京东日志平台es运维经验分享及源码改造实践

elastic

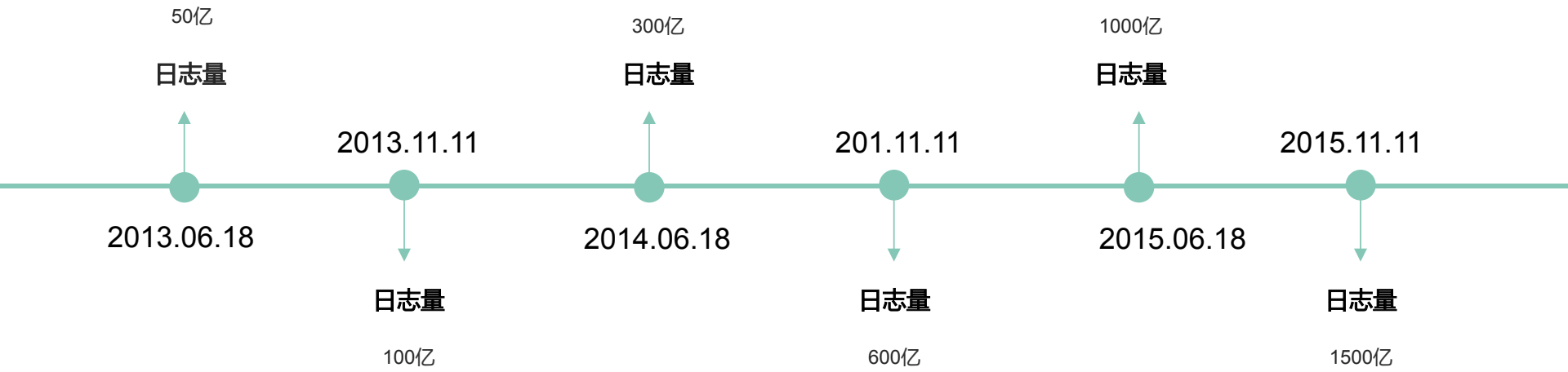
vivo 移动互联网

目录

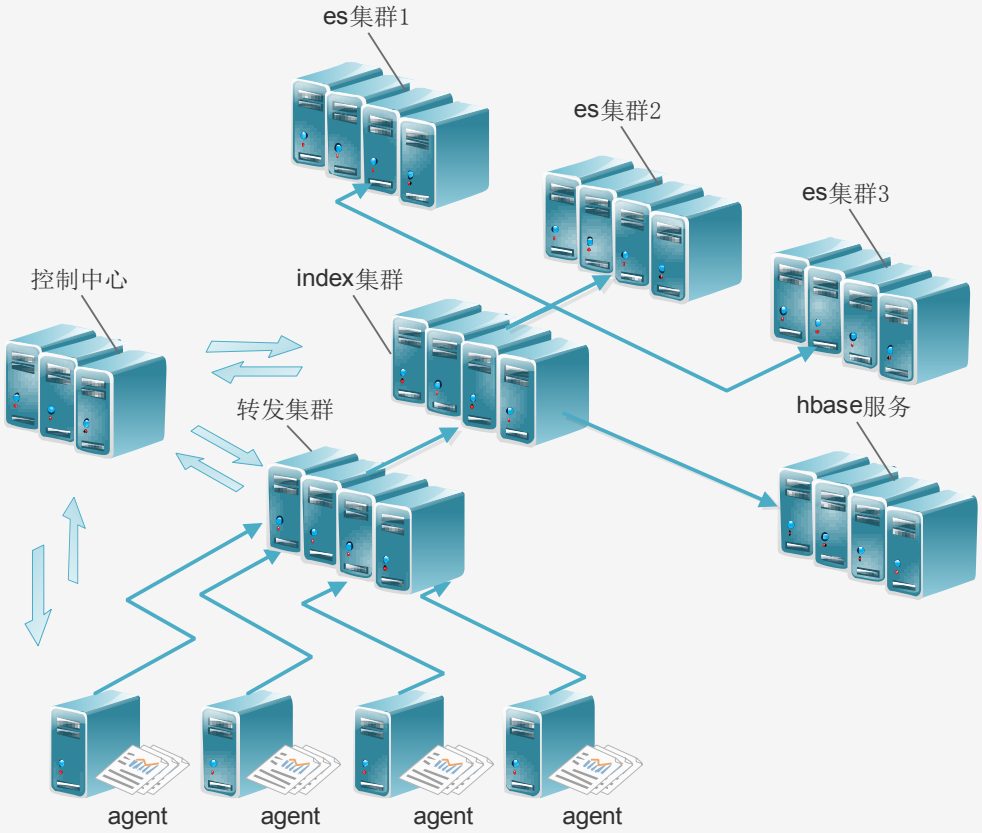
content

1 日志平台es运维分享

1 日志平台的历史


50亿
日志量
2013.06.18
2013.11.11
日志量
100亿
300亿
日志量
2014.06.18
201.11.11
日志量
600亿
1000亿
日志量
2015.06.18
2015.11.11
日志量
1500亿

1 总体架构



子系统介绍

- Agent： 负责日志采集以及发送，java实现
- 转发器：负责数据的高效转发和临时暂存，c实现
- Indexer: 负责接收日志数据，解压和拼接日志数据，控制建索引的策略等，java实现
- 控制器：负责以上三个子系统的策略调度，java实现

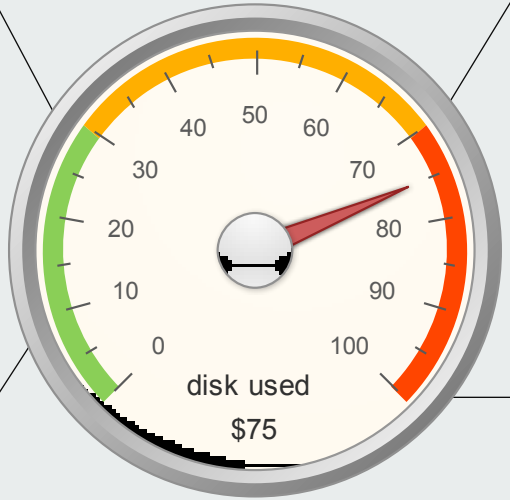
1 磁盘不够用了

