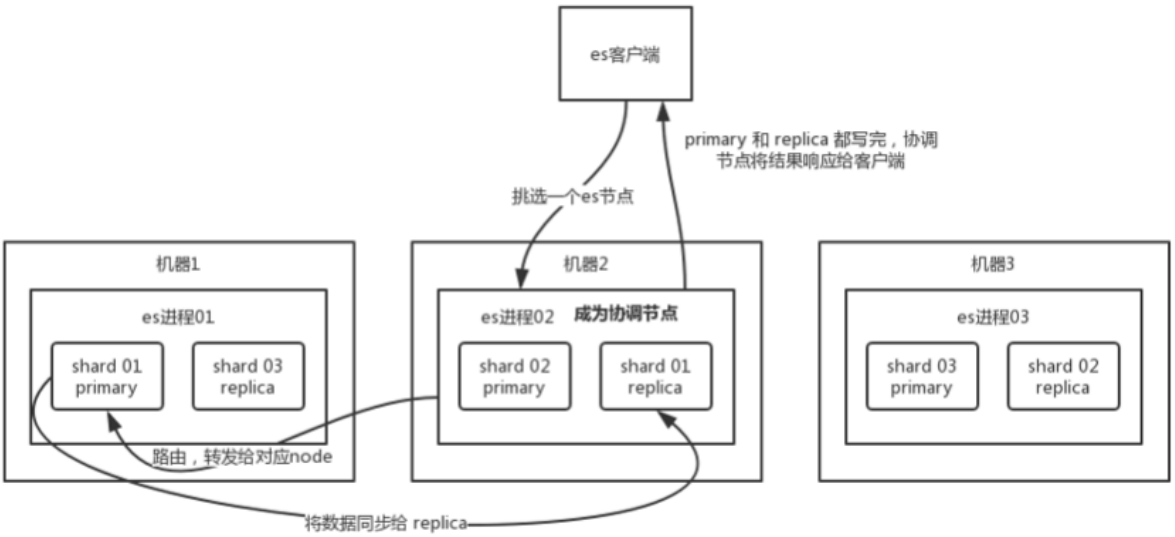
ES写数据过程：

- 客户端选择一个node发送请求过去，这个node就是coordinating node(协调节点)
- coordinating node对document进行路由，将请求转发给对应的node（有primary node）
- 实际的node上的primary shard处理请求，然后将数据同步到replica node
- coordinating node如果发现primary node和所有的replica node都搞定之后，就返回响应结果给客户端。



ES读数据过程：

可以通过doc id来查询，会根据doc id进行hash，判断出来当时把doc id分配到了哪个shard上面去，从哪个shard去查询。

- 客户端发送请求到任意一个node，成为coordinating node。
- coordinating node对doc id进行哈希路由，将请求转发到对应的node，此时会使用round-robin随机轮询算法，在primary shard以及其所有的replica中随机选择一个，让读请求负载均衡。
- 接收请求的node返回document给coordinating node
- coordinating node返回document给客户端。

ES搜索数据过程：

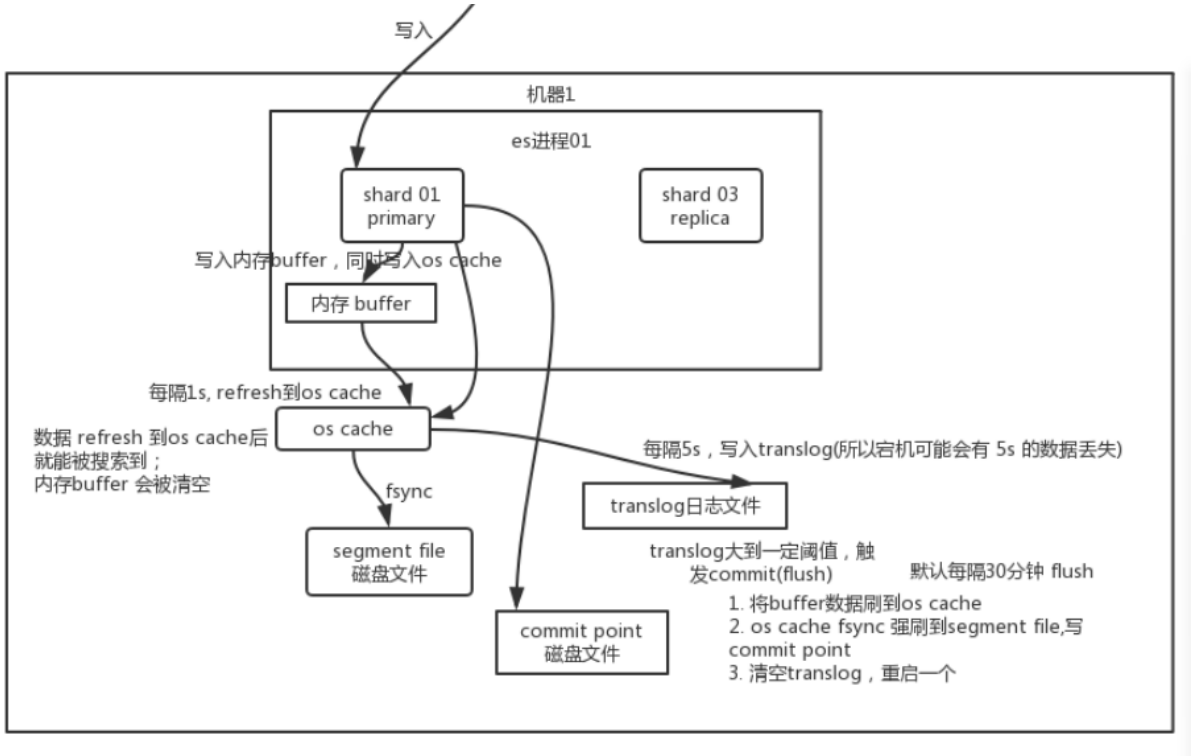
- ES最强大的是做全文检索，比如有三条数据。

```
java真好玩儿啊
java好难学啊
j2ee特别牛
```

