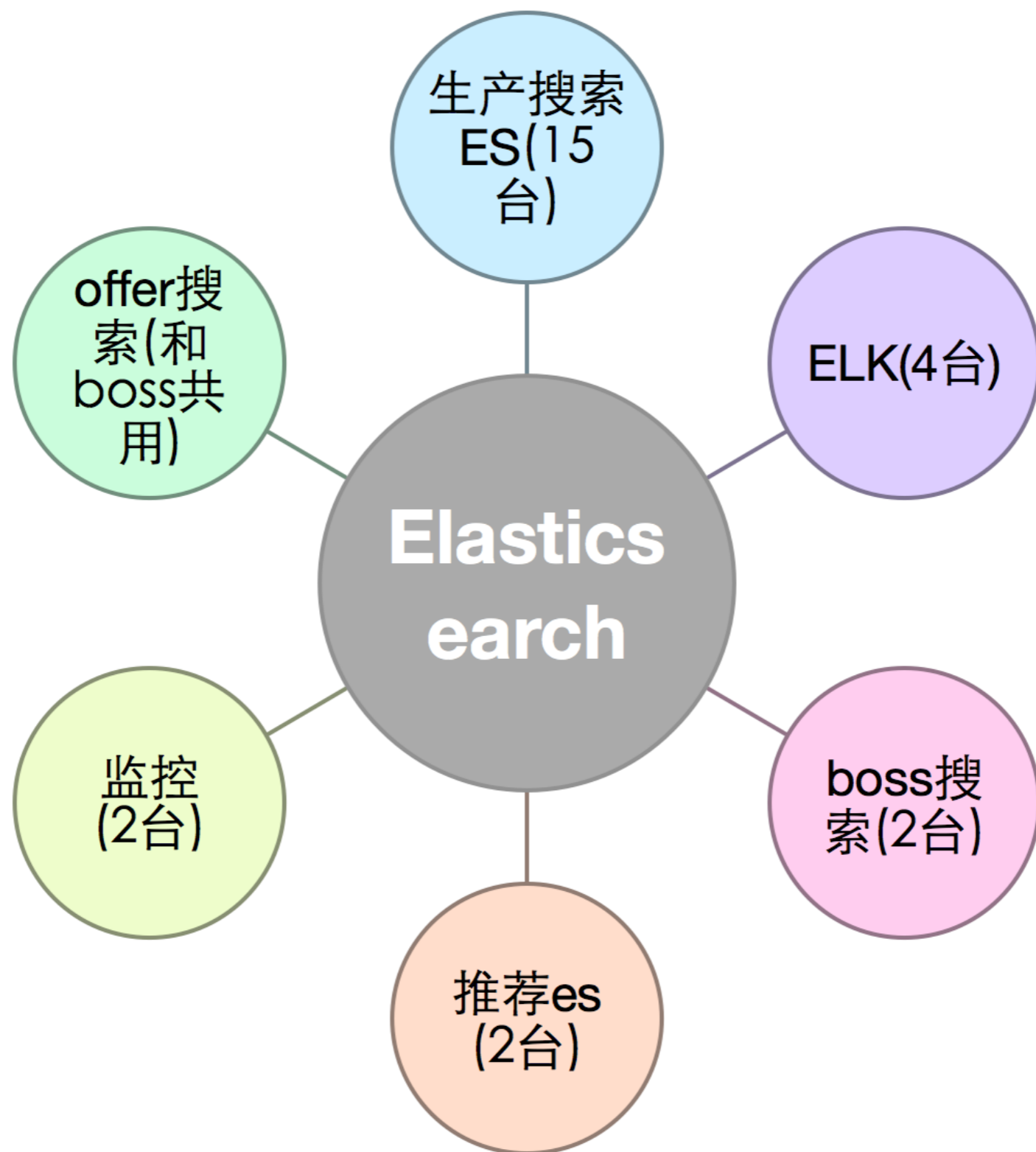




5miles 基于es的对外搜索业务实践

Agenda

- 5miles es的应用场景
- home的优化实践
- 生产数据两集群实现
- Easy-es



home场景介绍:

