存留给 **filesystem cache** 的是 **100G**，那么你就将索引数据控制在 **100G** 以内，这样的话，你的数据几乎全部走内存来搜索，性能非常之高，一般可以在 **1** 秒以内。

比如说你现在有一行数据。**id,name,age 30** 个字段。但是你现在在搜索，只需要根据 **id,name,age** 三个字段来搜索。如果你傻乎乎往 **es** 里写入一行数据所有的字段，就会导致说 **90%** 的数据是不用来搜索的，结果硬是占据了 **es** 机器上的 **filesystem cache** 的空间，单条数据的数据量越大，就会导致 **filesystem cahce** 能缓存的数据就越少。其实，仅仅写入 **es** 中要用来检索的少数几个字段就可以了，比如说就写入 **es id,name,age** 三个字段，然后你可以把其他的字段数据存在 **mysql/hbase** 里，我们一般是建议用 **es + hbase** 这么一个架构。

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，但是不要做复杂的搜索，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 **name** 和 **age** 去搜索，拿到的结果可能就 **20** 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。

写入 **es** 的数据最好小于等于，或者是略微大于 **es** 的 **filesystem cache** 的内存容量。然后你从 **es** 检索可能就花费 **20ms**，然后再根据 **es** 返回的 **id** 去 **hbase** 里查询，查 **20** 条数据，可能也就耗费个 **30ms**，可能你原来那么玩儿，**1T** 数据都放 **es**，会每次查询都是 **5~10s**，现在可能性能就会很高，每次查询就是 **50ms**。

数据预热

假如说，哪怕是你就按照上述的方案去做了，**es** 集群中每个机器写入的数据量还是超过了 **filesystem cache** 一倍，比如说你写入一台机器 **60G** 数据，结果 **filesystem cache** 就 **30G**，还是有 **30G** 数据留在了磁盘上。

其实可以做数据预热。

举个例子，拿微博来说，你可以把一些大V，平时看的人很多的数据，你自己提前后台搞个系统，每隔一会儿，自己的后台系统去搜索一下热数据，刷到 **filesystem cache** 里去，后面用户实际上来看这个热数据的时候，他们就是直接从内存里搜索了，很快。

或者是电商，你可以将平时查看最多的一些商品，比如说 **iphone 8**，热数据提前后台搞个程序，每隔 **1** 分钟自己主动访问一次，刷到 **filesystem cache** 里去。

对于那些你觉得比较热的、经常会有人访问的数据，最好做一个专门的缓存预热子系统，就是对热数据每隔一段时间，就提前访问一下，让数据进入 **filesystem cache** 里面去。这样下次别人访问的时候，性能一定会好很多。

冷热分离

es 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。

你看，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.....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

首页论坛|社区

导读

博客

专题

浏览

阅读推荐

群组

分类搜索

请输

es的分页是较坑的，为啥呢？举个例子吧，假如你每页是 10 条数据，你现在要查询第 100 页，实际上是会把每个 shard 上存储的前 1000 条数据都查到一个协调节点上，如果你有个 5 个 shard，那么就有 5000 条数据，接着协调节点对这 5000 条数据进行一些合并、处理，再获取到最终第 100 页的 10 条数据。

分布式的，你要查第 100 页的 10 条数据，不可能说从 5 个 shard，每个 shard 就查 2 条数据，最后到协调节点合并成 10 条数据吧？你必须得从每个 shard 都查 1000 条数据过来，然后根据你的需求进行排序、筛选等等操作，最后再次分页，拿到里面第 100 页的数据。你翻页的时候，翻的越深，每个 shard 返回的数据就越多，而且协调节点处理的时间越长，非常坑爹。所以用 es 做分页的时候，你会发现越翻到后面，就越是慢。

我们之前也是遇到过这个问题，用 es 作分页，前几页就几十毫秒，翻到 10 页或者几十页的时候，基本上就要 5~10 秒才能查出来一页数据了。

有什么解决方案吗？

不允许深度分页（默认深度分页性能很差）

跟产品经理说，你系统不允许翻那么深的页，默认翻的越深，性能就越差。

类似于 app 里的推荐商品不断下拉出来一页一页的

类似于微博中，下拉刷微博，刷出来一页一页的，你可以用 scroll api，关于如何使用，自行上网搜索。

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。

但是，唯一的一点就是，这个适合于那种类似微博下拉翻页的，不能随意跳到任何一页的场景。也就是说，你不能先进入第 10 页，然后去第 120 页，然后又回到第 58 页，不能随意乱跳页。所以现在很多产品，都是不允许你随意翻页的，app，也有一些网站，做的就是你只能往下拉，一页一页的翻。

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。



作者：节操泛滥的程序员
原文链接：<https://zhuanlan.zhihu.com/p/60458049>

分享到： QQ好友和群

收藏6 转播 分享 淘帖 赞 踩 分享到微信

回复

使用道具

恋枫缩影 发表于 2019-8-3 18:24:29 | 只看该作者

收藏学习了

点评

回复

支持

反对

首页

论坛 | 社区

导读

博客

专题

浏览

阅读推荐

群组

分类搜索

请输

使用道具

举报



美丽天空

 发表于 2019-8-5 13:33:19 | 只看该作者

板凳

感谢分享了

点评

回复


支持

反对

使用道具

举报

返回列表



高级模式

☐ 回帖并转播

☐ 回帖后跳转到最后一页

本版积分规则