

京东到家订单中心 Elasticsearch 演进历程

张sir 京东技术 2018-12-03

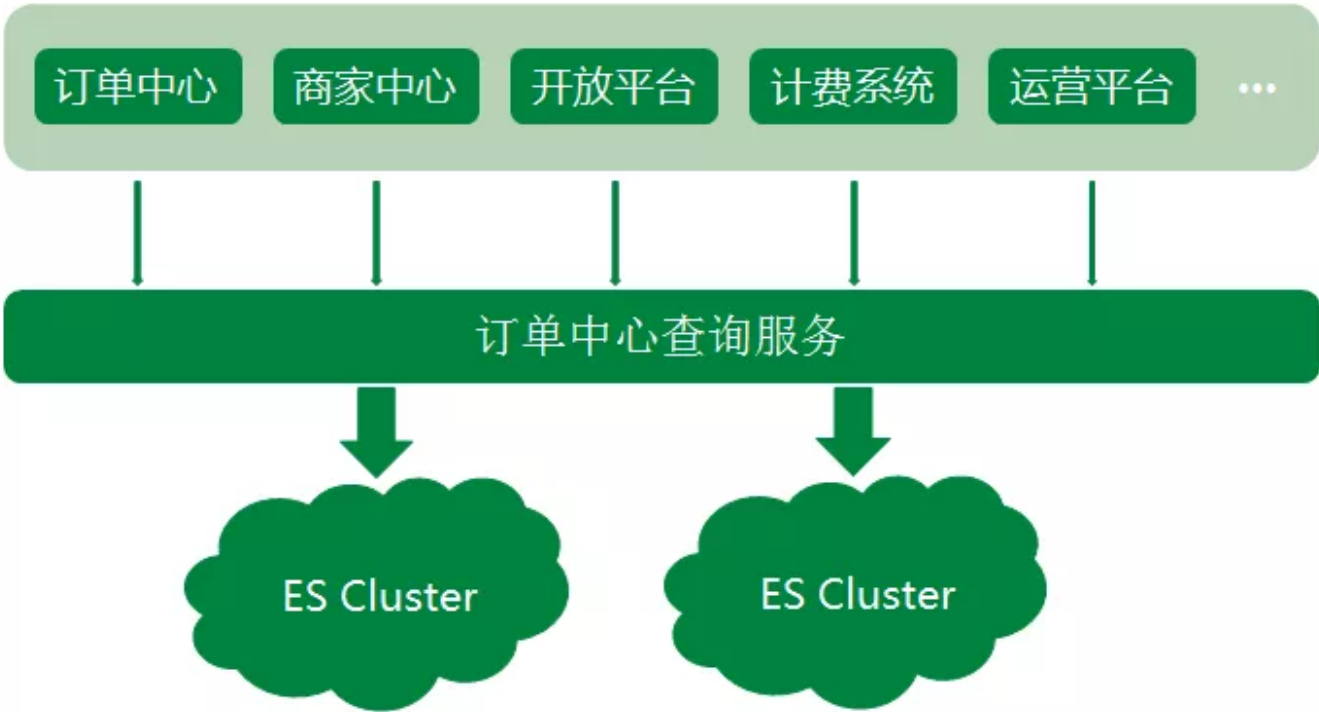
○ 来这里找志同道合的小伙伴！

作者

张sir，京东到家研发工程师，主要负责订单中心、商家中心、计费系统等系统。

背景

京东到家订单中心系统业务中，无论是外部商家的订单生产，或是内部上下游系统的依赖，订单查询的调用量都非常大，造成了订单数据读多写少的情况。京东到家的订单数据存储在Mysql中，但显然只通过DB来支撑大量的查询是不可取的，同时对于一些复杂的查询，Mysql支持得不够友好，所以订单中心系统使用了Elasticsearch来承载订单查询的主要压力。



