## 原理

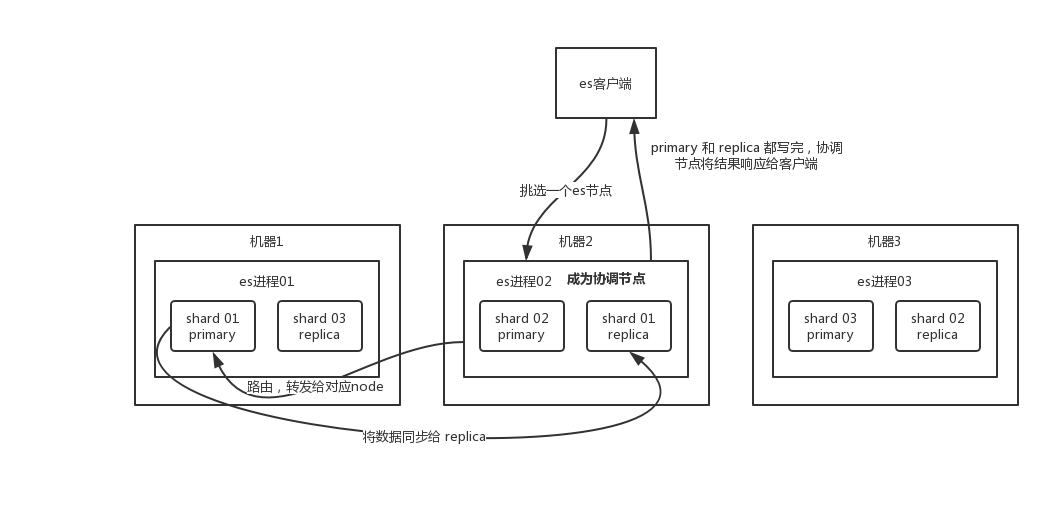
**底层 lucene**

简单来说，lucene 就是一个 jar 包，里面包含了封装好的各种建立倒排索引的算法代码。我们用 Java 开发的时候，引入 lucene jar，然后基于 lucene 的 api 去开发就可以了。

通过 lucene，我们可以将已有的数据建立索引，lucene 会在本地磁盘上面，给我们组织索引的数据结构

### 写数据

primary, 再同步



1. ES客户端选择一个node发出写请求, 该node称为coordinating node(协调节点).
2. Coordinating node 将请求转发到该document的primary node
3. Primary node 处理请求，同时将数据同步到所有的replica node
4. 所有replica node 都同步完成后, coordinating node 响应客户端。

### 读数据

随机轮询所有节点。

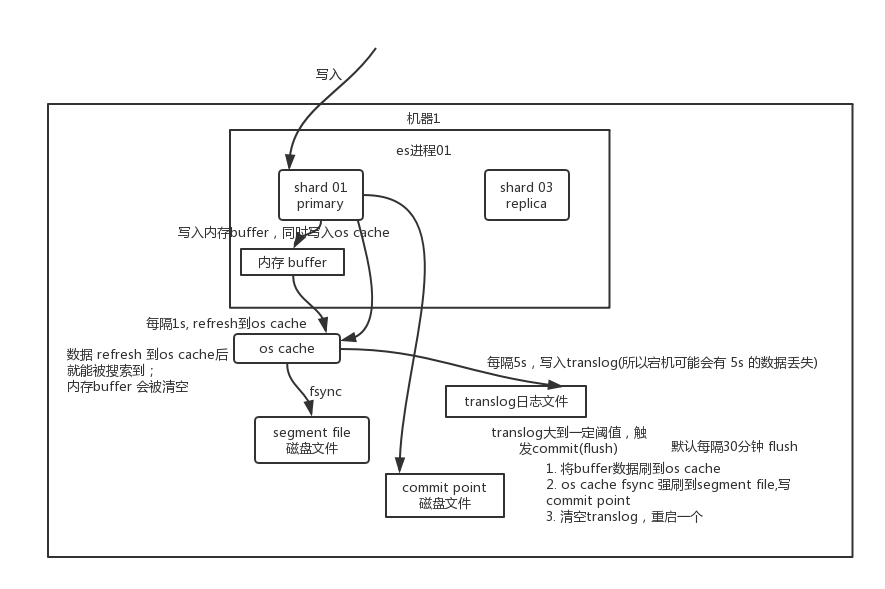
1. ES客户端选择一个node发出读请求，该node称为coordinating node.
2. Coordinating node 对doc id 进行hash, 判断doc id分配到了哪个shard, 将请求转发到该node. Primary, replica 中使用round-robin随机轮询负载均衡算法，选择一个node.
3. Node 返回document 给 coordinating node.
4. Coordinating node 响应客户端。

### 搜索数据

1. ES客户端发送请求到coordinating node
2. Coordinating node 将搜索请求转发到所有shard对应的primary node 或者replica node.
3. Query phase: 每个shard将搜索结果(其实就是doc id)返回给coordinating node, 由coordinating node 进行合并、排序、分页等。
4. Fetch phase: coordinating node 根据doc id去各个节点上获取实际的document, 最终返回给客户端。

我认为可以类比于mysql 查询， 上面的读数据类似于mysql中的主键查询，搜索数据类似于非主键查询，需要回表查询。

### 写数据底层



我认为可以类比于Redis的AOF持久化.

内存buffer <-> AOF缓冲区.

Os cache: 操作系统特有，写入磁盘前使用的专用缓冲区.

Segment file <-> AOF文件.

先写入buffer, buffer中内容搜索不到.

translog作用: crash safe, 类似于mysql 中的redo log.

**commit**

默认30min

将buffer refresh 到 OS cache, 清空buffer.

将OS cache fsync 到磁盘文件.

清空translog, 重启新的translog.

translog 同样先写入OS cache， 每隔5s 同步到磁盘。

ES是准实时(NRT): 每隔1s refresh, 写入1s后才可以搜索到。

ES 可能丢数据: 有5s的数据, 停留在buffer, translog OS cache, 此时宕机，会导致5s内的数据丢失。

数据写入 segment file 之后，同时就建立好了倒排索引.

### 删除、更新底层

删除操作，commit 的时候会生成一个 .del 文件，里面将某个 doc 标识为 deleted 状态，那么搜索的时候根据 .del 文件就知道这个 doc 是否被删除了。

更新操作，就是将原来的 doc 标识为 deleted 状态，然后新写入一条数据。

1 秒钟一个 segment file，这样下来 segment file 会越来越多，此时会定期执行 merge。每次 merge 的时候，会将多个 segment file 合并成一个，同时这里会将标识为 deleted 的 doc 给物理删除掉。