- 每个用户来了得计算
- 新商品和reboost商品得有合理的分布
- 用户做了reboost之后总是会去看自己的商品
- 保证home请求100%在1s内
- 无论如何保证home有商品展示

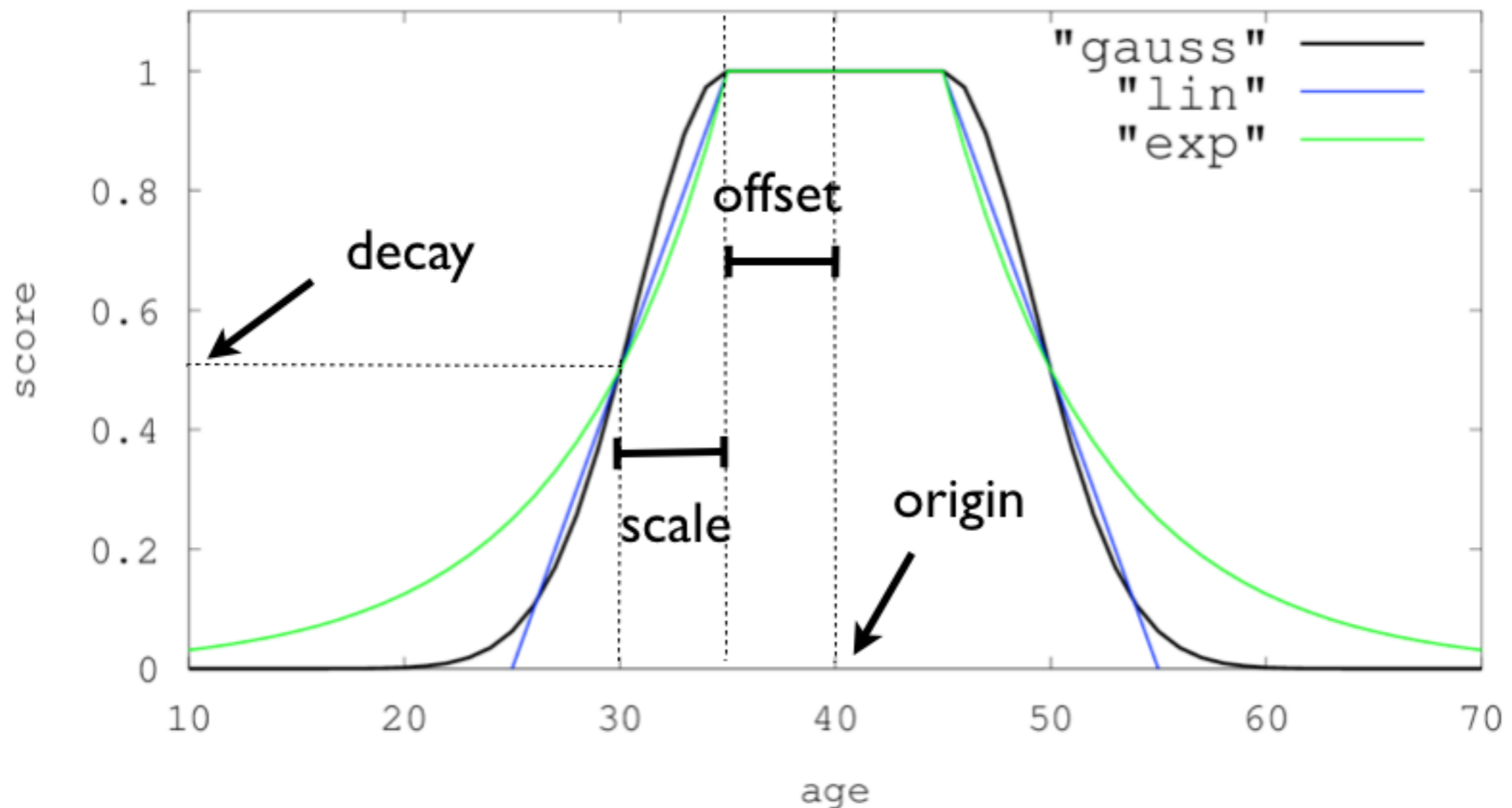


5miles的解决方案(减少计算量, 优化展示速度):

