Profile API

- 用法/Usage

原文链接：https://www.elastic.co/guide/en/elasticsearch/reference/current/search-profile.html

译文链接：Profile API

贡献者：王晗

> 此功能是实验性功能，可能在未来的版本中完全更改或删除。

Profile
API提供了在搜索请求中执行单个组件的详细时间信息。它让用户了解在底层如何执行搜索请求，这样用户可以理解为什么某些请求是缓慢的，并采取措施改善他们。

Profile API的输出非常详细，特别是对于跨多个分片的复杂请求执行。推荐使用pretty打印响应信息，这样有助于理解输出结果。

用法/Usage

任意_search请求可以通过添加一个顶级profile参数来实现概要描述。

```
curl -XGET 'localhost:9200/_search?pretty' -H 'Content-Type:
application/json' -d'
{
  "profile": true, 1
  "query" : {
    "match" : { "message" : "message number" }
  }
}
'
```

（1）设置顶级profile参数为true，开启搜索概要描述

这将产生以下结果：

```
 {
   "took": 25,
   "timed_out": false,
   "_shards": {
      "total": 1,
      "successful": 1,
      "failed": 0
   },
   "hits": {
      "total": 4,
      "max_score": 0.5093388,
      "hits": [...]
   },
   "profile": {
     "shards": [
        {
           "id": "[2aE02wS1R8q_QFnYu6vDVQ][twitter][1]",
           "searches": [
              {
                 "query": [
```

```
{
   "type": "BooleanQuery",
   "description": "message:message message:number",
   "time": "1.873811000ms",
   "time_in_nanos": "1873811",
   "breakdown": {
      "score": 51306,
      "score_count": 4,
      "build_scorer": 2935582,
      "build_scorer_count": 1,
      "match": 0,
      "match_count": 0,
      "create_weight": 919297,
      "create_weight_count": 1,
      "next_doc": 53876,
      "next_doc_count": 5,
      "advance": 0,
      "advance_count": 0
   },
   "children": [
      {
         "type": "TermQuery",
         "description": "message:message",
         "time": "0.3919430000ms",
         "time_in_nanos": "391943",
         "breakdown": {
            "score": 28776,
            "score_count": 4,
            "build_scorer": 784451,
            "build_scorer_count": 1,
            "match": 0,
            "match_count": 0,
            "create_weight": 1669564,
            "create_weight_count": 1,
            "next_doc": 10111,
            "next_doc_count": 5,
            "advance": 0,
            "advance_count": 0
         }
      },
      {
         "type": "TermQuery",
         "description": "message:number",
         "time": "0.2106820000ms",
         "time_in_nanos": "210682",
         "breakdown": {
            "score": 4552,
            "score_count": 4,
            "build_scorer": 42602,
            "build_scorer_count": 1,
            "match": 0,
            "match_count": 0,
            "create_weight": 89323,
```

```
                                "create_weight_count": 1,
                                "next_doc": 2852,
                                "next_doc_count": 5,
                                "advance": 0,
                                "advance_count": 0
                             }
                          }
                       ]
                    }
                 ],
                 "rewrite_time": 51443,
                 "collector": [
                    {
                       "name": "CancellableCollector",
                       "reason": "search_cancelled",
                       "time": "0.3043110000ms",
                       "time_in_nanos": "304311",
                       "children": [
                         {
                           "name": "SimpleTopScoreDocCollector",
                           "reason": "search_top_hits",
                           "time": "0.03227300000ms",
                           "time_in_nanos": "32273"
                         }
                       ]
                    }
                 ]
              }
           ],
           "aggregations": []
        }
     ]
  }
```

```
}
```

（1）返回的搜索结果，为简便起见，这里省略

即使对于一个简单的查询，响应过程也是相对复杂的。在深入更复杂的例子之前，让我们先全面剖析它。

首先，profile响应的整体结构如下:

```
{
   "profile": {

        "shards": [

           {
              "id": "[2aE02wS1R8q_QFnYu6vDVQ][twitter][1]",  (1)
              "searches": [
                    {
        "query": [...],           (2)
        "rewrite_time": 51443,    (3)
        "collector": [...]        (4)
                    }

                 ],
              "aggregations": [...]                      (5)
           }
        ]
     }
}
```

(1)profile返回参与响应的每一个分片，这些分片由唯一ID标识
(2)每个概要都包含关于查询执行的详细信息部分
(3)每个概要都有一个单独的rewrite_time累计时间。
(4)每个概要还包含关于运行搜索的lucene Collector部分
(5)每个概要都包含有关聚合执行的详细信息部分

因为一个搜索请求可能在一个或多个索引分片上执行，并且搜索范围覆盖一个或多个索引，profile的响应中顶层元素是一个shard对象数组。每个分片对象列表都列出唯一标识分片的id。ID的格式是[nodeID][indexName][shardID]

profile本身可能包含一个或多个"searches"字段，其中每个搜索search是针对底层Lucene索引执行的查询。用户提交的大多数搜索请求只会执行对Lucene索引的一个search。但偶尔也会执行多个搜索searches，如包括全局聚合(这需要执行第二个“match_all”查询全局上下文)。

在每个搜索对象里，会有两个概要信息的数组：query 数组和collector 数组。与搜索对象并肩的是一个聚合aggregations 对象，它包含聚合的概要信息。在未来，可以添加更多的部分，如建议 suggest，高亮highlight等。

这也会有一个rewrite 度量显示重写查询的总时间 (以纳秒为单位)。