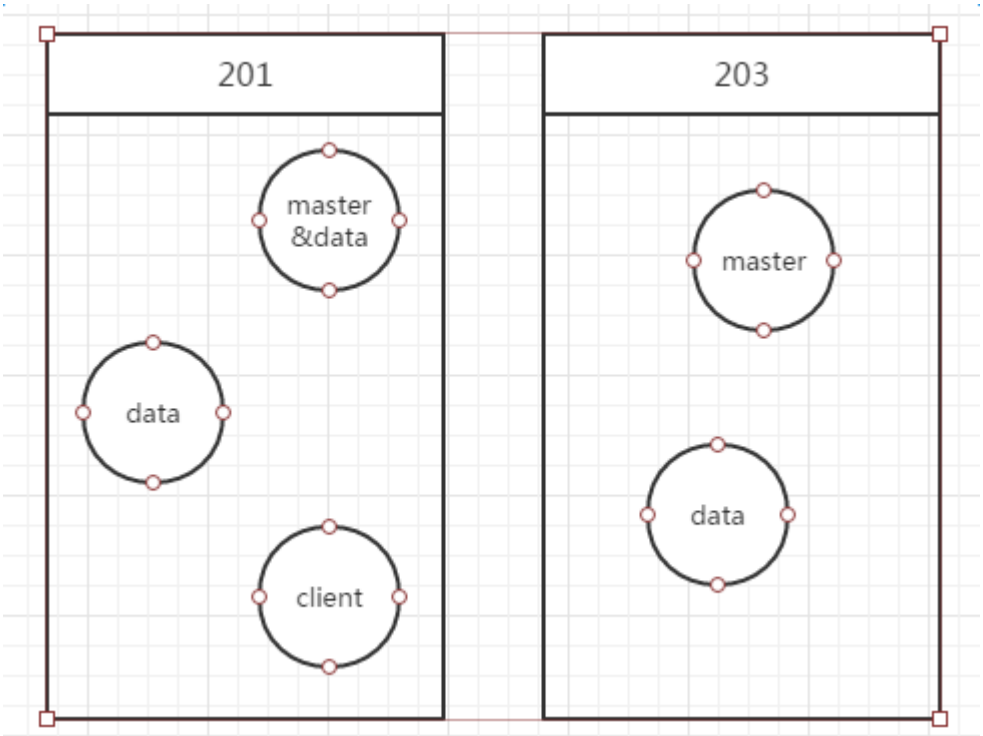
ES 写入速度优化实践

ES 的默认设置,是综合考虑数据可靠性,搜索实时性,写入速度等因素的,当你离开默认设置,追求极致的写入速度时,很多是以牺牲可靠性和搜索实时性为代价的.有时候,业务上对两者要求并不高,反而对写入速度要求很高。例如在初次构建索引后需要一次性导入大量数据，这时索引还未完成数据导入，因此对于查询的需求几乎可以忽略；同时对于某些场景下，数据不需要强一致性，可以容忍一定程度的丢失。

基于以上背景，我们可以尝试从多个角度去优化 ES 的写入速度，使得大数据量情况下可以快速导入：



本次实践主要使用了 3 中优化策略：

1. 暂时关闭副本和自动刷新

在 ES 中默认会对每个分片分配一个副本，这样在每次索引文档时，所有分片都会对文档建立一次索引。取消副本后，每次请求时需要建立索引的分片数直接减半，在数据完成导入后我们再将副本数恢复，这时只需简单的进行分片拷贝即可，省去了建立索引的步骤。

同时 ES 会有一个自动刷新索引的过程，主要是为了查询的实时性，在暂时不需要查询的情况下，可以将索引改为手动刷新，数据导入完成后恢复即可。

以下是相关配置：

```
"number_of_shards": 5,
"number_of_replicas": 0,
"index.refresh_interval": -1 #-1 表示不自动刷新
```

2. translog flush 间隔调整（影响最大）

默认设置下,translog 的持久化策略为:每个请求都 flush，translog 用于

记录请求相关信息，避免数据丢失，刷新的目的是将内存中缓存的 log 写入磁盘。
默认配置项为:
`index.translog.durability: request`
如果系统可以接受一定几率的数据丢失,调整 translog 持久化策略为周期性和一定大小的时候 flush:

```
index.translog.durability: async #根据 sync_interval 时间周期性刷新
index.translog.sync_interval: 120s #刷新周期
index.translog.flush_threshold_size: 1024mb #超过 size 后强制刷新
```