- 每个请求设置Box
- 改原先使用的distance filter 改为 geo bound box
- 异步的计算box (通过agg一步步加保证box 1万在线商品)
- 计算box的filter放所有filter第一个
- 经过数据统计, 设置300km为默认box
- 请求异步化, 当用户访问第n页的时候, 第n+1页的数据已经计算好

5miles的解决方案(实现自定义评分):

- function score guass的使用





5miles的解决方案：

<https://github.com/imotov/elasticsearch-native-script-example>

- 实现AbstractSearchScript 类，然后在run方法中返回重载的评分
- 实现NativeScriptFactory，把自定义的plugin返回
- 需要注意的事情：
 - ES重写了类型对象
 - 千万别使用SourceLookup,非常慢，但DocLookup获取的数据是分词过的，所以可能会大小写不敏感的问题



5miles的解决方案:

- 重写了distance距离计算方法(美团 地理空间距离计算优化)
- 评分都做归一化处理, 评分默认最高分为1
- gauss出现了score为0的情况, 设置minboost



5miles的优化效果对比：

优化前：

home请求占慢请求(>200ms)的75%

大于1s的请求占0.3%

优化后：

home请求占慢请求(>200ms)的6%

home 99.6%的请求在200ms内， 0.4在200-500ms,无大于1s的请求



5miles的在做的优化:

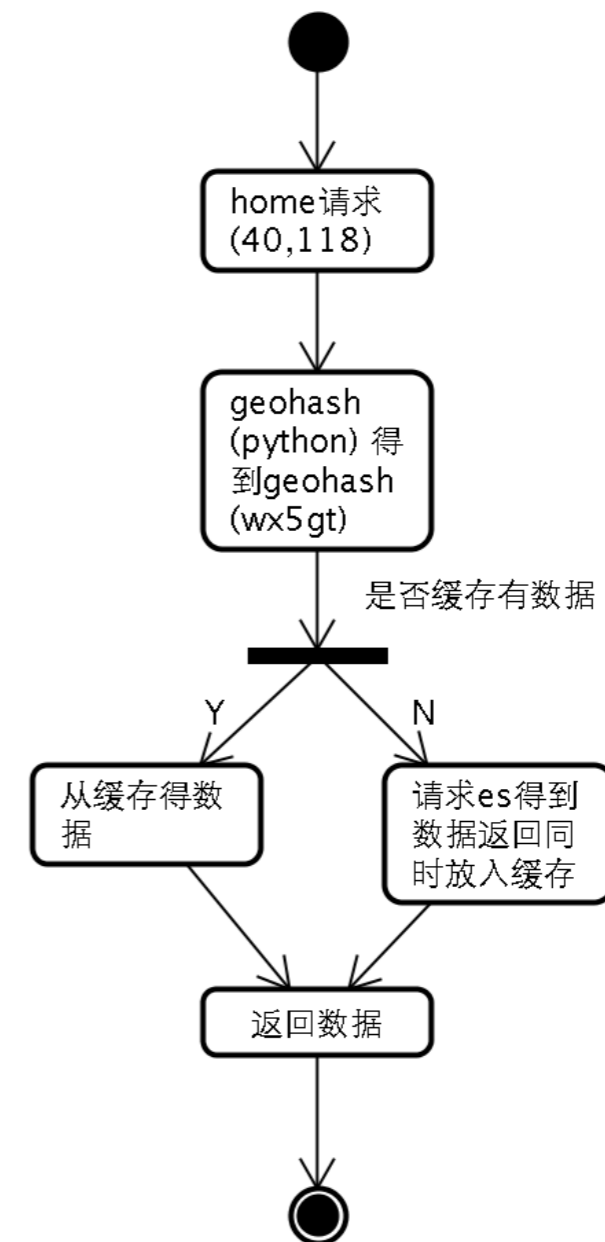
geohash 的使用

Geohash	Level	Dimensions
gc	2	~ 1,251km x 625km
gcp	3	~ 156km x 156km
gcpu	4	~ 39km x 19.5km
gcpuu	5	~ 4.9km x 4.9km
gcpuuz	6	~ 1.2km x 0.61km
gcpuuz9	7	~ 152.8m x 152.8m
gcpuuz94	8	~ 38.2m x 19.1m
gcpuuz94k	9	~ 4.78m x 4.78m
gcpuuz94kk	10	~ 1.19m x 0.60m
gcpuuz94kkp	11	~ 14.9cm x 14.9cm
gcpuuz94kkp5	12	~ 3.7cm x 1.8cm

5

5miles的在做的优化：

5miles的最小购物半径是20mi,所以我们可以假定4.9km之内的用户看到的商品是一样的



5

5miles的在做的优化:

mapping 设置

```
"location": {  
  "type": "geo_point",  
  "geohash": true,  
  "geohash_prefix": true,  
  "geohash_precision": 5  
}
```

Filter Query

```
"filter": {  
  "geohash_cell": {  
    "location": {  
      "lat": 40.718,  
      "lon": -73.983  
    },  
    "neighbors": true,  
    "precision": "2km"  
  }  
}
```



5miles生产两集群的出现：

为什么会有这样的需求

- reindex的问题（集群不能停止服务，mapping的错误设置most filed）
- 消耗性能的请求不影响主营业务的正常工作（计算密度的聚合）
- 同样的数据可能在不同的场景中需要（搜索，推荐，boss等）
- 异步的去计算一些数据供主营业务使用



5miles生产两集群的出现：

实现方式

- Reindexing Your Data (Elasticsearch: The Definitive Guide)
- scroll bulk 去增量实现(实践证明对系统性能影响很小)
- 改写了logstash的Elasticsearch Input Plugin，加入了schedule
- 根据幂等原理启动了两个logstash保证服务可靠性
- zabbix做数据的同步监控
- 定期的删除is_removed的document



Easy-es

- 我们尝试使用了SQL插件，但感觉好多请求无法转换到sql层上
- es query DSL编写很麻烦，但结构其实很简单
- 我们的API人员长期接触es，明白es的语句，但写不出来



Easy-es

<https://github.com/whybangbang/easy-es>

The screenshot shows the 'Easy Es' web application interface. At the top, there is a dark header bar with the logo '{ e } Easy Es' on the left, and buttons for 'start', 'open', 'save', and 'Document' on the right. Below the header, the main area is divided into two sections. On the left, there is a query builder interface. It starts with a minus sign and the text 'Elasticsearch' followed by a plus sign. Below this, there are two main sections: 'query' and 'filter'. The 'query' section has a minus sign, the word 'query', and a plus sign. Underneath it is a 'term' section with a minus sign, the word 'term', and an 'X' button. Below 'term' is a field 'author' with a value 'seaasun' and an 'X' button. The 'filter' section has a minus sign, the word 'filter', and a plus sign. Underneath it is a 'range' section with a minus sign, the word 'range', and an 'X' button. Below 'range' is an 'age' section with a minus sign, the word 'age', and two input fields. The first field is 'gte' with a value '1' and an 'X' button. The second field is 'lte' with a value '1' and an 'X' button. On the right side of the interface, there are two tabs: 'formate&validator' (selected) and 'cope code'. The 'formate&validator' tab shows a JSON output of the query:

```
{  "query": {    "term": {      "author": "seaasun"    }  },  "filter": {    "range": {      "age": {        "gte": "1"      }    }  }}
```



Q & A

5miles招聘邮箱: lv yi@wespoke.com