电商系统如何做搜索引擎？

李伟山

作者：李伟山，历任华为，阿里技术专家，米么技术总监，在分布式，大容量高并发等领域，有一定积累。

来自：米么骚客

搜索引擎(search engine)是指根据一定的策略、运用特定的计算机程序搜集互联网上的信息，在对信息进行组织和处理后，为用户提供检索服务的系统。数据其实就是一块块的砖头，当用户需要的时候搜索引擎搬过来。我们的宗旨就是在最短的时间内，让用户找到他们最想要的东西。

电商搜索业务特点：

第一点，　电商系统的商品数量『庞大』，搜索页的PV高。某宝2013年有7亿线上商品，　List的PV7亿+相当于每秒有 8000个请求

第二点，　电商的搜索引擎并没有爬虫系统，因为所有的数据都是结构化的，一般都是Mysql或者 Oracle 的数据库，所以不用像百度一样用『爬虫』去不断去别的网站找内容，当然，电商其实也有自己的『爬虫』系统，一般都是抓取友商的价格，再对自己进行调整。

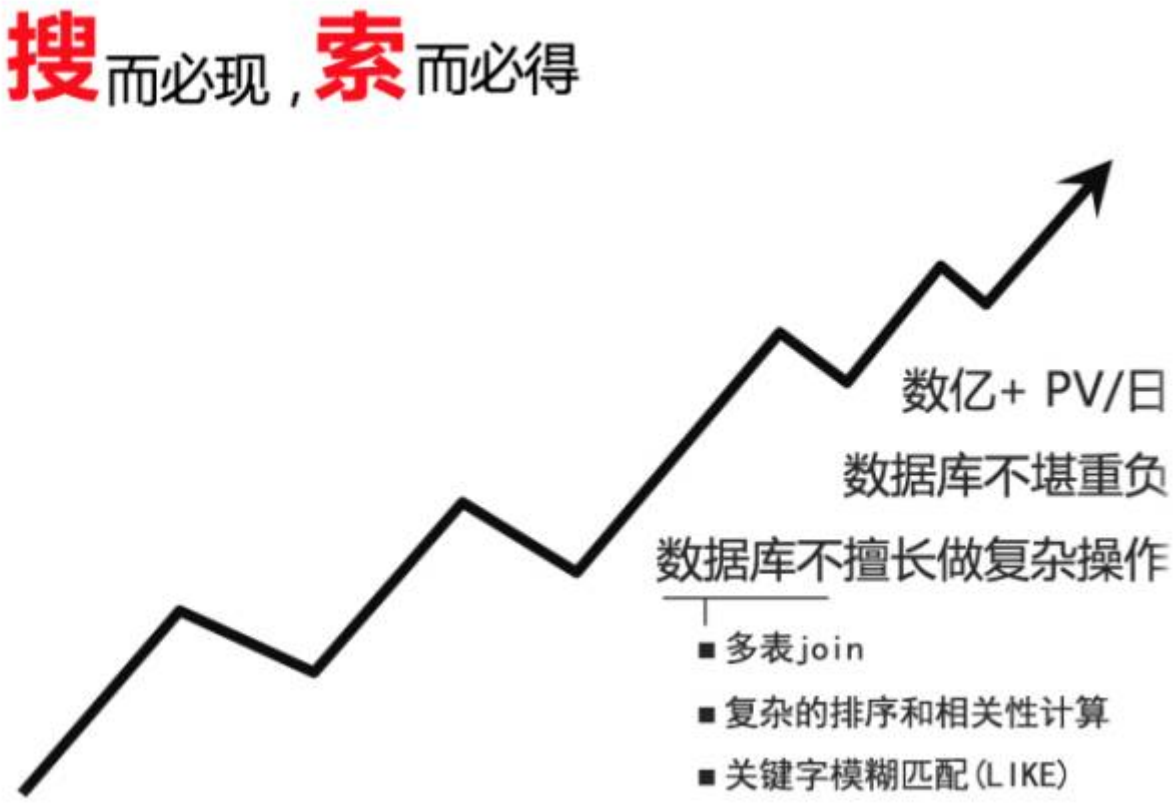
第三点，　电商搜索引擎的过滤功能其实比搜索功能要常用，甚至大于搜索本身。什么是过滤功能？一般我们网站买东西的时候，搜了一个关健词，比如运动鞋，然后所有相关品牌或者其他分类的选择就会呈现在我们面前。对百度而言，搜什么词就是什么词，如果是新闻的话，可能在时间上会有一个过滤的选项。