去掉备份

早期的es版本数据同步存在bug
当集群不稳定时，集群发生数据同步，会消耗更多资源，引发集群更加不稳定。
维护一份备份的代价太高
节约50%空间

"_source" : {"enabled" : false}

去掉es存储的json格式用于fetch的_source部分
节约40%空间



"store":"no"

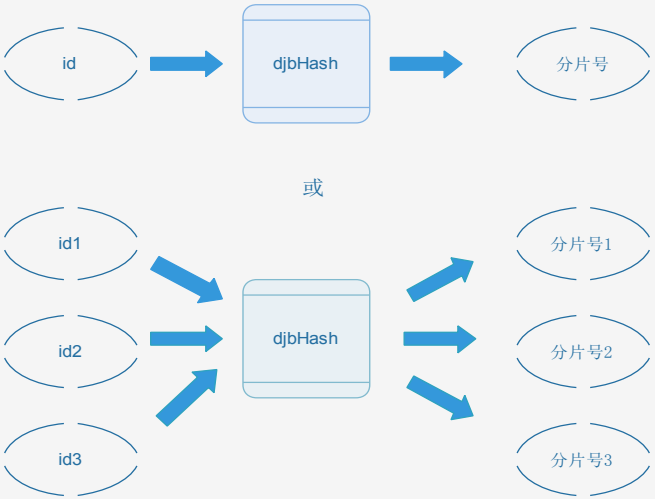
去掉索引的存储部分
对于查询出的一页数据内容转而从hbase中利用uuid取出
节约60%空间

修改分词器和"omit_norms":true

1.修改分词器，增加一个tokenfilter,filter中维护一个set，用于记录一次Input分词过程中已分出的token。如分词过程中产生的token在set中已存在，则该token将被过滤掉。这样将节约position和offset这部分存储。
2.修改mapping,"omit_norms":true,es将不存储.nrm文件。Boost全部为1.0,存储没意义。