你根据java关键词来搜索，将包含java的document给搜索出来，es就会给你返回：java真好玩儿，java好难学啊。

- 客户端发送请求到一个coordinating node
- coordinating node将搜索请求转发到所有的shard对应的primary shard或replica shard，都可以。
- query phase：每个shard将自己的搜索结果(其实就是一些doc id)返回给coordinating node，由coordinating node进行数据的合并、排序、分页等操作，产出最终结果。
- fetch phase：接着由coordinating node根据doc id去各个节点上拉取实际的docment数据，最终返回给客户端。

写数据底层原理：



先写入内存buffer，在buffer里的时候数据是搜索不到的；同时将数据写入translog日志文件。

如果buffer快满了，或者到一定时间，就会将内存buffer数据refresh到一个新的segment file中，但是此时数据不是直接进入segment file磁盘文件，而是先进入os cache。这个过程就是refresh。

每隔1秒钟，es将buffer中的数据写入一个新的segment file，每秒钟会产生一个新的磁盘文件segment file，这个segment file，这个segment file中就存储最近1秒内buffer中写入的数据。

但是如果buffer里面此时没有数据，不会执行refresh操作，如果buffer里面有数据，默认1秒钟执行一次refresh操作，刷入一个新的segment file中。
操作系统里面，磁盘文件其实都有一个东西，叫做os cache，即操作系统缓存，就是说数据写入磁盘之前，会先进入os cache，先进入操作系统级别的一个内存缓存中去。只要buffer中的数据被refresh操作刷入os cache中，这个数据就可以被搜索到了。

为什么叫es是准实时的？？
默认是1秒refresh一次的，所以es是准实时的，因为写入的数据1秒之后才能被看到。可以通过es的restful api或者java api，手动执行一次refresh操作，就是手动将buffer中的数据刷入os cache中，让数据立马就可以搜索到。只要数据被输入os cache中，buffer就会被清空了，因为不需要保留buffer了，数据在translog里面已经持久化到磁盘一份了。
重复上面的步骤，新的数据不断进入buffer和translog，不断将buffer数据写入一个又一个新的segment file中去，每次refresh完buffer清空，translog保留。随着这个过程推进，translog会变得越来越大。当translog达到一定长度的时候，就会触发一次commit操作。
commit操作发生第一步，就是将buffer中现有的数据refresh到os cache中去，清空buffer。然后，将一个commit point写入磁盘文件，里面标识着这个commit point对应的所有segment file，同时强行将os cache中目前所有的数据都fsync到磁盘文件中去。最后清空现有translog日志文件，重启一个translog，此时commit操作完成。
这个commit操作叫做flush。默认30分钟自动执行一次flush，但如果translog过大，也会触发flush。flush操作就对应着commit的全过程，我们可以通过es api，手动执行flush操作，手动将os cache中的数据fsync强刷到磁盘上去。
translog日志文件的作用是什么？
在执行commit操作之前，数据要么是停留在buffer中，要么是停留在os cache中，无论是buffer还是os cache都是内存，一旦这台机器挂了，内存中的数据就全部丢了。所以需要将数据对应的操作写入一个专门的日志文件translog中，一旦此时机器宕机，再次重启的时候，es会自动读取translog日志文件中的数据，恢复到内存buffer和os cache中去。
translog其实也是可以先写入os cache的，默认每隔5s刷一次到磁盘中去，