

# 10 | 搜索引擎设计：信息搜索怎么避免大海捞针？

2022-03-09 李智慧

《李智慧·高并发架构实战课》

[课程介绍 >](#)



讲述：李智慧

时长 14:15 大小 13.06M



你好，我是李智慧。

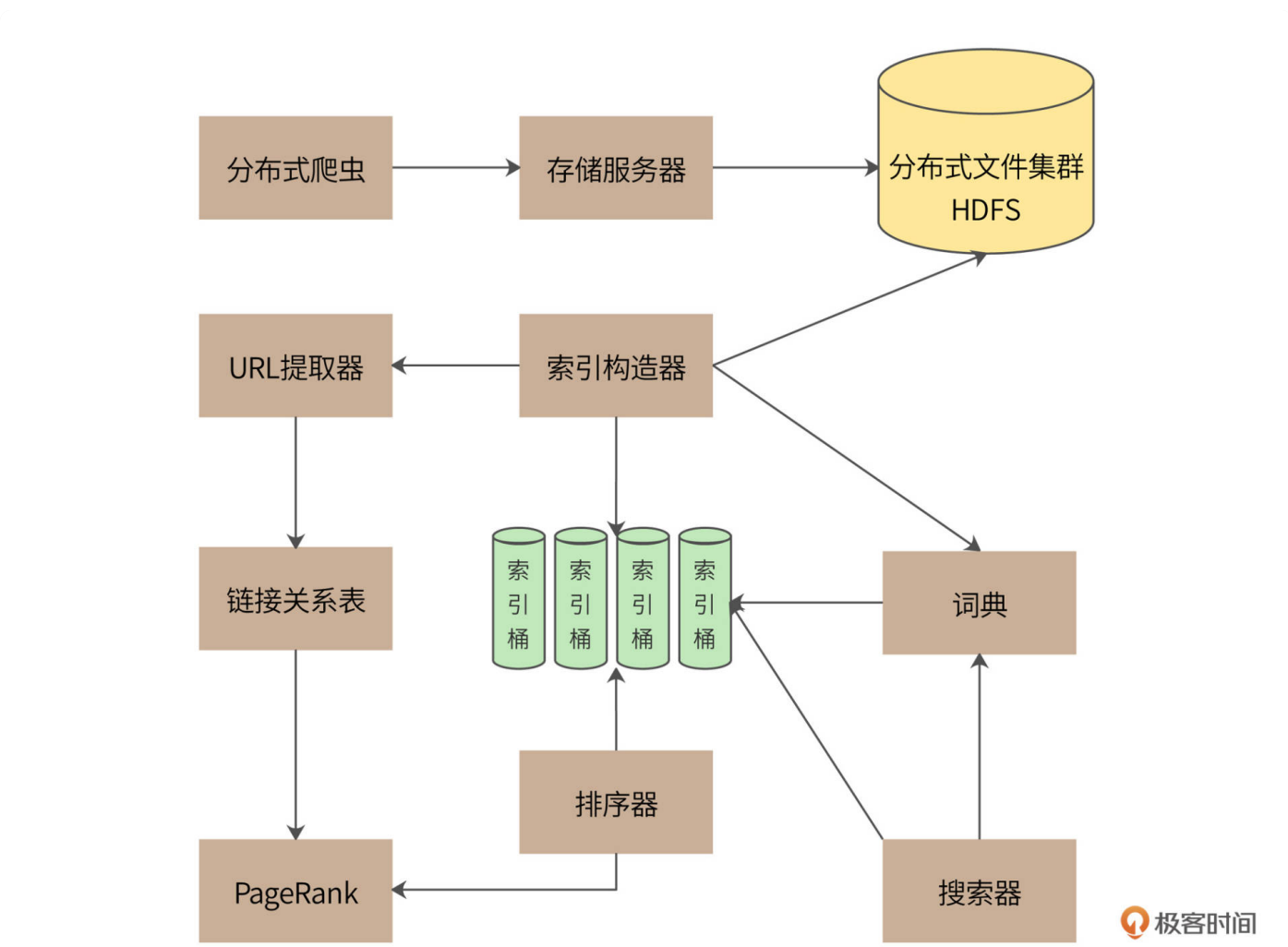
在 04 讲中，我们讨论了大型分布式网络爬虫的架构设计，但是网络爬虫只是从互联网获取信息，海量的互联网信息如何呈现给用户，还需要使用搜索引擎完成。因此，我们准备开发一个针对全网内容的搜索引擎，产品名称为“Bingoo”。