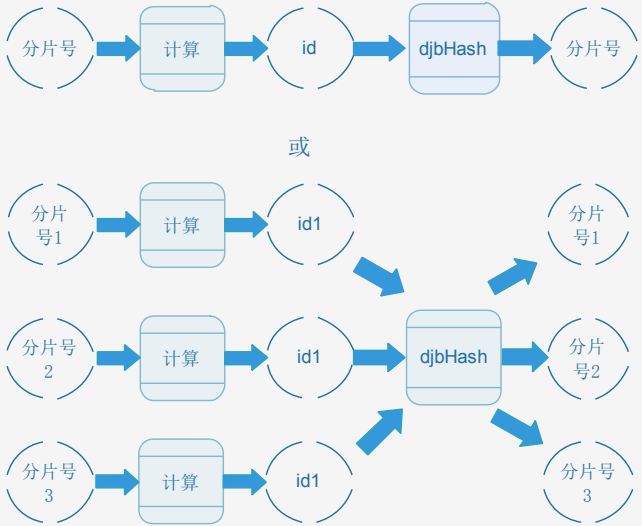
1 精准路由

一般情况下对于es路由的使用：



因为djbhash的计算是在es节点计算的，所以建索引前不确定数据具体分布在哪些分片

精准路由：



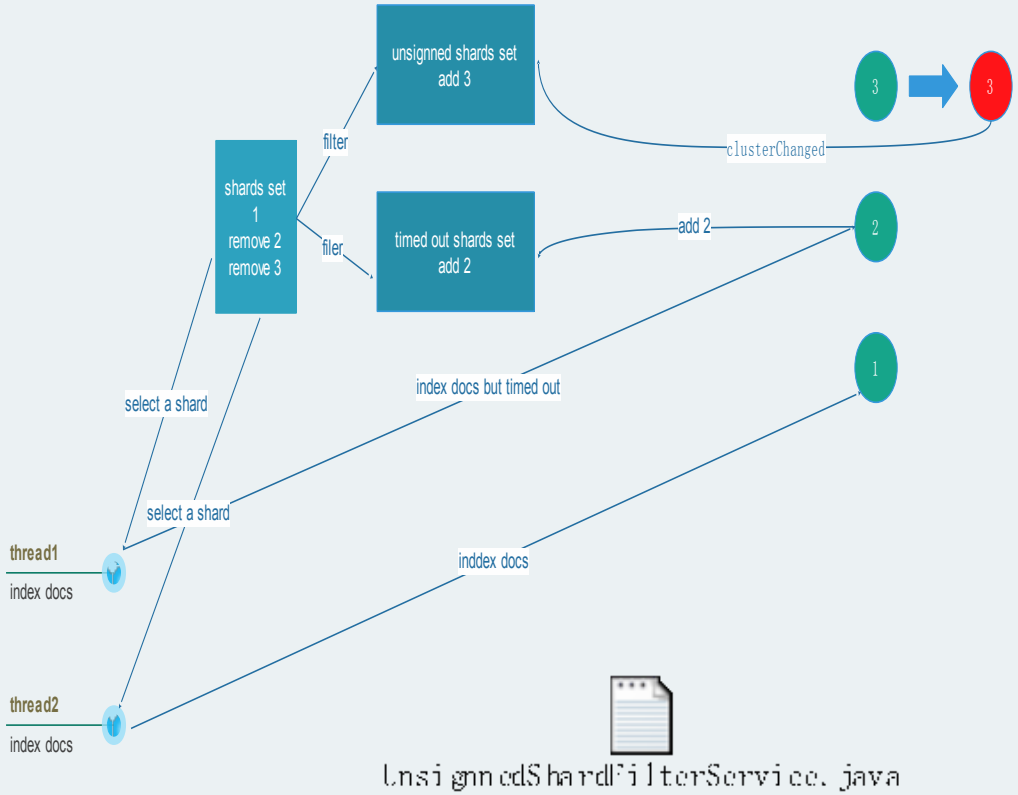
预先根据djbhash反向计算具体分片号对应的路由id,然后使用该id精确控制数据分布的分片。

1 使用nodeClient

nodeClient

- 更高效
- 及时获取集群状态和分片情况
- 消耗自身更多的资源
- 不存储数据也不参与查询，增加集群负担

1 过滤不可用的分片



简要描述

1.indexer模块,使用node client连接集群,使indexer节点成为es集群的一个节点.

2.实现es的ClusterStateListener接口写一个UnsignnedShardFilterListerner,

Override clusterChanged方法,将该listener加入到clusterService中。

3.假设集群出现不稳定的情况下,3号分片的状态由绿转红,变为unsigned状态。

4.集群将触发集群状态改变事件,该事件将在集群内传播,

所有的node节点注册的Listener将触发执行clusterChanged方法.

当然我们自己实现的Listener也将被触发执行.