第四点，　电商搜索引擎支持各种维度的排序，包括支持人气、销量、信用、价格、发货地等属性的排序，且对数据的实时性要求非常高。对一般的搜索引擎，只有非常重要的网站，比如一些重量级的门户网站，百度的收录是非常快的，但是对那些流量很小的网站，可能一个月才会爬一次。电商搜索对数据的实时性要求主要体现在价格和库存两个方面。

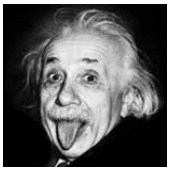
第五点，　电商搜索引擎的效果不仅要考虑买家（信息消费方，结果多样性），还得考虑卖家（信息提供方，曝光率）。

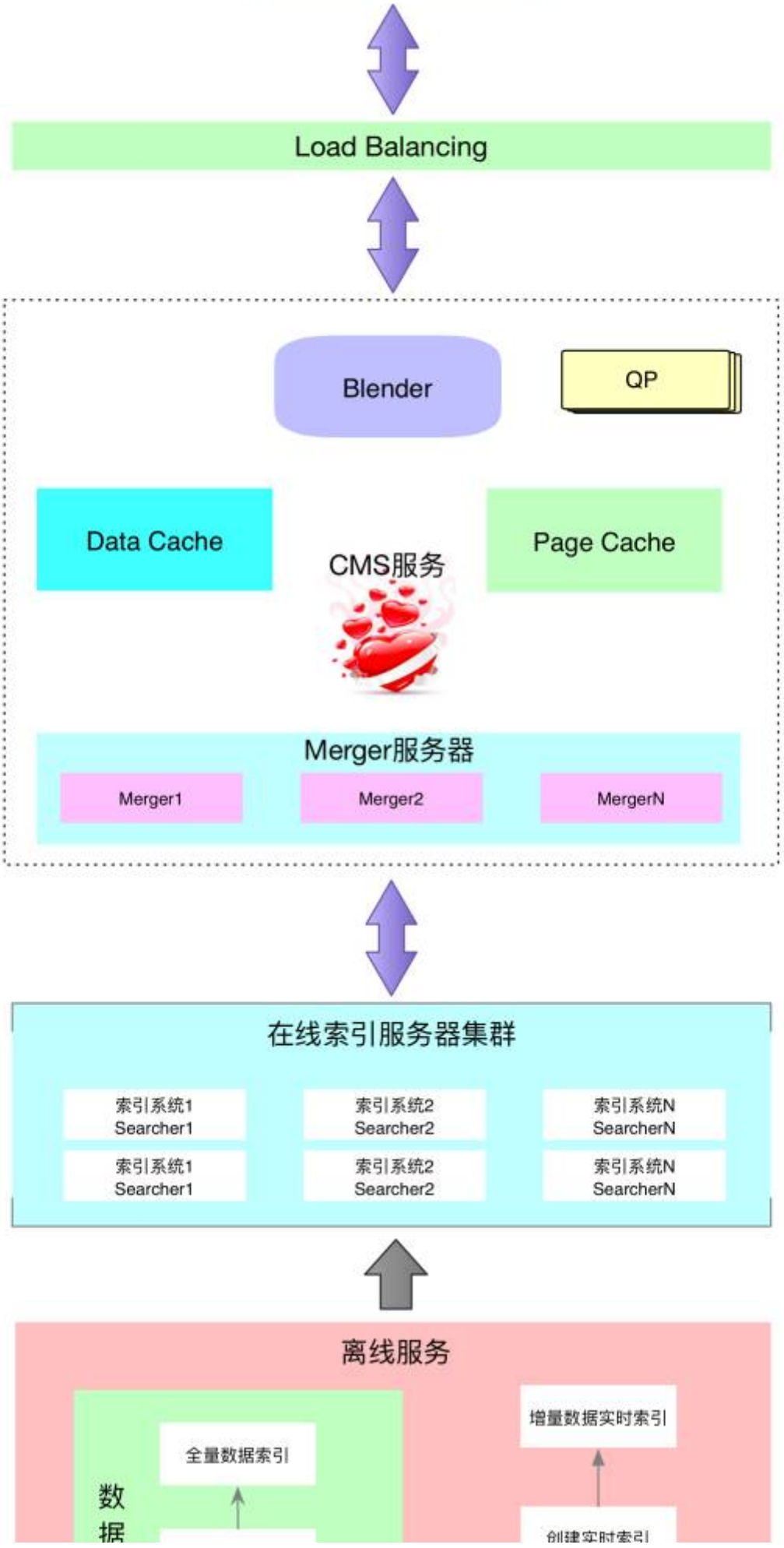
第六点，　电商搜索引擎另一个特点就是不能丢品，比如我们在淘宝、天猫开了个店铺，然后好不容易搞了一次活动，但是却搜不到了，这是无法忍受的。除此之外，电商搜索引擎与推荐系统和广告系统是相互融合的，因为搜索引擎对流量的贡献是最大的，所以大家都希望把广告系统能跟其融合。