Bingoo 的主要技术挑战包括：

1. 针对爬虫获取的海量数据，如何高效地进行数据管理；
2. 当用户输入搜索词的时候，如何快速查找包含搜索词的网页内容；
3. 如何对搜索结果的网页内容进行排序，使排在搜索结果列表前面的网页，正好是用户期望看到的内容。

概要设计

一个完整的搜索引擎包括分布式爬虫、索引构造器、网页排名算法、搜索器等组成部分，Bingoo 的系统架构如下。



分布式爬虫通过存储服务器将爬取的网页存储到分布式文件集群 HDFS，为了提高存储效率，网页将被压缩后存储。存储的时候，网页一个文件挨着一个文件地连续存储，存储格式如下。

docID	URL长度	数据长度（压缩后）	URL	网页内容数据（压缩后）
-------	-------	-----------	-----	-------------

每个网页被分配得到一个 8 字节长整型 docID，docID 之后用 2 个字节记录网页的 URL 的长度，之后 4 个字节记录压缩后网页内容数据的长度，所有存储的网页的头 14 个字节都是同样的格式。之后存储 URL 字符串和压缩后的网页内容数据。读取文件的时候，先读 14 个字节的头信息，根据头信息中记录的 URL 长度和数据长度，再读取对应长度的 URL 和网页内容数据。

搜索引擎能够快速查找的核心就是利用索引，根据用户的查询内容查找匹配的索引，根据索引列表构建结果页面。索引的构造主要通过索引构造器完成，索引构造器读取 HDFS 中的网页内容，解压缩后提取网页中的单词，构建一个“docID-> 单词列表”的正排索引。然后，索引构造器再根据这个正排索引构建一个“单词 ->docID 列表”的倒排索引，“docID 列表”就是包含了这个单词的所有网页列表。利用这个倒排索引，搜索器可以快速获得用户搜索词对应的所有网页。

网页中所有的单词构成了一个词典，实际上，词典就是一个 Hash 表，key 就是单词，value 就是倒排索引的网页列表。虽然互联网页的内容非常庞大，但是使用到的单词其实是非常有限的。根据 Google 的报告，256M 内存可以存放 1400 万个单词，这差不多就是英文单词的全部了。

在构建索引的过程中，因为要不断修改索引列表，还要进行排序，所以，有很多操作是需要进行加锁同步完成的。对于海量的互联网页的计算，这样的索引构建速度太慢了。因此我们设计了 64 个索引桶，根据 docID 取模，将不同网页分配到不同的桶中，在每个桶中分别进行索引构建，通过并行计算来加快索引处理速度。

索引构造器在读取网页内容、构造索引的时候，还会调用 URL 提取器，将网页中包含的 URL 提取出来，构建一个链接关系表。链接关系表的格式是“docID->docID”，前一个 docID 是当前网页的 docID，后一个 docID 是当前网页中包含的 URL 对应的 docID。一个网页中会包含很多个 URL，也就是会构建出很多个这样的链接关系。后面会利用这个链接关系表，使用 PageRank 排名算法对所有网页进行打分排名，当索引器得到查找的网页列表时，利用 PageRank 值进行排名，最终呈现给用户，保证用户最先看到的网页是最接近用户期望的结果页面。

## 详细设计

一个运行良好的搜索引擎的核心技术就是索引和排名，所以我们将分别说明这两种技术要点。

### 索引

索引构造器从 HDFS 读取网页内容后，解析每个页面，提取网页里的每个单词。如果是英文，那么每个单词都用空格分隔，比较容易；如果是中文，需要使用中文分词器才能提取到每个单词，比如“高并发架构”，使用中文分词器得到的就是“高并发”、“架构”两个词。

首先，索引构造器将所有的网页都读取完，构建出所有的“docID-> 单词列表”正排索引。

1	架构，分布式，缓存，队列，微服务
2	高并发，架构，设计，文档，写作

然后遍历所有的正排索引，再按照“单词 → docID 列表”的方式组织起来，就是倒排索引了。

单词	网页列表
高并发	2、4、5、7
架构	1、2、4

我们这个例子中只有两个单词、7 个网页。事实上，Bingoo 数以千亿的网页就是这样通过倒排索引组织起来的，网页数量虽然庞大，但是单词数却是比较有限的。所以，整个倒排索引的大小相比于网页数量要小得多。Bingoo 将每个单词对应的网页列表存储在硬盘中，而单词则存储在内存的 Hash 表，也就是词典中，词典示例：