Elasticsearch 作为一款功能强大的分布式搜索引擎，支持近实时的存储、搜索数据，在京东到家订单系统中发挥着巨大作用，目前订单中心ES集群存储数据量达到10亿个文档，日均查询量达到5亿。随着京东到家近几年业务的快速发展，订单中心ES架设方案也不断演进，发展至今ES集群架

设是一套实时互备方案，很好的保障了ES集群读写的稳定性，下面就给大家介绍一下这个历程以及遇到的一些坑。

ES集群架设演进历程

1、初始阶段

订单中心ES初始阶段好如一张白纸，架设方案基本没有，很多配置都是保持集群默认配置。整个集群部署在集团的弹性云上，ES集群的节点以及机器部署都比较混乱。同时按照集群维度来看，一个ES集群会有单点问题，显然对于订单中心业务来说也是不被允许的。

2、集群隔离阶段

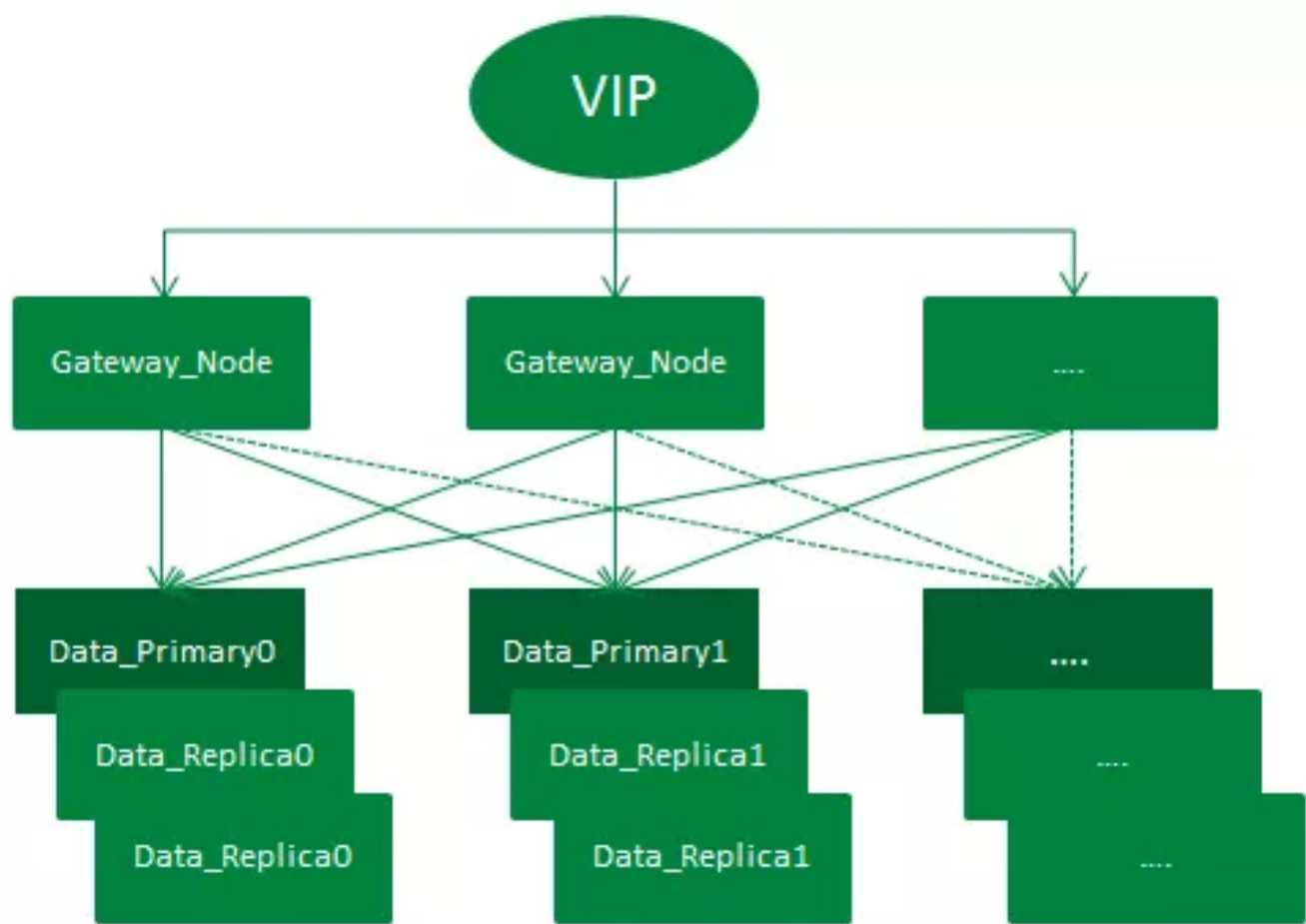
和很多业务一样，ES集群采用的混布的方式。但由于订单中心ES存储的是线上订单数据，偶尔会发生混布集群抢占系统大量资源，导致整个订单中心ES服务异常的情况。

