1、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？

2、面试官心里分析

问这个问题，是肯定的，说白了，就是看你有没有实际干过es，因为啥？es说白了其实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秒~10秒，坑爹了。第一次搜索的时候，是5~10秒，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

3、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。

一块一块来分析吧

在这个海量数据的场景下，如何提升es搜索的性能，也是我们之前生产环境实践经验所得

（1）性能优化的杀手锏——filesystem cache

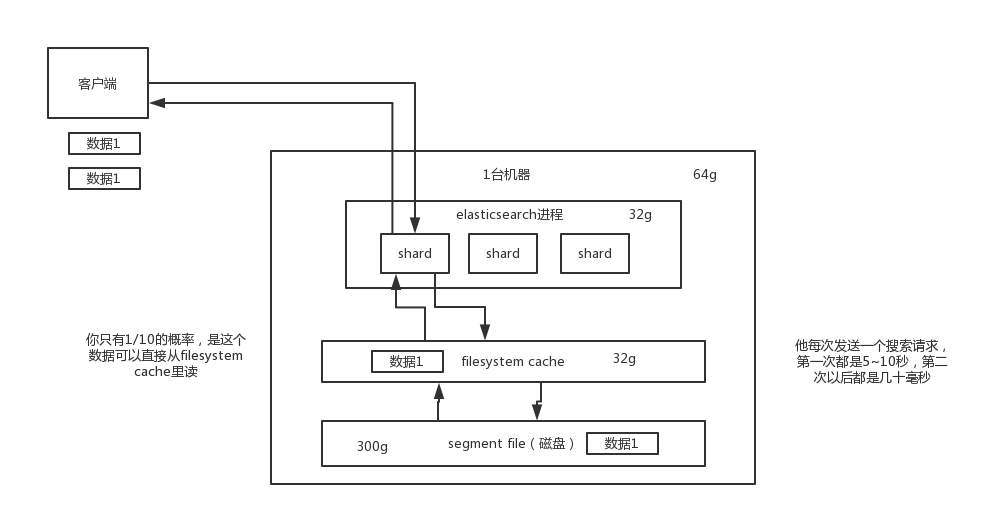
os cache，操作系统的缓存

你往es里写的数据，实际上都写到磁盘文件里去了，磁盘文件里的数据操作系统会自动将里面的数据缓存到os 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indx 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。

性能差距可以有大，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。但是如果是走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。

之前有个学员，一直在问我，说他的搜索性能，聚合性能，倒排索引，正排索引，磁盘文件，十几秒。。。。



学员的真实案例

比如说，你，es节点有3台机器，每台机器，看起来内存很多，64G，总内存，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

我就问他，ok，那么就是你往es集群里写入的数据有多少数据量？

如果你此时，你整个，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，你的es数据量是1t，每台机器的数据量是300g

你觉得你的性能能好吗？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

当时他们的情况就是这样子，es在测试，弄了3台机器，自己觉得还不错，64G内存的物理机。自以为可以容纳1T的数据量。

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

比如说，你一共要在es中存储1T的数据，那么你的多台机器留个filesystem cache的内存加起来综合，至少要到512G，至少半数的情况下，搜索是走内存的，性能一般可以到几秒钟，2秒，3秒，5秒

如果最佳的情况下，我们自己的生产环境实践经验，所以说我们当时的策略，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，内存留给filesystem cache的，就100G，那么你就控制在100gb以内，相当于是，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

但是你现在搜索，只需要根据id name age三个字段来搜索

如果你傻乎乎的往es里写入一行数据所有的字段，就会导致说70%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，单挑数据的数据量越大，就会导致filesystem cahce能缓存的数据就越少

仅仅只是写入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三个字段就可以了，然后你可以把其他的字段数据存在mysql里面，我们一般是建议用es + hbase的这么一个架构。

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，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。

你最好是写入es的数据小于等于，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5~10秒，现在可能性能就会很高，每次查询就是50ms。

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。

所以之前有些学员也是问，我也是跟他们说，尽量在es里，就存储必须用来搜索的数据，比如说你现在有一份数据，有100个字段，其实用来搜索的只有10个字段，建议是将10个字段的数据，存入es，剩下90个字段的数据，可以放mysql，hadoop hbase，都可以

这样的话，es数据量很少，10个字段的数据，都可以放内存，就用来搜索，搜索出来一些id，通过id去mysql，hbase里面去查询明细的数据

（2）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。

举个例子，就比如说，微博，你可以把一些大v，平时看的人很多的数据给提前你自己后台搞个系统，每隔一会儿，你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。

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，你就提前访问一下，让数据进入filesystem cache里面去。这样期待下次别人访问的时候，一定性能会好一些。

（3）冷热分离

关于es性能优化，数据拆分，我之前说将大量不搜索的字段，拆分到别的存储中去，这个就是类似于后面我最后要讲的mysql分库分表的垂直拆分。

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

你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，你大量的时候是在访问热数据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都不再相同的机器上，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。

（4）document模型设计

有不少同学问我，mysql，有两张表

订单表：id order\_code total\_price

1 测试订单 5000

订单条目表：id order\_id goods\_id purchase\_count price

1 1 1 2 2000

2 1 2 5 200

我在mysql里，都是select \* from order join order\_item on order.id=order\_item.order\_id where order.id=1

1 测试订单 5000 1 1 1 2 2000

1 测试订单 5000 2 1 2 5 200

在es里该怎么玩儿，es里面的复杂的关联查询，复杂的查询语法，尽量别用，一旦用了性能一般都不太好

设计es里的数据模型

写入es的时候，搞成两个索引，order索引，orderItem索引

order索引，里面就包含id order\_code total\_price

orderItem索引，里面写入进去的时候，就完成join操作，id order\_code total\_price id order\_id goods\_id purchase\_count price

写入es的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的搜索语法去完成join来搜索了

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能支持的操作就是那么多，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。

很多同学在问我，很多复杂的乱七八糟的一些操作，如何执行

两个思路，在搜索/查询的时候，要执行一些业务强相关的特别复杂的操作：

1）在写入数据的时候，就设计好模型，加几个字段，把处理好的数据写入加的字段里面

2）自己用java程序封装，es能做的，用es来做，搜索出来的数据，在java程序里面去做，比如说我们，基于es，用java封装一些特别复杂的操作

（5）分页性能优化

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，你是不可能说从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？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页之后，几十页的时候，基本上就要5~10秒才能查出来一页数据了

1）不允许深度分页/默认深度分页性能很惨

你系统不允许他翻那么深的页，pm，默认翻的越深，性能就越差

2）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，自己百度

scroll会一次性给你生成所有数据的一个快照，然后每次翻页就是通过游标移动，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也高很多很多

针对这个问题，你可以考虑用scroll来进行处理，scroll的原理实际上是保留一个数据快照，然后在一定时间内，你如果不断的滑动往后翻页的时候，类似于你现在在浏览微博，不断往下刷新翻页。那么就用scroll不断通过游标获取下一页数据，这个性能是很高的，比es实际翻页要好的多的多。

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。同时这个scroll是要保留一段时间内的数据快照的，你需要确保用户不会持续不断翻页翻几个小时。

无论翻多少页，性能基本上都是毫秒级的

因为scroll api是只能一页一页往后翻的，是不能说，先进入第10页，然后去120页，回到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