3. **优化节点间的任务分布,将任务尽量均匀分发到各节点**
   每次 bulk 请求随机挑选集群中不同节点，避免都由同一个节点去接受处理请求，这样会导致单一节点压力过大，内存消耗严重。

```
self.es_client = [
    Elasticsearch([{
        'host': '192.168.100.201',
        'port': 9201
    }]),
    Elasticsearch([{
        'host': '192.168.100.201',
        'port': 9202
    }]),
    Elasticsearch([{
        'host': '192.168.100.201',
        'port': 9203
    }]),
    Elasticsearch([{
        'host': '192.168.100.203',
        'port': 9204
    }]),
    Elasticsearch([{
        'host': '192.168.100.203',
        'port': 9205
    }]),
]
```

```
success, _ = bulk(
    choice(self.es_client),
    iterms,
    index=self.index_name,
    raise_on_error=True)
```

4. **使用 bulk 请求**

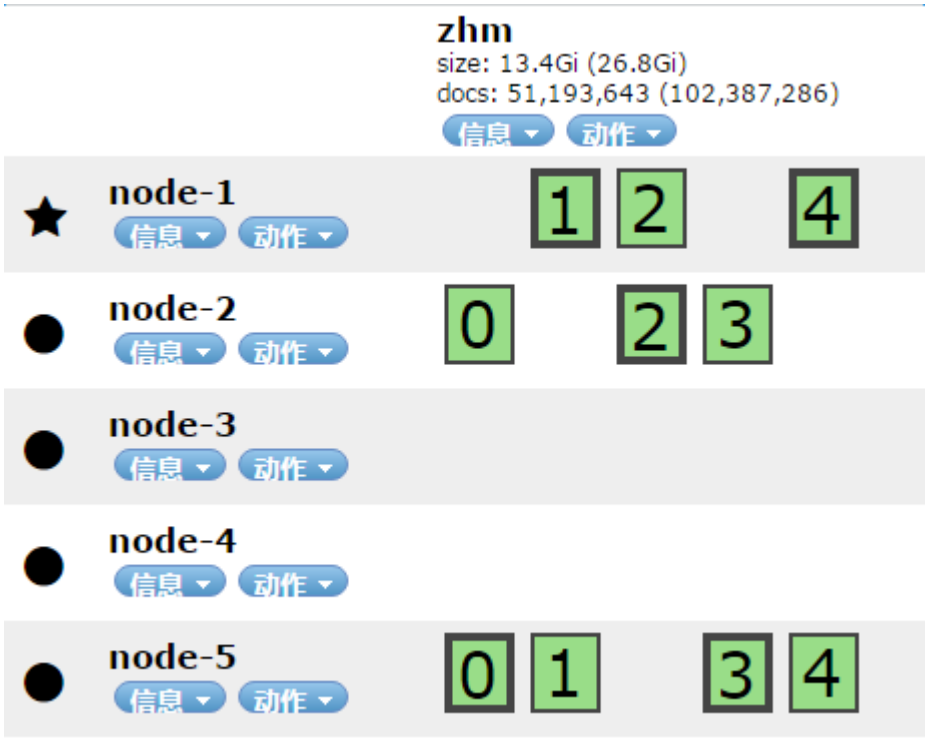
Bulk 可以进行批量文档导入，相比于一条条的索引请求极大提高了索引效率，是导入大量文档时的必选途径。
bulk 会把将要处理的数据载入内存中，所以数据量是有限制的，最佳的数据量不是一个确定的数值，它取决于你的硬件，你的文档大小以及复杂性，你的索引以及搜索的负载。
一般建议是 1000-5000 个文档，如果你的文档很大，可以适当减少队列，大小建议是 5-15MB，默认不能超过 100M，可以在 es 的配置文件（即$ES_HOME 下的 config 下的 elasticsearch.yml）中。

其他一些未使用的优化策略：_all、_source 字段优化；调整 bulk 线程池和队列；优化磁盘间的任务均匀情况,将 shard 尽量均匀分布到物理主机的各磁盘。

结果：

使用之前的脚本进行数据导入时，以每次 bulk 请求 2w 条为例，耗时对比：

	优化前	优化后
单次 bulk 请求（2w）	22s	<4s



参考：

https://www.zhihu.com/question/44976788

https://www.easyice.cn/archives/207