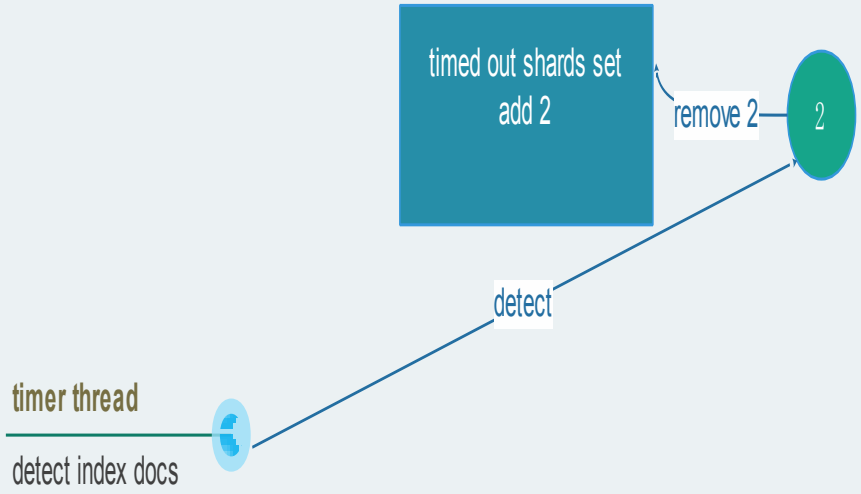
5.clusterChanged方法将Unsignned的分片号记录到本地的set中

6.本地线程在建索引时可以及时规避掉Unsignned分片

7.Timed out分片略有不同,

如某次建索引调用发生超时则将超时分片加入到timed out shards set 中。

1 重新启用不再超时的分片



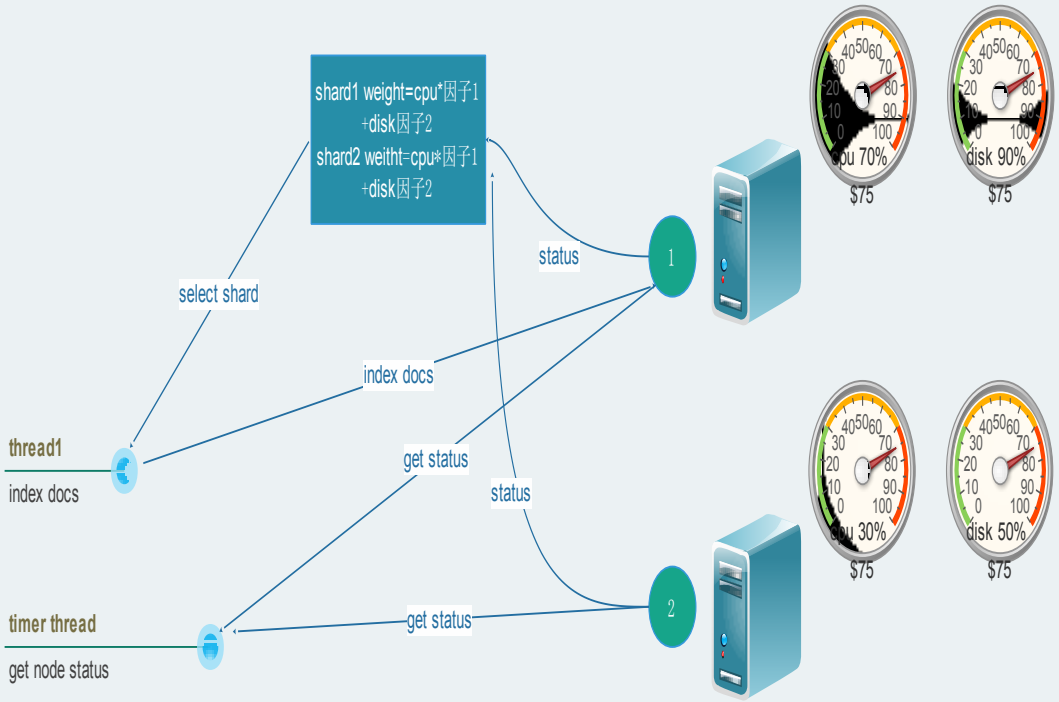
简要描述

1.indexer模块中启动一个timer，定时检测timed out shards set是否存在分片

2. 有分片则逐个分片执行一次建索引操作

3.如分片没有超时，则将从set中移除

1 根据cpu和disk权重选取分片



简要描述

1.indexer模块启动一个timer,定时获取集群状态

NodesStatsResponse response = builder.execute().actionGet();

