**1、什么是node**

ES集群中每一个节点就是一个node，或者一个Elasticsearch实例就是一个节点。

node分类：

a、主节点：主节点不接受客户端的请求，他主要控制Elasticsearch集群，负责集群中的操作，比如创建/删除一个索引，跟踪哪些节点是群集的一部分，并决定哪些分片分配给相关的节点。主节点处理集群的状态并广播到其他节点，并接收其他节点的确认响应。 默认情况下任何一个集群中的节点都有可能被选为主节点，每个节点都可以通过设定配置文件elasticsearch.yml中的node.master属性为true(默认)，node.data属性设置为false，成为主节点。对于大型的生产集群来说，推荐使用一个专门的主节点来控制集群，该节点将不处理任何用户请求，稳定的主节点对集群的健康是非常重要的。

b、数据节点：该节点具有存储数据和执行数据相关的操作，如增删改查，搜索，和聚合操作。数据节点对cpu，内存，io要求较高，在优化的时候需要监控数据节点的状态，当资源不够的时候，需要在集群中添加新的节点。默认情况下，每个节点都可以通过设定配置文件elasticsearch.yml中的node.data属性为true(默认)成为数据节点。如果我们要使用一个专门的主节点，应将其node.data属性设置为false。

c、客户端节点：该节点主要将客户端的请求路由到集群中的各个节点，扮演一个负载均衡的角色。将node.master属性和node.data属性都设置为false，那么该节点就是一个客户端节点。

d、部落节点：部落节点可以跨越多个集群，它可以接收每个集群的状态，然后合并成一个全局集群的状态，它可以读写所有节点上的数据，部落节点在elasticsearch.yml中的配置如下：tribe:\*:

**2.分片(shard)**

主分片：在索引建立的时候就已经确定了主分片数，但是副本分片数可以随时修改，索引内任意一个文档都归属于一个主分片，所以主分片的数目决定着索引能够保存的最大数据量。

副分片：一个副本分片只是一个主分片的拷贝。副本分片作为硬件故障时保护数据不丢失的冗余备份，并为搜索和返回文档等读操作提供服务。

**3、什么是documnet routing（数据路由）**

　　当客户端发起创建document的时候，es需要确定这个document放在该index哪个shard上。这个过程就是数据路由。

路由过程：

　　路由算法：shard = hash(routing) % number\_of\_primary\_shards

　　routing:每次增删改查一个document的时候，都会带过来一个routing number，他的默认值就是这个document的\_id（可能是手动指定，也可能是自动生成）routing = \_id，所以决定一个document在哪个shard上，最重要的一个值就是routing值。相同的routing值，从hash函数中，产出的hash值一定是相同的。

　　　　number\_of\_primary\_shards：主分片。

　 例如：假设\_id=1会将这个routing值，传入一个hash函数中，产出一个routing值的hash值，hash(routing) = 21，然后将hash函数产出的值对这个index的primaryshard的数量求余数，21 % 3 = 0 就决定了，这个document就放在P0上。

**4、ES document增删改的处理流程**

ES增删改的处理流程：增删改的请求一定作用在主分片上。

假如我们es集群有3个node，每个node上一个主分片一个复制分片。

1、客户端发起一个PUT请求，假如该请求被发送到第一个node节点，那么该节点将成为协调节点(coordinating node)，如图P1所在的节点就是协调节点。他将根据该请求的路由信息计算，该document将被存储到哪个分片。

2、第二步 通过计算发现该document被存储到p0分片，那么就将请求转发到node2节点。

3、第三步 P0根据请求信息创建document，和相应的索引信息，创建完毕后将信息同步到自己的副本节点R0上。

4、第四步 P0和R0将通知我们的协调节点，任务完成情况。

5、第五部 协调节点响应客户端最终的处理结果。

请求先写到Memory Buffer，然后定时（默认1s）写到（Filesystem Cache） （内存到磁盘）

这个过程叫做refresh

当然在某些情况下，存在Momery Buffer和Filesystem Cache的数据可能会丢失，ES是通过translog的机制来保证数据的可靠性的。其实现机制是接收到请求后，同时也会写入到translog中，当Filesystem cache中的数据写入到磁盘中时，才会清除掉，这个过程叫做flush；