显然任何影响到订单查询稳定性都是无法容忍的，所以针对于这个情况，先是对订单中心ES所在的弹性云，迁出那些系统资源抢占很高的集群节点，ES集群状况稍有好转。但随着集群数据不断增加，弹性云配置已经不太能满足ES集群，且为了完全的物理隔离，最终干脆将订单中心ES集群部署到高配置的物理机上，ES集群性能又得到提升。

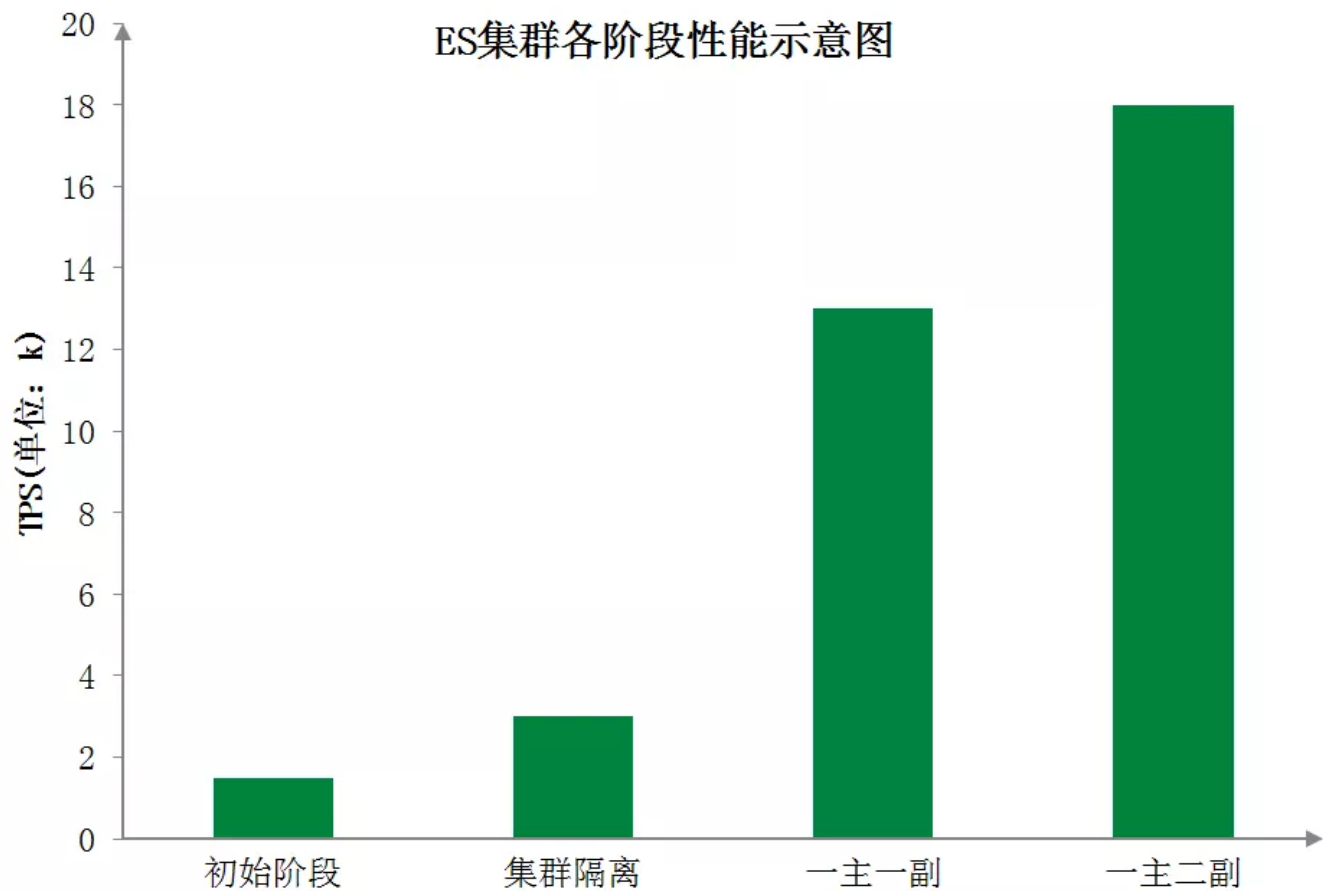
3、节点副本调优阶段

ES的性能跟硬件资源有很大关系，当ES集群单独部署到物理机器上时，集群内部的节点并不是独占整台物理机资源，在集群运行的时候同一物理机上的节点仍会出现资源抢占的问题。所以在这种情况下，为了让ES单个节点能够使用最大程度的机器资源，采用每个ES节点部署在单独一台物理机上方式。

但紧接着，问题又来了，如果单个节点出现瓶颈了呢？我们应该怎么再优化呢？ES查询的原理，当请求打到某号分片的时候，如果没有指定分片类型（preference参数）查询，请求会负载到对应分片号的各个节点上。而集群默认副本配置是一主一副，针对于此，我们想到了扩容副本的方式，由默认的一主一副变为一主二副，同时增加相应物理机。



如上图，订单中心ES集群架设示意图。整个架设方式通过VIP来负载均衡外部请求，第一层 gateway节点实质为ES中client node，相当于一个智能负载均衡器，充当着分发请求的角色。第二层为data node，负责存储数据以及执行数据的相关操作。整个集群有一套主分片，二套副分片（一主二副），从网关节点转发过来的请求，会在打到数据节点之前通过轮询的方式进行均衡。集群增加一套副本并扩容机器的方式，增加了集群吞吐量，从而提升了整个集群查询性能。下图为订单中心ES集群各阶段性能示意图，直观的展示了各阶段优化后ES集群性能的显著提升。