单词	网页列表地址
高并发	0xa46fc960
架构	0x8a8f29be

对于部分热门的单词，整个网页列表也可以存储在内存中，相当于缓存。在词典中，每个单词记录下硬盘或者内存中的网页列表地址，这样只要搜索单词，就可以快速得到对应的网页地址列表。Bingoo 根据列表中的网页编号 docID，展示对应的网页信息摘要，就完成了海量数据的快速检索。

如果用户的搜索词正好是一个单词，比如“高并发”，那么直接查找词典，得到网页列表就完成查找了。但是如果用户输入的是一个句话，那么搜索器就需要将这句话拆分成几个单词，然后分别查找倒排索引。这样的话，得到的就是几个网页列表，还需要对这几个网页列表求交集，才能得到最终的结果列表。

比如，用户输入“高并发架构”进行搜索，那么搜索器就会拆分成两个词：“高并发”、“架构”，得到两个倒排索引：

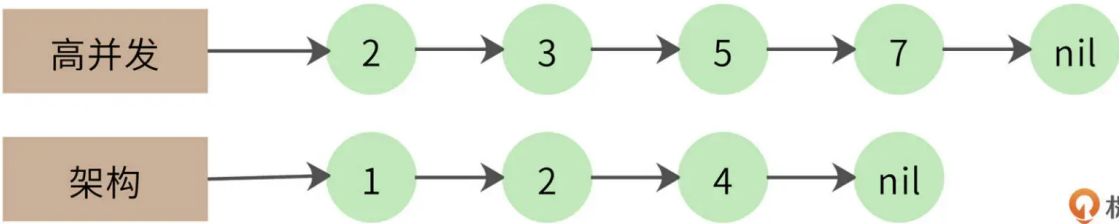
高并发 ->2,3,5,7

架构 ->1,2,4

需要对这两个倒排索引求交集，也就是同时包含“高并发”和“架构”的网页才是符合搜索要求的结果，最终的交集结果应该是只有一篇网页，即 docID 为 2 的满足要求。

列表求交集最简单的实现就是双层 for 循环，但是这种算法的时间复杂度是  $O(n^2)$ ，我们的网页列表长度（n）可能有千万级甚至更高，这样的计算效率太低。

一个改进的算法是**拉链法**，我们将网页列表先按照 docID 的编号进行排序，得到的就是这样两个有序链表：

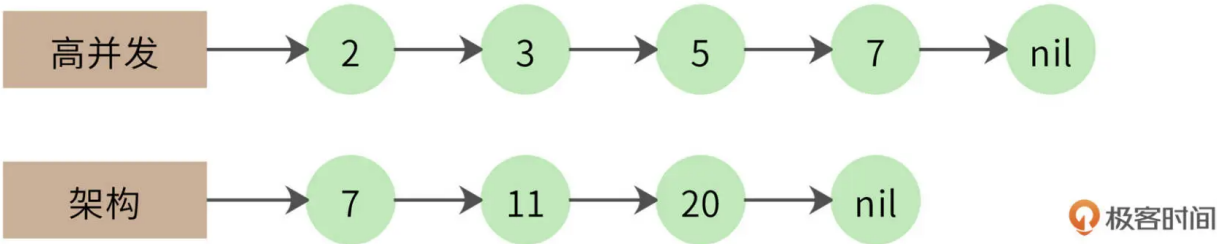


同时遍历两个链表，如果其中一个链表当前指向的元素小于另一个链表当前指向的元素，那么这个链表就继续向前遍历；如果两个链表当前指向的元素相同，该元素就是交集元素，记录在结果列表中；依此继续向前遍历，直到其中一个链表指向自己的尾部 nil。