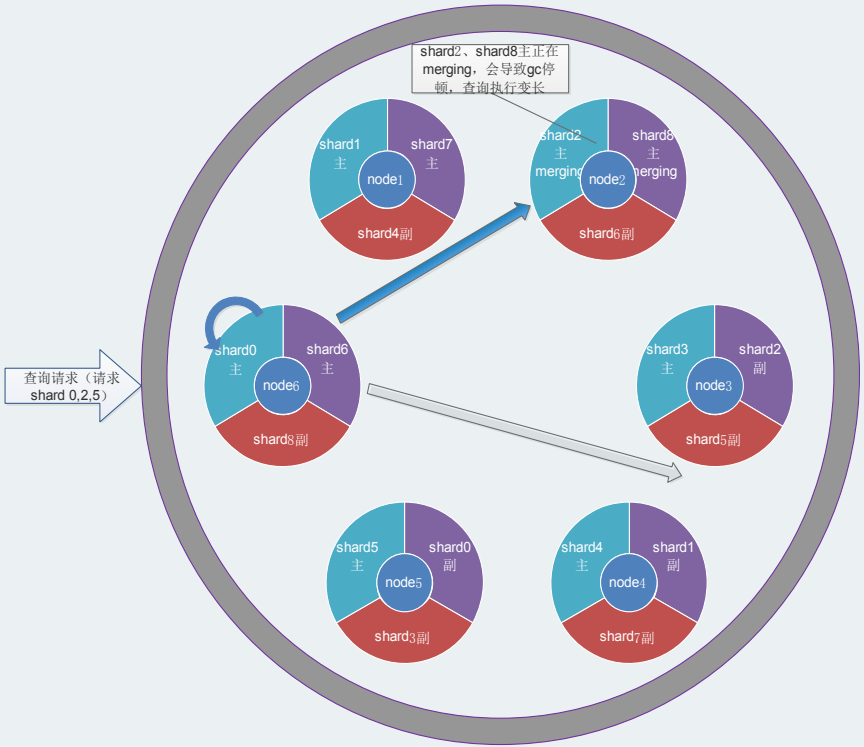
2.从返回的集群状态中获取disk和cpu超标的的节点以及该节点下的对应分片

3.根据配置计算分片的权重

3.本地线程在建索引时尽量选择权重值低的分片

2 源码改造实践一

2 在集群中传递令牌



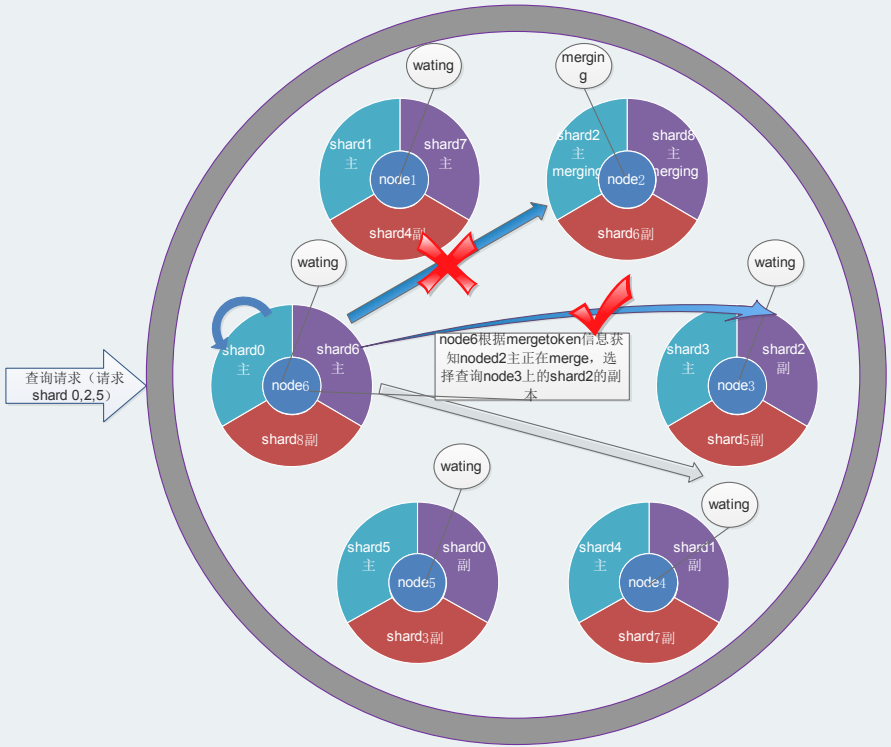
背景

业务场景要求低延迟并且查询性能要求高的情况下，配置的刷新间隔会比较短。如果每次数据更新条数又比较少，每次写入磁盘的segment都比较小，这样这些小段会经过很多次merge操作才能合并成大的segment。在多次merge操作中，相同的数据会反复占用内存，从而易引发jvm 发生时间较长的gc 垃圾回收。gc 垃圾回收会带来jvm 内部线程停顿，导致查询时被查询到的Node查询执行时间变长，影响查询效率。

举个栗子

如果查询转发路由随机组合为{0主,2主，5副}（左图蓝色箭头表示），node2节点上的shard2主和shard8主正在merge，易导致node2 gc停顿，查询请求执行时间变长。

2 在集群中传递令牌



技术方案简要描述

