

1、深度解析 _search 搜索结果

```
1 {
2   "took": 3,
3   "timed_out": false,
4   "_shards": {
5     "total": 11,
6     "successful": 11,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": 8,
12    "max_score": 1,
13    "hits": [
14      {
15        "_index": "test_index",
16        "_type": "test_type",
17        "_id": "5",
18        "_score": 1,
19        "_source": {
20          "name": "test_doc5"
21        }
22      }
23    ]
24  }
25 }
26 以上数据有删减
```

took: 整个搜索请求话费多少毫秒

hits.total: 本次搜索, 返回了几条数据

hits.max_score: 本次搜索的所有结果中, 最大的相关分数是多少, 每一天 document 对于 search 的相关度, 越相关, _score 分数越大, 排位越靠前

hits.hits: 前10条, 完整数据, _score降序排列, max_score

shards: shards fail的条件 (primary 和 replica全部挂掉), 不影响其他shard, 默认情况下来说, 一个搜索请求, 会打到一个 index 的所有 primary shard上去, 当然了, 每个 primary shard都可能会有一个或多个 replica shard, 所以请求 也可以到 primary shard 的其中一个 replica shard 上去。

2、剖析 timeout 机制

timeout: 默认无 timeout, latency 平衡 completeness, 手动指定 timeout, timeout 查询执行 (timeout=10ms, timeout=1s, timeout=1m)

我们有些搜索应用, 对时间是很敏感的, 比如说我的电商网站, 你不能说让用户等10分钟, 才能等到一次搜索请求的结果, 如果那样的话, 人家早就走了, 不来买东西了。

timeout机制, 指定每个shard, 就只能在timeout时间范围内, 将搜索到的部分数据 (也可能全都搜索到了), 直接理解返回给client程序, 而不是等到所有的数据全都搜索出来以后再返回。

确保说, 一次搜索请求可以在用户指定的timeout时长内完成。为一些时间敏感的搜索应用提供良好的支持。

默认情况下, 没有所谓的 timeout, 比如说, 如果你的搜索特别特别慢, 每个shard都要花好几分钟才能查询出来所有的数据, 那么你的搜索请求也会等待好几分钟之后才会返回