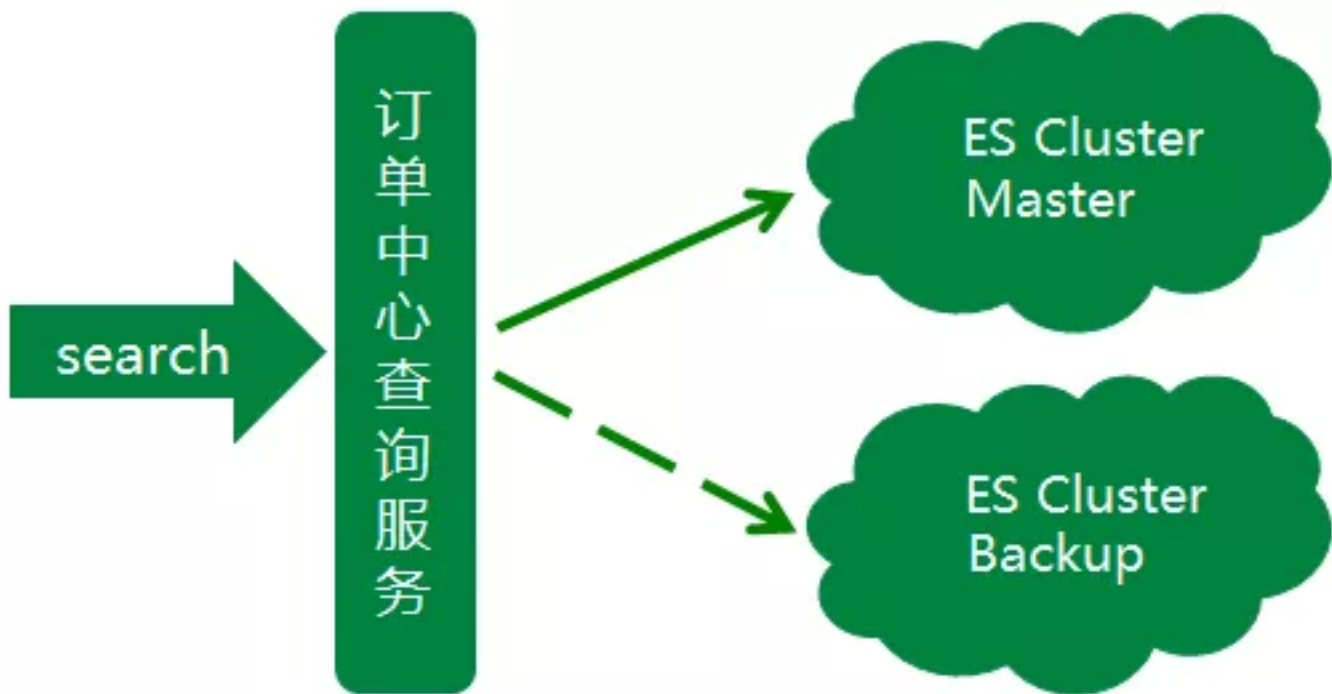
当然分片数量和分片副本数量并不是越多越好，在此阶段中，对选择适当的分片数量做了进一步探索。分片数可以理解为Mysql中的分库分表，而当前订单中心ES查询主要分为两类：单ID查询以及分页查询。分片数越大，集群横向扩容规模也更大，根据分片路由的单ID查询吞吐量也能大大提升，但对于聚合的分页查询性能则将降低。分片数越小，集群横向扩容规模更小，单ID的查询性能也将下降，但对于分页查询，性能将会得到提升。所以如何均衡分片数量和现有查询业务，我们做了很多次调整压测，最终选择了集群性能较好的分片数。

4、主从集群调整阶段

到此，订单中心的ES集群已经初具规模，但由于订单中心业务时效性要求高，对于ES查询稳定性要求也高，如果集群中有节点发生异常，查询服务会受到影响，从而影响到整个订单生产流程。显而易见这种异常情况是致命，所以为了应对这种情况，我们初步设想是增加一个备用集群，当主集群发生异常时，可以实时的将查询流量降级到备用集群。

那备用集群应该怎么来搭？主备之间数据如何同步？备用集群应该存储什么样的数据？考虑到ES集群暂时没有很好的主备方案，同时为了更好的控制ES数据写入，我们采用业务双写的方式来搭设主备集群。每次业务操作需要写入ES数据时，同步的写入主集群数据，然后异步的写入备集群数据。同时由于大部分ES查询的流量都来源于近几天的订单，且订单中心数据库数据已有一套归档机制，将指定天数之前已经关闭的订单转移到历史订单库。

所以归档机制中增加删除备集群文档的逻辑，让新搭建的备集群存储的订单数据与订单中心线上数据库中的数据量保持一致。同时使用ZK在查询服务中做了流量控制开关，保证查询流量能够实时的降级到备集群。在此，订单中心主从集群完成，ES查询服务稳定性大大提升。