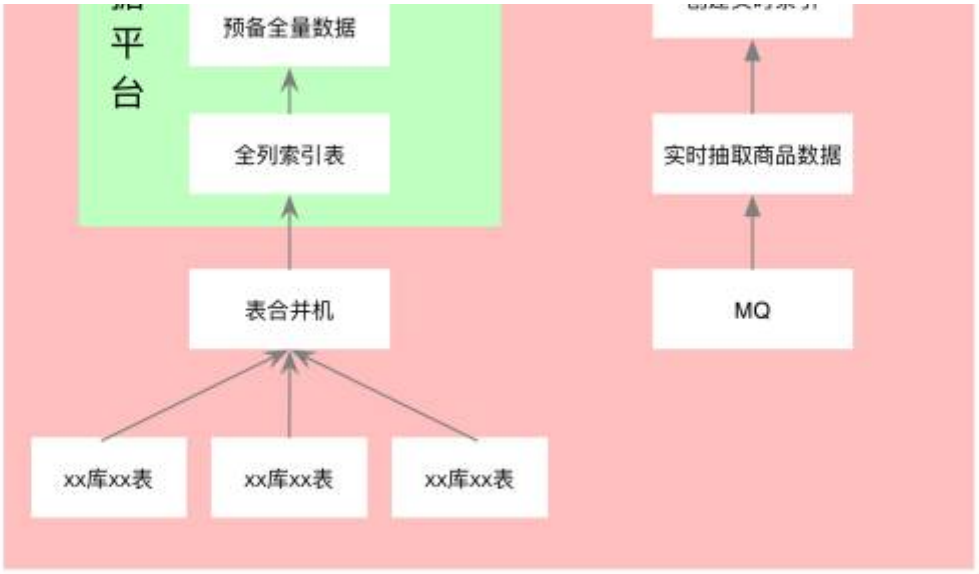
第七点，　保证高可用，容灾、异常保护、降级(降级：QPS维度、在Clustermap上来做，正常来说，我们有20列，如果系统负载高的话查询只分布到10列，这样就高了1倍的QPS) 。异常保护：Latency 、在Searcher上来做，如果系统负载较高的话，Searcher上会直接丢弃一些耗时的Query。



综上所述，电商系统中搜索引擎的必要性显而易见。




Load Balancing
Blender
QP
Data Cache
CMS服务
Page Cache
Merger服务器
Merger1
Merger2
MergerN
在线索引服务器集群
索引系统1
Searcher1
索引系统2
Searcher2
索引系统N
SearcherN
索引系统1
Searcher1
索引系统2
Searcher2
索引系统N
SearcherN
离线服务
全量数据索引
数
据
增量数据实时索引



该系统真正接受用户请求并响应的系统。为了用户体验的需要，首先增加Query Processor服务，负责查询意图分析提升搜索的准确性。随着访问量的增长，接着增加缓存模块，提升请求处理性能。接着随着数据量（商品量）的增长，将CMS服务从检索服务中独立出去，成为Detail服务。数据量的进一步增长，对数据进行类似数据库分库分表的分片操作。这时候，在线检索服务由多个分片的Searcher列组成。自然而然，需要一个Merger服务，将多个分片的结果进行合并。