主节点master node执行一个定时程序，定时来更新令牌mergetoken并在整个集群中传递令牌信息。非master node获取到mergetoken信息并存放在本地。在merge触发前会判断是否当前本地节点获取到mergetoken令牌，如果获取到令牌则执行merge，没有获取到则放弃执行merge。同时集群中所有节点都同步保存集群所有节点的mergetoken令牌信息。这样查询请求到达时，可以将查询请求路由到当前没有mergetoken的节点，从而查询请求不会受到merge的影响。

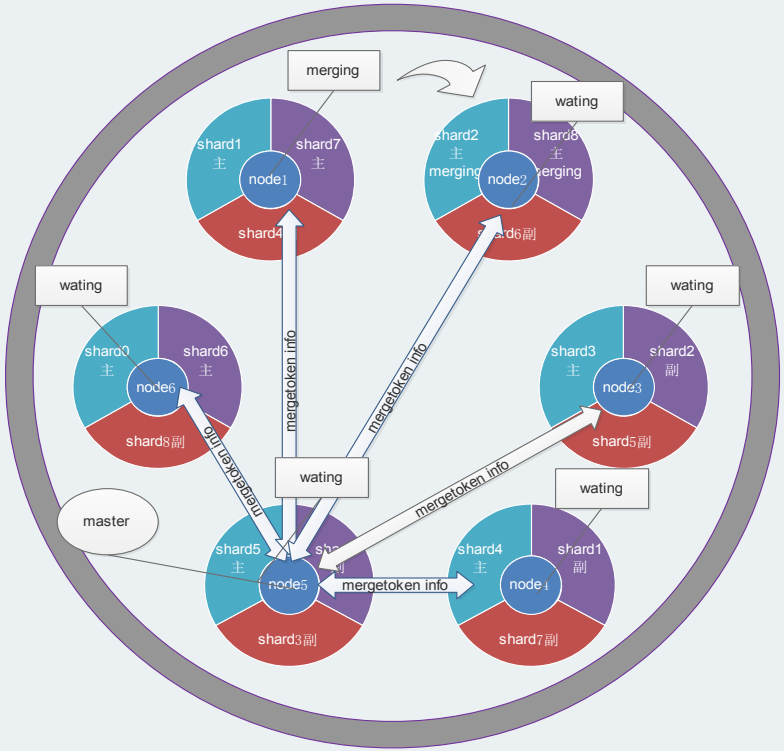
Mergetoken的状态

- Waiting—等待获取令牌
- Merging—获取到令牌
- Merged—令牌使用完毕

Mergetoken的策略

- 轮询策略

2 在集群中传递令牌



简要执行步骤

•步骤1 如下图，主节点Master node(node5)相关类初始化， 将集群内节点进行排序。将节点队列第一个节点（node1）令牌信息更新为merging。启动定时程序，将令牌信息发送给集群内所有节点。

•步骤2 第一个节点在获取到令牌后开始merge，merge结束后将令牌状态更新到merged，并保存到本地。

•步骤3 当master定时程序下一次同步整个集群同步令牌信息时将更新后的merged状态令牌返回给master,master获取后将第一个节点令牌信息更新为waiting,将第二个节点（node2）更新为merging。