5、现今：实时互备双集群阶段

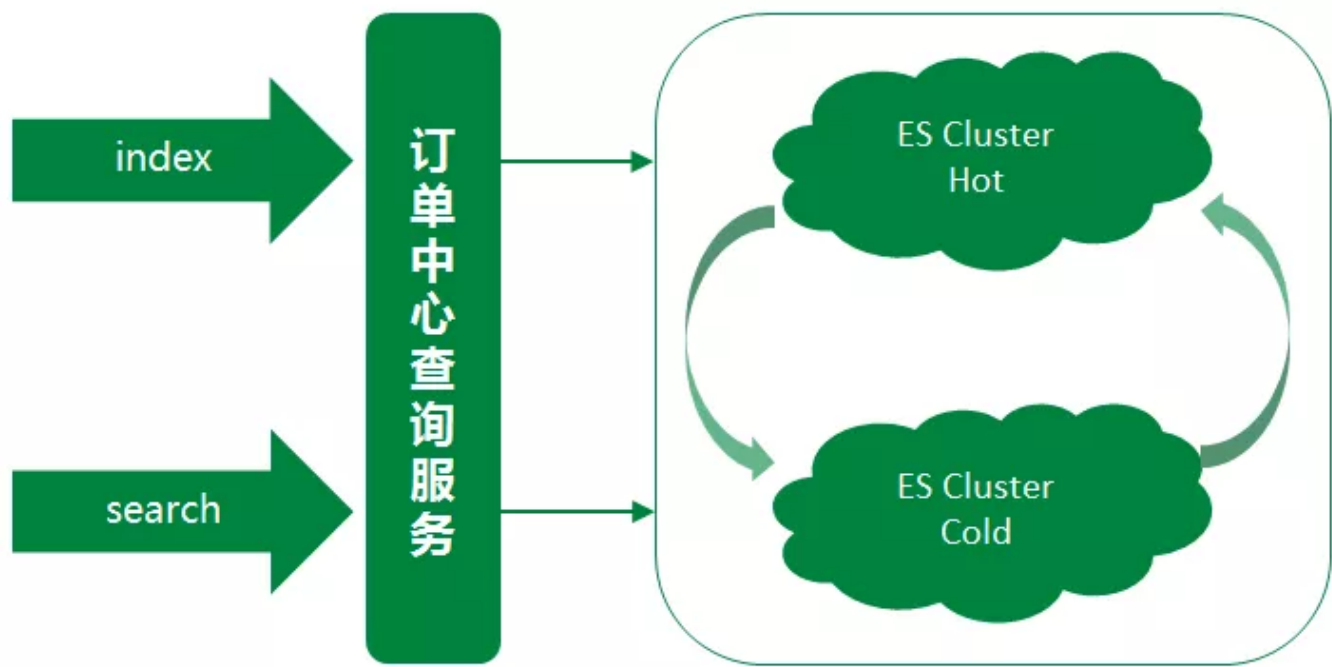
期间由于主集群ES版本是较低的1.7，而现今ES稳定版本都以及迭代到6.x，新版本的ES不仅性能方面优化很大，更提供了一些新的好用的功能，所以我们对主集群进行了一次版本升级，直接从原来的1.7升级到6.x版本。集群升级的过程繁琐而漫长，不但需要保证线上业务无任何影响，平滑无感知升级，同时由于ES集群暂不支持从1.7到6.x跨越多个版本的数据迁移，所以需要通过重建索引的方式来升级主集群，具体升级过程就不在此赘述了。

主集群升级的时候必不可免的会发生不可用的情况，但对于订单中心ES查询服务，这种情况是不允许的。所以在升级的阶段中，备集群暂时顶上充当主集群，来支撑所有的线上ES查询，保证升级过程不影响正常线上服务。同时针对于线上业务，我们对两个集群做了重新的规划定义，承担的线上查询流量也做了重新的划分。