1. 客户端请求通过Load Blance到Blender；

2. Blender调用QP，QP调用运营平台，其中运营平台主要负责将日常运营数据服务化，QP负责分析Query；

3. Blender请求Page Cache和Data Cache同时请求Merger调用在线搜索服务；

4. Merger调用UserInfoSystem获取用户标签信息；

5. Merger将请求发给每列Searcher；

6. 每个Searcher召回商品并返给Merger；

7. Merger合并多列searcher的结果，确定需要输出的商品，请求CMS封装对应的商品信息；

8. CMS封装商品信息返给Merger；

9. Merger将包装好的商品返给Blender；

10. Blender将Merger返回的结果与其他垂直搜索结果进行合并，最终返回给前端。

为了保证高召回率和低响应延时，搜索服务流程的处理全部放在内存当中进行计算。多个Searcher并发处理请求，同时单个Searcher内部采用线程池技术，即所有线程之间共享倒排索引和商品属性信息，提高内存使用效率；每个查询使用一个独立线程串行执行，保证并发的多个查询线程之间互不影响。此外通过合理的设置线程池的大小，保证系统的CPU资源得到充分利用。并且采用多级缓存来保证系统的高可用。

Page Cache：由于搜索符合互联网的二八法则，20%热门查询频度非常高，占每天搜索请求量80%。针对这一特点，搜索第一级缓存以查询请求为Key，将返回给

用户的页面作为Value。对于完全相同的请求，直接从缓存返回结果。页面缓存策略上线伊始，缓存命中率就接近了30%，基本解决了当时的性能问题。

Data Cache：随着业务的发展，排序结果需要针对不同用户实现个性化订制，这就导致请求中会包含用户的UserInfo。如果直接将UserInfo放入缓存作为Key，会导致Data Cache的key数量激增，不但需要超大的缓存空间，同时缓存的命中率也会极低，最终会导致线上个性化服务的体验满意度降低。为了解决这个问题，将UserInfo加入Key，但是Value只保存排序好的商品ID，这样需要的缓存空间远远小于Data Cache。当命中缓存后，调用CMS直接进行结果包装。为了进一步提高缓存命中率，利用用户搜索的翻页习惯，即离线统计出用户的翻页数最大值，然后在Value中缓存这些页面涉及到所有的商品ID，从实践效果来看，用户后续的翻页请求大部分会命中Cache。

该系统是搜索技术的核心，在进入这个系统之前，搜索信息仍然是以商品维度进行存储的。索引系统负责生成一种以关键字维度进行存储的信息，一般称之为倒排索引。系统对于全量和增量的处理是一致的，唯一的区别在于待处理数据量的差异。一般情况下，全量数据索引由于数据量庞大，采用Hadoop进行；实时数据量小，采用单机进行索引生产。

倒排索引

倒排索引（英语：Inverted index），也常被称为反向索引、置入档案或反向档案，是一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文

档中的存储位置的映射，它是文档检索系统中最常用的数据结构。

有两种不同的反向索引形式：

一条记录的水平反向索引（或者反向档案索引）包含每个引用单词的文档的列表。

一个单词的水平反向索引（或者完全反向索引）又包含每个单词在一个文档中的位置。

后者的形式提供了更多的兼容性（比如短语搜索），但是需要更多的时间和空间来创建。

(REF:http://zh.wikipedia.org/wiki/%E5%80%92%E6%8E%92%E7%B4%A2%E5%BC%95)

如何构建倒排索引

E.g. 被索引的文本：

T0 = "it is what it is"

T1 = "what is it"

T2 = "it is a banana"