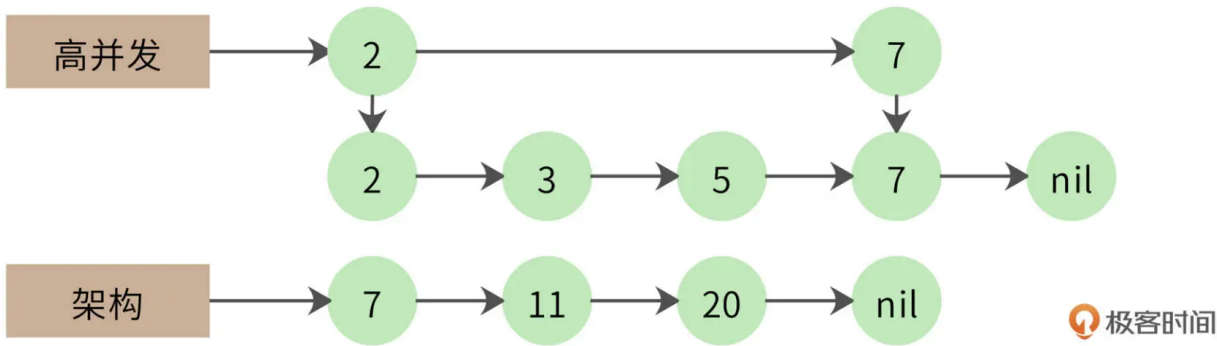
拉链法的时间复杂度是  $O(2n)$ ，远优于双层循环。但是对于千万级的数据而言，还是太慢。我们还可以采用**数据分片**的方式进行并行计算，以实现性能优化。

比如，我们的 docID 分布在[0, 1 万亿) 区间，而每个倒排索引链表平均包含 1 千万个 docID。我们把所有的 docID 按照 1 千亿进行数据分片，就会得到 10 个区间[0, 1 千亿)[1 千亿，2 千亿).....[9 千亿，1 万亿)。每个倒排索引链表大致均匀分布在这 10 个区间，我们就可以依照这 10 个区间范围，将每个要遍历的链表切分为 10 片，每片大约包含 1 百万个 docID。两个链表只在自己对应的分片内求交集即可，因此我们可以启动 10 个线程对 10 个分片进行并行计算，速度可提高 10 倍。

事实上，两个 1 千万长度的链表求交集，最终的结果可能不过几万，也就是说，大部分的比较都是不相等的。比如下面的例子。



第一个链表遍历到自己的最后一个元素，才和第二个链表的第一个元素相同。那么第一个链表能不能跳过前面那些元素呢？很自然，我们想到可以用**跳表**来实现，如下图。



**跳表实际上是在链表上构建多级索引**，在索引上遍历可以跳过底层的部分数据，我们可以利用这个特性实现链表的跳跃式比较，加快计算速度。使用跳表的交集计算时间复杂度大约是  $O(\log(n))$ 。

此外，虽然搜索引擎利用倒排索引已经能很快得到搜索结果了，但搜索引擎应用还会使用缓存对搜索进行加速，将整个搜索词对应的搜索结果直接放入缓存，以减少倒排索引的访问压力，以及不必要的集合计算。



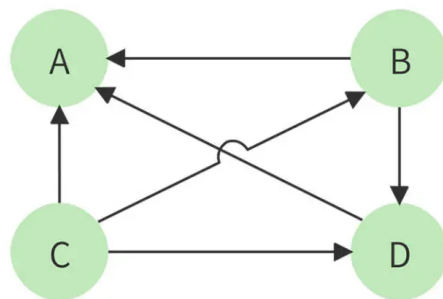
## PageRank 排名算法

Bingoo 使用 PageRank 算法进行网页结果排名，以保证搜索结果更符合用户期待。

PageRank 算法会根据网页的链接关系给网页打分。如果一个网页 A 包含另一个网页 B 的超链接，那么就认为 A 网页给 B 网页投了一票。一个网页得到的投票越多，说明自己越重要；越重要的网页给自己投票，自己也越重要。

PageRank 算法就是计算每个网页的 PageRank 值，最终的搜索结果也是以网页的 PageRank 值排序，展示给用户。事实证明，这种排名方法非常有效，PageRank 值更高的网页，确实更满足用户的搜索期望。

以下面四个网页 A、B、C、D 举例，带箭头的线条表示链接。



极客时间

B 网页包含了 A、D 两个页面的超链接，相当于 B 网页给 A、D 每个页面投了一票，如果初始的时候，所有页面都是 1 分，那么经过这次投票后，B 给了 A 和 D 每个页面 1/2 分（B 包含了 A、D 两个超链接，所以每个投票值 1/2 分），自己从 C 页面得到 1/3 分（C 包含了 A、B、D 三个页面的超链接，每个投票值 1/3 分）。

而 A 页面则从 B、C、D 分别得到 1/2，1/3，1 分。用公式表示就是

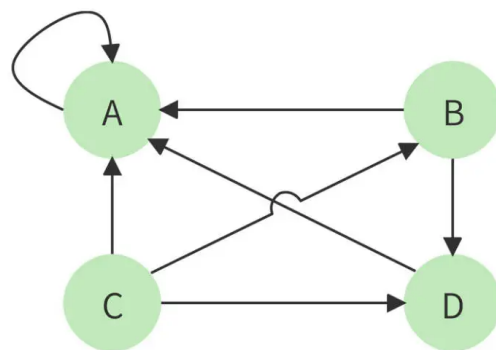
$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{3} + \frac{PR(D)}{1}$$