**5、ES document读请求流程**

　搜索被执行成一个两阶段过程，我们称之为 Query Then Fetch；

　　在初始查询阶段时，查询会广播到索引中每一个分片拷贝（主分片或者副本分片）。 每个分片在本地执行搜索并构建一个匹配文档的大小为 from + size 的优先队列。PS：在搜索的时候是会查询Filesystem Cache的，但是有部分数据还在Memory Buffer，所以搜索是近实时的。

　　每个分片返回各自优先队列中 所有文档的 ID 和排序值 给协调节点，它合并这些值到自己的优先队列中来产生一个全局排序后的结果列表。

　　接下来就是 取回阶段，协调节点辨别出哪些文档需要被取回并向相关的分片提交多个 GET 请求。每个分片加载并 丰富 文档，如果有需要的话，接着返回文档给协调节点。一旦所有的文档都被取回了，协调节点返回结果给客户端。

补充：Query Then Fetch的搜索类型在文档相关性打分的时候参考的是本分片的数据，这样在文档数量较少的时候可能不够准确，DFS Query Then Fetch增加了一个预查询的处理，询问Term和Document frequency，这个评分更准确，但是性能会变差。

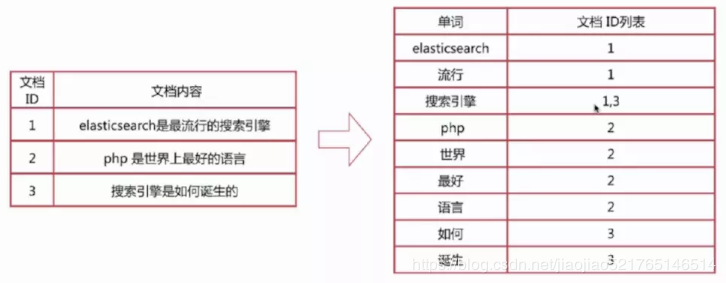
**6.在并发情况下，Elasticsearch如果保证读写一致？**

写：可以通过版本号使用乐观并发控制，以确保新版本不会被旧版本覆盖，由应用层来处理具体的冲突；

读：对于读操作，可以设置replication为sync(默认)，这使得操作在主分片和副本分片都完成后才会返回；如果设置replication为async时，也可以通过设置搜索请求参数\_preference为primary来查询主分片，确保文档是最新版本

**7倒排索引**

文档内容分词，由内容索引到id。倒排索引是一种像数据结构一样的散列图，可将用户从单词导向文档或网页



倒排索引-查询过程

查询包含“搜索引擎”的文档

通过倒排索引获得“搜索引擎”对应的文档id列表，有1，3

通过正排索引查询1和3的完整内容

返回最终结果

**8 master选举 （bully算法）**

1. 选举时间点

选举的时间点有两个

某个新节点加入了集群或者某个节点从宕机中恢复

集群中的某个节点检测到leader崩溃

2. 选举流程

选举的流程说明如下

a节点node向所有比自己大的节点发送选举消息(选举为election消息)

b如果节点node得不到任何回复(回复为alive消息)，那么节点node成为master，并向所有的其它节点宣布自己是master(宣布为Victory消息)

c如果node得到了任何回复，node节点就一定不是master，同时等待Victory消息，如果等待Victory超时那么重新发起选举

3. Elasticsearch编号比较

Elasticsearch编号比较的判断依据有两个，首先是ClusterState版本号的比较，版本号越大优先级越高，然后是节点id的比较，id越小优先级越高。ClusterState是Master向集群中各个节点发送的集群状态，这个状态有一个版本号码，如果集群状态发生了变化，比如集群新增了节点成员或者有节点成员退出了，那么这个版本号就会加一，比对这个版本号的目的是让拥有最新状态的节点成为Master的优先级最高。

4. Bully算法缺陷

4.1 Master假死

Master节点承担的职责负载过重的情况下，可能无法即时对组内成员作出响应，这种便是假死。如果上图中的P6节点假死，于是P5节点成为了Master节点，但是在P6节点负载减轻之后，P6节点又对组内成员作出了响应，P6节点又会成为Master节点，如此反复，整个集群状态就会非常不可靠。