得到的倒排索引：

"a": {2}

"banana": {2}

"is": {0, 1, 2}

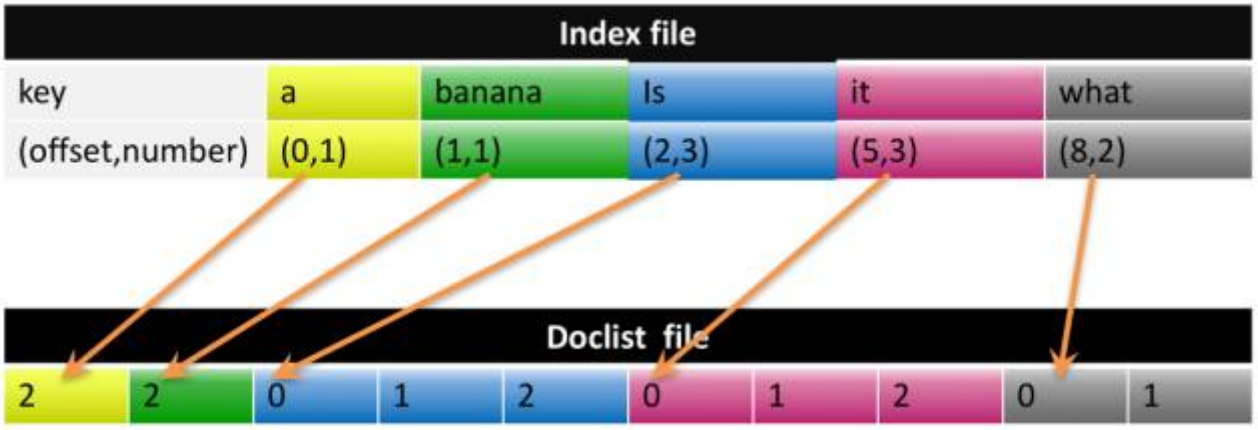
"it": {0, 1, 2}

"what": {0, 1}

检索的条件"what","is"和"it"将对应集合的交集。

正向索引开发出来用来存储每个文档的单词的列表。正向索引的查询往往满足每个文档有序频繁的全文查询和每个单词在校验文档中的验证这样的查询。在正向索引中，文档占据了中心的位置，每个文档指向了一个它所包含的索引项的序列。也就是说文档指向了它包含的那些单词，而反向索引则是单词指向了包含它的文档，很容易看到这个反向的关系。

简单索引的文件格式



Index file可以采用闭链Hash的结构来存储，这样查询效率会很高，但是空间利用率很低。也可以对Key做排序后顺序存储，查询时使用二分查找。查询效率较低，但是不会浪费内存。Doclist file的存储就很简单了，整体来看就是一个Int型的数组，为了提高内存利用率，通常还会对Doclist进行压缩。

问题1：

假如有7亿的宝贝，其中有1亿宝贝标题中包含”正品”这个词，那么正品这个词的倒排链长度就是1亿

解决方案：

1，索引压缩

要求：在解压速度快的基础上，压缩比尽量高

2，索引截断

要求：不影响用户体验的前提下，倒排链尽量短

正排索引

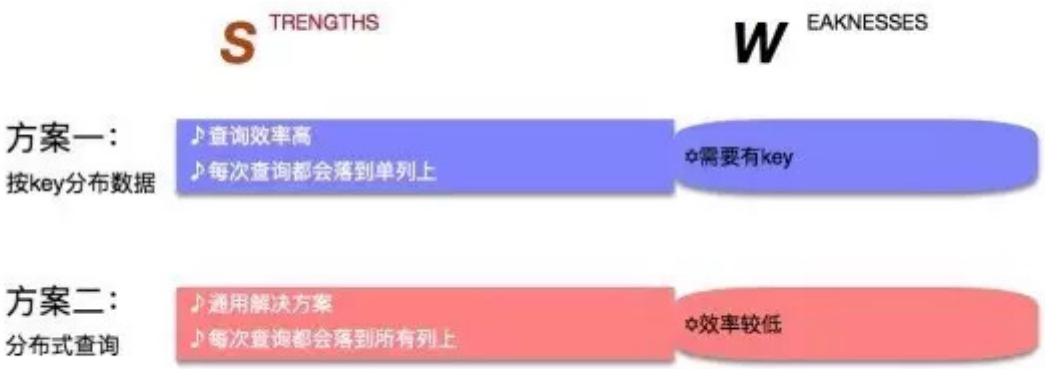
一种索引方法，被用来存储在全文搜索下某个文档ID与其对应的部分字段存储位置的映射。正排表是以文档的ID为关键字，表中记录文档中每个字的位置信息，查找时扫描表中每个文档中字的信息直到找出所有包含查询关键字的文档。

id	Category	Starts	User_id	Reserve_price	Isb
39512	39512	526.22	39512	33.5	1

问题2：

1，单台机器内存资源有限，如何容纳更多的文档？

2，高并发下如何快速响应请求？



多行用于做负载均衡、冗余备份。如果我们一行能承担1000的QPS，那么10行就能承担1000*10的QPS，多列用于提高索引量。如果我们一台机器受限于内存只能放1000w的宝贝，那么1亿的宝贝就需要10台机器，也就是10列。