等号左边是经过一次投票后，A 页面的 PageRank 分值；等号右边每一项的分子是包含 A 页面超链接的页面的 PageRank 分值，分母是该页面包含的超链接数目。

这样经过一次计算后，每个页面的 PageRank 分值就会重新分配，重复同样的算法过程，经过几次计算后，根据每个页面 PageRank 分值进行排序，就得到一个页面重要程度的排名

表。根据这个排名表，将用户搜索出来的网页结果排序，排在前面的通常也正是用户期待的结果。

但是这个算法还有个问题，如果某个页面只包含指向自己的超链接，其他页面不断给它送分，而自己一分不出，随着计算执行次数越多，它的分值也就越高，这显然是不合理的。这种情况就像下图所示的，A 页面只包含指向自己的超链接。



解决方案是，设想浏览一个页面的时候，有一定概率不是点击超链接，而是在地址栏输入一个 URL 访问其他页面，表示在公式上，就是

$$PR(A) = \alpha \left( \frac{PR(B)}{2} + \frac{PR(C)}{3} + \frac{PR(D)}{1} \right) + \frac{(1-\alpha)}{4}$$

上面  $(1 - \alpha)$  就是跳转到其他任何页面的概率，通常取经验值 0.15(即  $\alpha$  为 0.85)，因为有一定概率输入的 URL 是自己的，所以加上上面公式最后一项，其中分母 4 表示所有网页的总数。

那么对于 N 个网页，任何一个页面  $P_i$  的 PageRank 计算公式如下：

$$PageRank(P_i) = \alpha \sum_{P_j \in M(P_i)} \frac{PageRank(P_j)}{L(P_j)} + \frac{1-\alpha}{N}$$

公式中， $P_j \in M(P_i)$  表示所有包含有  $P_i$  超链接的  $P_j$ ， $L(P_j)$  表示  $P_j$  页面包含的超链接数，N 表示所有的网页总和。由于 Bingoo 要对全世界的网页进行排名，所以这里的 N 是一个万亿级的数字。

计算开始的时候，将所有页面的 PageRank 值设为 1，带入上面公式计算，每个页面都得到一个新的 PageRank 值。再把这些新的 PageRank 值带入上面的公式，继续得到更新的 PageRank 值，如此迭代计算，直到所有页面的 PageRank 值几乎不再有大的变化才停止。



## 小结

PageRank 算法我们现在看起来平平无奇，但是正是这个算法造就了 Google 近 2 万亿美元的商业帝国。在 Google 之前，Yahoo 已经是互联网最大的搜索引擎公司。按照一般的商业规律，如果一个创新公司不能带来十倍的效率或者体验提升，就根本没有机会挑战现有的巨头。而 Google 刚一出现，就给 Yahoo 和旧有的搜索引擎世界带来摧枯拉朽的扫荡，用户体验的提升不止十倍，这其中的秘诀正是 PageRank。

二十几年前，我刚刚接触编程的时候，我们中国也有很多这样的编程英雄，王选、王江民、求伯君、雷军等等，他们几乎凭一己之力就创造出一个行业。正是对这些英雄们的崇拜和敬仰，引领我在编程这条路上一直走下去。软件编程是一个可以创造奇迹的地方，而不只是为了混碗饭吃。梦想不能当饭吃，但是梦想带来的可不止是一碗饭。


## 思考题

PageRank 的计算，需要在万亿级的数据上进行多次迭代计算才能完成。数据量和计算量都非常大，如何完成这样的计算？也就是说，具体编程实现是怎样的？

欢迎在评论区分享你的思考，我们共同进步。

分享给需要的人，Ta 订阅超级会员，你最高得 50 元

Ta 单独购买本课程，你将得 20 元

 生成海报并分享

 赞 3  提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 09 | 交友系统设计：哪种地理空间邻近算法更快？

下一篇 11 | 反应式编程框架设计：如何使方法调用无阻塞等待？

更多学习推荐

《架构实战营》

# 跟着阿里 P9 系统提升你的架构能力

立抢课程大额优惠



李运华  
前阿里资深技术专家 (P9)