Elasticsearch是如何解决这个问题的呢？在Bully算法中，Master节点P6因为负载重，来不及对P3节点作出响应，所以P3节点通知P4,P5节点进行选举。在Elasticsearch中，P3节点发现Master P6对自己长时间不作出响应，P3节点会请求其它节点判断P6节点是否存活，如果有1/2以上节点都认定P6存活，那么P3就会放弃发起选举

4.2 脑裂问题

脑裂问题指的是一个集群中出现了两个及以上的Master节点。

比如上图中集群因为网络原因分成了两个部分，一个部分称为partition1包含P3,P5,P6节点，另一部分称为partition2包含P2,P1,P4节点，这两个partition因为网络原因比如路由器短时故障造成不能相互通信的问题。

那么网络分区一根据Bully算法选举出了P6作为master，而网络分区二选举出了P4作为master，这就是脑裂问题。

Elasticsearch集群机制采用了最小参与节点的方案解决的。假设elasticsearch集群中有资格投票的实例个数是n，节点想要成为master必须要得到n/2 +1票数(在本示例中是4)。上图中的分区一P6和分区二中的P4，任何一个master节点所在的分区集群的候选节点数目都小于4，更不可能得到4个选票，所以整个Elasticsearch集群处于瘫痪状态

我们也可以强制指定elasticsearch节点在有M个候选节点的情况下能选举出一个主节点，但是如果配置数小于上文提到的 n/2 +1 那么会出现脑裂的情况，M的配置参数如下

discovery.zen.minimum\_master\_nodes

4.3 Master降级

Master主动降级发生在两种情况。

第一种是master发现自己能连接到的其它节点数目小于n/2 + 1，那么master自动降级为candidate。

第二种是Master在ping其它节点时候，如果发现了其它master，那么当前的master会比较cluster\_state的version，如果当前master的version小，那么主动降级为candidate并主动加入另外一个master节点

**9 性能优化**

1. 写优化

(1) refresh\_interval加大，减少刷新次数 (搜索实时性降低)

(2)使用bulkapi ，批量提交（Bulk 默认设置批量提交的数据量不能超过 100M。数据条数一般是根据文档的大小和服务器性能而定的，但是单次批处理的数据大小应从 5MB～15MB 逐渐增加，当性能没有提升时，把这个数据量作为最大值）

(3)使用SSD

（4）Flush 的主要目的是把文件缓存系统中的段持久化到硬盘，当 Translog 的数据量达到 512MB 或者 30 分钟时，会触发一次 Flush。

index.translog.flush\_threshold\_size 参数的默认值是 512MB，我们进行修改。

增加参数值意味着文件缓存系统中可能需要存储更多的数据，所以我们需要为操作系统的文件缓存系统留下足够的空间

（5）减少副本数量，需要大批量进行写入操作，可以先禁止 Replica 复制，设置 index.number\_of\_replicas: 0 关闭副本。在写入完成后，Replica 修改回正常的状态

1. 读优化

（1）避免大结果集和深度分页

使用scrollApi:查询100页，每页大小100

普通查询，查每页的时候，每个分片查100条数据，然后汇总。

Scroll,一次从每个分片查10000，同节点聚合 n×10000 条数据进行合并、排序，并将排名前 10000 的结果快照起来。这样做的好处是减少了查询和排序的次数。

（2）定期删除

由于在 Lucene 中段具有不变性，每次进行删除操作后不会立即从硬盘中进行实际的删除，而是产生一个 .del 文件记录删除动作。

随着删除操作的增长，.del 文件会越来也多。当我们进行查询操作的时候，被删除的数据还会参与检索中，然后根据 .del 文件进行过滤。.del 文件越多，查询过滤过程越长，进而影响查询的效率。

当机器空闲时，我们可以通过如下命令删除文件，来提升查询的效率：

$ curl -XPOST localhost:9200/chandler/\_forcemerge?only\_expunge\_deletes=true

{"\_shards":{"total":10,"successful":5,"failed":0}}

（3）合理设置字段类型

不涉及到模糊或者范围查询的，用keyword

1. 系统层面

设置合理的堆大小，不宜过小（增加gc压力）

也不宜过大（32G以内，否则内存指针膨胀到64位，并且gc会有更大停顿）

堆大小不要超过50%，否则lucence没有足够内存

禁用swap(虚拟内存)

节点数不宜过多