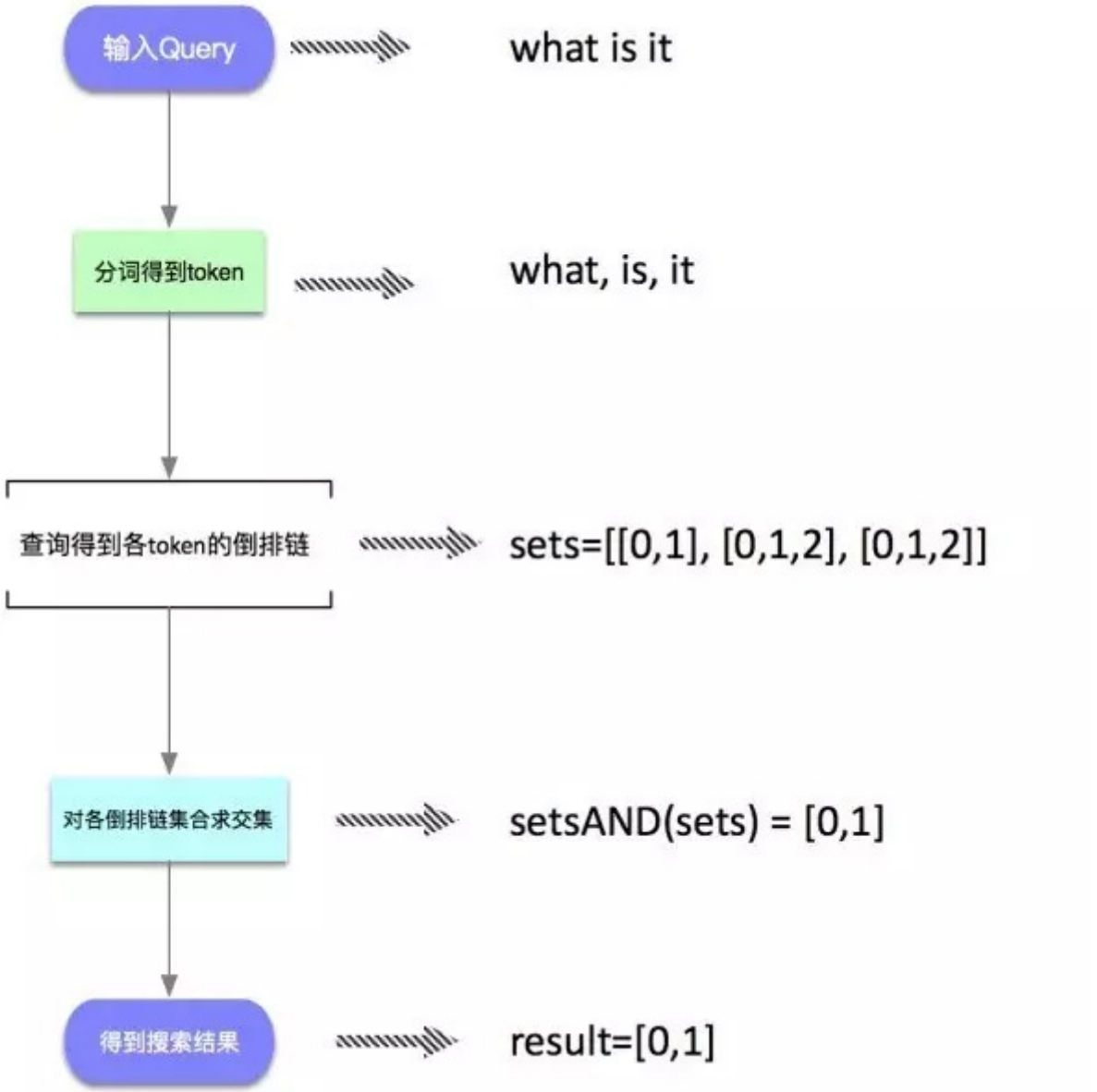
在分布式集群里，通常会有多行多列。当然如果数据量足够小，可以只有一列，但是考虑到容灾备份，就算流量非常低也会有至少2行。