备集群存储的是线上近几天的热点数据，数据规模远小于主集群，大约是主集群文档数的十分之一左右。集群数据量小，在相同的集群部署规模下，备集群的性能要优于主集群。然而在线上真实场景中，线上大部分查询流量也来源于热点数据，所以用备集群来承载这些热点数据的查询，而备集群也慢慢演变成一个热数据集群。之前的主集群存储的是全量数据，用该集群来支撑剩余较小部分

的查询流量，这部分查询主要是需要搜索全量订单的特殊场景查询以及订单中心系统内部查询等，而主集群也慢慢演变成一个冷数据集群。

同时备集群增加一键降级到主集群的功能，两个集群地位同等重要，但都可以各自降级到另一个集群。双写策略也优化为：假设有A B集群，正常同步方式写主（A集群）异步方式写备（B集群）。A集群发生异常时，同步写B集群（主），异步写A集群（备）。



ES订单数据的同步方案

Mysql数据同步到ES中，大致总结可以分为两种方案：

方案1：监听mysql的binlog，分析binlog将数据同步到ES集群中

优点：业务与ES数据耦合度低，业务逻辑中不需要关心ES数据的写入。

缺点：binglog模式只能使用ROW模式，且引入了新的同步服务，增加了开发量以及维护成本，也增大了ES同步的风险。

方案2：直接通过ES API将数据写入到ES集群中

优点：简洁明了，能够灵活的控制数据的写入

缺点：与业务耦合严重，强依赖于业务系统的写入方式

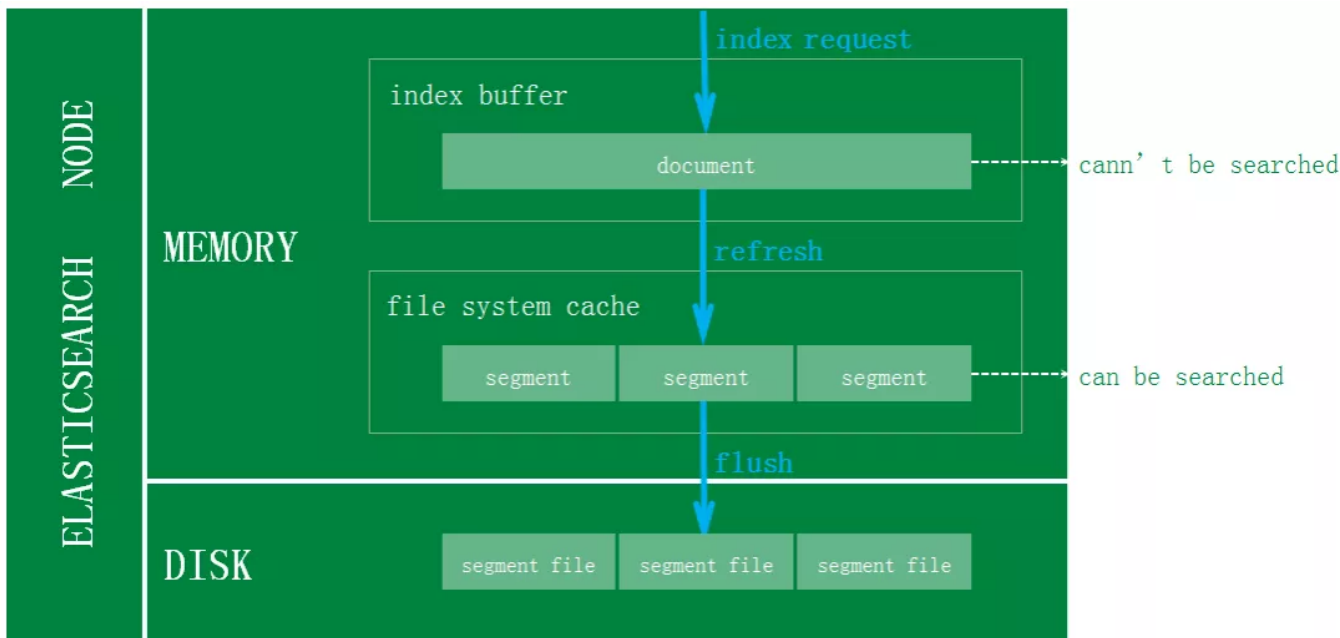
考虑到订单系统ES服务的业务特殊性，对于订单数据的实时性较高，显然监听binlog的方式相当于异步同步，有可能会产生较大的延时性。且方案1实质上跟方案2类似，但又引入了新的系统，维护成本也增高。所以订单中心ES采用了直接通过ES API写入订单数据的方式，该方式简洁灵活，能够很好的满足订单中心数据同步到ES的需求。