•步骤4 循环步骤2到4，直到node进程终止，或者本节点失去master地位

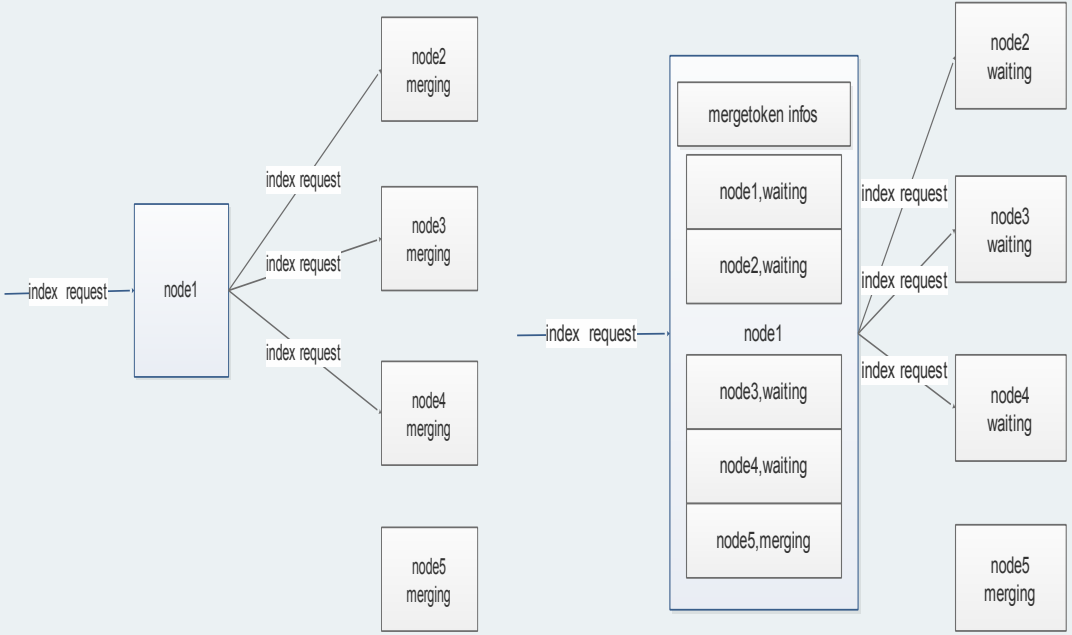
•备注：

•1.如一段时间没有收到第一个节点返回merged状态，将强制收回令牌。

•2.集群发生master节点重选，新master将重新开始执行定时程序。

2 在集群中传递令牌

查询请求路由对比图



限制条件

- 1.shard分片至少存在一个备份分片
- 没有备份的情况下不可以使用本方案。
- 2.集群数据绝大部分数据相对固定，即shard分片下相对大的segment固定，新写入的segment和大的segment相差非常大。
- 该种情况下本方案对查询效率的提升较大，其他情况和原有方案效率差别不大。
- 3.集群节点较少，轮询一遍的时间不会过长。
- 4 如集群节点过多，可修改策略，让多个node获得mergeoken。只要保证wating状态的node上存在一套完整的index.

3 源码改造实践二

3 数据不写入索引

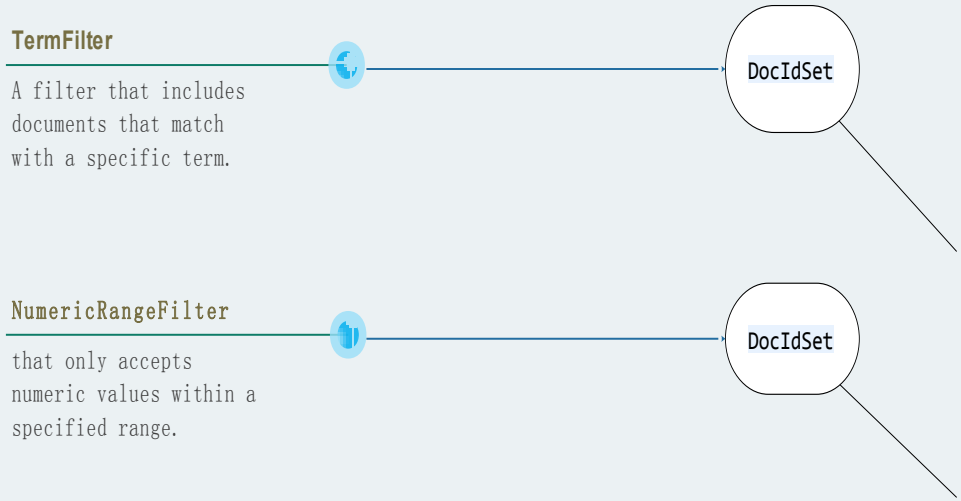
Id（业务id）	content	status1	status2
100	abcdefg	1	n
101	elasticseach	0	y
102	阿猫阿狗	1	n
103	Lucene action	0	y