方案一：个性化搜索

方案二：主搜索

Ksearch与Isearch的区别：Ksearch是按Key来分布数据，而Isearch虽然也支持按Key查询，但是主要功能是分布式。

检索过程



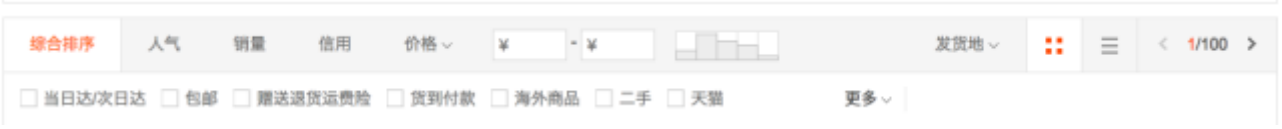
问题3：为什么做OR查询会比较影响性能？

1，OR操作分配的内存空间不可预估，而且会很大

2，合并的时候要遍历所有的倒排链

3，倒排链越长意味着最后排序的数据就越多

过滤统计排序



问题4：在分布式的集群中，假设我们要取销量排序第三页的宝贝，即s=80&n=40。要如何获取？

需要考虑的问题：

我们的宝贝分布在不同的机器上

排序效率要高

也许某台机器上销量最差的宝贝也比其它机器上销量最好的宝贝要好

解决方案：

每台Searcher机器返回Top120条结果Merger再将所有机器返回的Top120宝贝混到一起，再Heapsort取出Top 120

Rank

常见问题：

在人气，或者销量排序的时候马太效应怎么解决？

马太效应：宝贝销量越高排名越靠前，越靠前销量越好

首先：宝贝销量越高，确实搜索的排名会靠前，但是排名越靠前，只能说明搜索引导的成交会高一些，但是不会在大程度上影响宝贝的销量，因为搜索本身带来的成交才20-30%

常用排序规则

阿基米德排序：

相关性(标题类目 亿级) + 下架时间(ends 千万级) + 宝贝人气(百万级) + 卖家质量分(十万)

人气排序：

标题 （1）+ 类目（1）+ 宝贝人气（1）+ 卖（0.2）

肯定不是用户想要的

单维度排序：

按照：销量，信用，价格，总价，单价排序

单维度排序下不会做打散，可以认为这是用户的意愿只想找价格低的

类目混排：

类目混排会对宝贝按类目进行分档，档内才按价格排序

打散：卖家打散，款式打散

Anti-spam

降权与屏蔽

常见的作弊类型

为了提升用户体验，保证搜索的公平性，作弊是卖家通过一些不正当的行为，提高自己的搜索排名。

虚假交易，包括炒作信用和炒作销量。以增加“会员积累信用”为目的或通过炒作商品销量提高商品人气而发布的商品，会被判定为虚假交易商品。

通过发布完全相同的商品来争取更多的展现机会，直接降低了搜索的精准度，降低了消费者的购物体验，也是搜索控制的重点。

商品描述不详、无实际商品、仅提供发布者联系方式以及非商品信息的商品，判定为发布广告商品。

换宝贝：指卖家为了累积销量或人气，修改原有的商品的标题、价格、图片、详情等变成另外一种商品继续出售。

Sku作弊：滥用商品属性（如：套餐），将常规商品和瑕疵品、单机、样机、模型、二手等非常规商品放在一个宝贝里出售，且一口价为非常规商品的价格。

（注解：网站的PR值（全称为PageRank），是Google搜索排名算法中的一个组成部分，级别从1到10级，10级为满分，PR值越高说明该网页在搜索排名中的地位越重要）

商业电商搜索算法另外两个重要技术，一个是类目体系建立和应用，另一个是个性化技术。

类目体系目前主要使用机器学习的方法进行训练，个性化主要通过用户画像进行Query改写来实现。搜索算法是一个非常值得一个电商产品持续投入的技术，一方面如果技术人员要有良好的技术背景，可以借鉴很多成熟的技术，避免重复造轮子； 另一方面，每个产品的搜索都有自身的特点，需要深入研究产品的特性给出合理的解决方案。