由于ES订单数据的同步采用的是在业务中写入的方式，当新建或更新文档发生异常时，如果重试势必会影响业务正常操作的响应时间。所以每次业务操作只更新一次ES，如果发生错误或者异常，在数据库中插入一条补救任务，有worker任务会实时的扫这些数据，以数据库订单数据为基准来再次更新ES数据。通过此种补偿机制，来保证ES数据与数据库订单数据的最终一致性。

遇到的一些坑

1、实时性要求高的查询走db

对于ES写入机制的有了解的可能会知道，新增的文档会被收集到indexing buffer，然后写入到文件系统缓存中，到了文件系统缓存中就可以像其他的文件一样被索引到。然而默认情况文档从index buffer到文件系统缓存（即refresh操作）是每秒分片自动刷新，所以这就是我们说ES是近实时搜索而非实时的原因：文档的变化并不是立即对搜索可见，但会在一秒之内变为可见。当前订单系统ES采用的是默认refresh配置，故对于那些订单数据实时性比较高的业务，直接走数据库查询，保证数据的准确性。



2、避免深分页查询

ES集群的分页查询支持from和size参数，查询的时候每个分片必须构造一个长度为from+size的优先队列，然后回传到网关节点，网关节点再对这些优先队列进行排序找到正确的size个文档。假设在一个有6个主分片的索引中，from为10000，size为10，每个分片必须产生10010个结果，在网关节点中汇聚合并60060个结果，最终找到符合要求的10个文档。由此可见，当from足够大的时候，就算不发生OOM，也会影响到CPU和带宽等，从而影响到整个集群的性能。所以应该避免深分页查询，尽量不去使用。

3、FieldData与Doc Values

Fielddata：线上查询出现偶尔超时的情况，通过调试查询语句，定位到是跟排序有关系。排序在es 1.x版本使用的是fielddata 结构，fielddata占用的是jvm heap内存，jvm内存是有限，对于fielddata cache会设定一个阈值。如果空间不足时，使用最久未使用（LRU）算法移除fielddata，同时加载新的fielddata cache，加载的过程需要消耗系统资源，且耗时很大。所以导致这个查询的响应时间暴涨，甚至影响整个集群的性能。针对于这种问题，解决的方式是采用doc values。

Doc Values：Doc Values是一种列式的数据存储结构，跟fielddata很类似，但其存储位置是在Lucene文件中，即不会占用JVM heap。随着ES版本的迭代，doc values比fielddata更加稳定，doc values在2.x起为默认设置。

总结


架构的快速迭代源于业务的快速发展，正是由于近几年到家业务的高速发展，订单中心的架构也不断优化升级。而架构方案没有最好的，只有最合适的，相信再过几年，订单中心的架构又将是另一个面貌，但吞吐量更大，性能更好，稳定性更强，将是订单中心系统永远的追求。

-----END-----

下面的内容同样精彩

点击图片即可阅读





京东技术
---关注技术的公众号

长按识别二维码关注