背景

- 对于status1和status2这种字段我们一般采取filterd query并开启filter cache来构建查询。不过假如status1和status2 update非常频繁，还是会导致之前讨论的gc停顿的问题。

所以开始考虑，是否这种字段可以不去update？

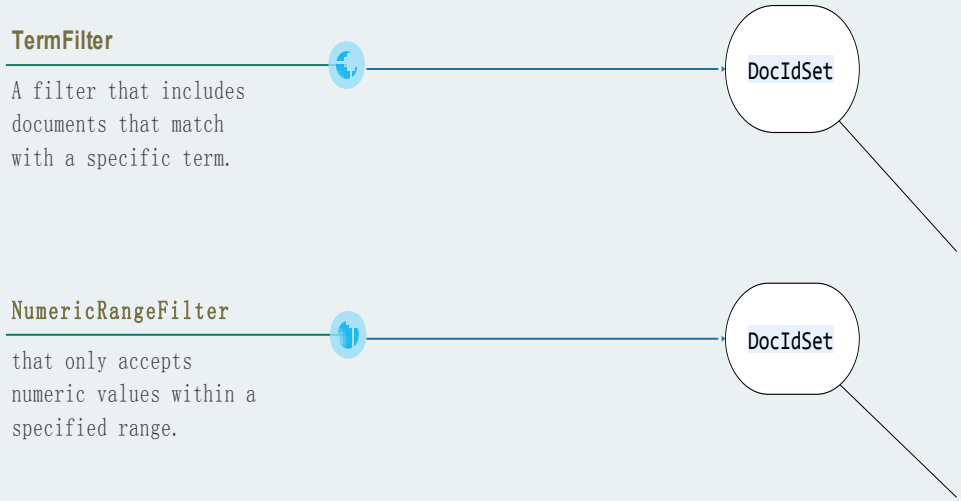
3 数据不写入索引



Filter

- public abstract class Filter {
- ...
- public abstract DocIdSet getDocIdSet(AtomicReaderContext context, Bits acceptDocs) throws IOException;
- }
- Filter的目的是产生一个docIDSet，用于最终合并倒排表，不过这里的docID是lucene索引的内部编号。

3 数据不写入索引



Filter

- public abstract class Filter {
- ...
- public abstract DocIdSet getDocIdSet(AtomicReaderContext context, Bits acceptDocs) throws IOException;
- }
- Filter的目的是产生一个docIDSet，用于最终合并倒排表，不过这里的docID是lucene索引的内部编号。

3 数据不写入索引

Id（业务id）	content	status1	status2
100	abcdefg	100	100
101	elasticseach	101	101
102	阿猫阿狗	102	102
103	Lucene in action	103	103

100	1
101	0
102	1
103	0

内部编号1	1
内部编号2	0
内部编号3	1
内部编号4	0

技术方案

1.写一个新filter，维护一个docIdset,初始值全部为1

2.将100→1这种key-value结构维护到外部的redis或者hbse

3.Es建立好redis或者hbase的client,定时增量同步这个key-value结构。

4.查询请求执行时将查到的status1字段值100转换为1或0

5.重新构建好docSetID

6.将docSetID缓存起来，这里原理和es 其他 filter cache一致

3 数据不写入索引

```
        "type": "string"
    },
    "isShow": {
        "doc_values": true,
        "type": "redis"
    }
```

使用方法

1.方案利用doc_values实现，需要打开doc_values

2.因定义了一个全新的field和对应的mapping，需要修改对应field的mapping.

THANK YOU

elastic

VMC | vivo 移动互联网

Elastic中文社区技术沙龙【深圳站】

elastic

VMIC | vivo 移动互联网

http://elastic.co

http://elasticsearch.cn

http://elasticsearch.cn/article/99

http://www.vivo.com.cn

http://dev.vivo.com.cn

http://hr.vivo.com.cn

http://www.vivo.com