## 精选留言 (7)

写留言



2022-03-14

再来说说思考题：

首先，大前提。海量数据计算，核心其实还是用算力去堆。分片之后堆足够的算力，也肯定是可以解决的。但是，受限于成本，肯定需要考虑如何效益最大化。

首先，还是利用算力闲置资源，因为这部分的实时性要求并不高。这部分涉及的知识应该是云原生服务编排，资源调度相关的知识点。把这部分算力，用空闲的资源提供计算服务。

具体步骤的话，首先将需要计算的数据先进行分片，使用不同的节点进行计算。其次，将数据的重要性进行排序，优先计算热点数据。比如，算力不足时，优先计算热点数据，算力充足时，再计算普通数据。

具体的本质，每次迭代都能影响PageRank评分，所以其实每一次迭代都能带来一次评分的影响。只不过越来越精确。每一次PageRank迭代过后，数据的可用性，和用户体验都会提升。我个人评估，前几次迭代，PageRank的用户体验已经比较不错了，算力就会侧重于用户体验还不太好的PageRank。

以上是我的整体思路。

作者回复: 很赞啊, 感觉你在不太了解大数据计算框架的前提下推导出大数据计算的核心思路方法。

PageRank的编程实现采用MapReduce之类的大数据计算框架, 编程方法可采用图计算或者矩阵计算。归根结底都是如何将数据分片后计算。



👍 5



**peter**

2022-03-09

请教老师几个问题:

**Q1:** URL长度字段需要两个字节吗? 有这么长的URL吗?

**Q2:** 峰值同时在线人数这个指标有意义吗?

在“期中测评: IM系统设计中”, 有一个指标是“峰值同时在线人数”, 这个指标很重要吗? 一般都是说并发数量, 峰值同时在线人数会占用系统很多资源吗?

**Q3:** 网页中哪些单词会被提取?

一个网页会有很多单词, 是全部提取吗? 还是选择一部分提取? 如果是部分提取, 那根据什么选词?

**Q4:** 百度的关键字提示是哪里处理的?

用百度搜索。输入一个词, 会给出很多提示。这个提示是浏览器实现的? 还是后端应用服务器实现的?

**Q5:** PageRank是google, 那百度用什么?

如果百度也用PageRank算法, 算侵权吗?

**Q6:** 现在搜索引擎使用AI了吗?

作者回复: 1 1个字节才256, 超过这个长度的URL应该很常见吧, 浏览器对URL长度的限制一般在2000以上

2 这个问题是不跟其他专栏串了, 我们还没有期中评测呢~~ 不急, 会有的~~~

3 所有单词

4 根据过往用户搜索进行前缀匹配, 服务器实现

5 也用pagerank

6 AI这个词太含糊了, pagerank也算是一种AI, Q4的提示也是一种AI, 搜索广告推荐也是AI, AI无处不在



👍 2



2022-03-14

这一篇专栏可以说相当硬核, 核心知识点:

1. 正, 倒排索引, 针对网页包含的关键词与关键词对应的网页的映射;
2. 多个集合查找集合的算法设计+海量数据计算的算法设计;

### 3. PageRank 模型的搭建思路与实际落地的方案；

除了数据模型这块，实在是吸收不了之外，其他的每一个点都让我受益匪浅。



1



江楠大盗

2022-04-07

数据是挨着存放的，知道docID怎么才能快速定位到数据？顺序遍历吗？

作者回复: 网页在HDFS存储是为了构建索引和计算PageRank，这个过程是顺序遍历的。

根据docID检索网页内容的场景可以用HBase解决。



aoe

2022-03-24

原来倒排索引是这样！感谢老师两张图，看了之后瞬间明白！



YY

2022-03-16

请问老师，感觉提取倒排索引和page rank是一个很慢的过程，应该是个离线系统。

但是有些时候，会有一个突发新闻比如微博大瓜，新的关键词的搜索量会突然上升，而且网页每时每刻都有更新，请问如何应对？

我们临时去爬这些内容然后构建page rank和倒排索引吗？如何短时间内提高性能

作者回复: 搜索引擎索引有全量索引和增量索引两种。新闻类网页采用增量索引，爬虫专门爬新闻类网页，爬取频率高，网页数量少，构建索引速度快。其实就是你说的临时的意思，不过这个临时是永不停歇的临时。



aroll

2022-03-14

可通过类似spark这种分布式计算框架进行计算，将海量数据分段迭代计算，再汇总结果，且可对一些计算过程进行缓存，完成快速高效的计算

作者回复: 对，可以更具体点吗？输入数据的格式是什么样的？如何分段计算？如何汇总？



