流式搜索引擎“htt2” 项目报告

目录

[集群说明 2](#_Toc534494746)

[模块功能简介 2](#_Toc534494747)

[爬虫模块 2](#_Toc534494748)

[索引计算模块 2](#_Toc534494749)

[搜索模块 3](#_Toc534494750)

[算法说明 3](#_Toc534494751)

[网页相关度计算算法（tf-idf的改良） 3](#_Toc534494752)

[索引建立过程 3](#_Toc534494753)

[搜索过程 4](#_Toc534494754)

[关联词相关度计算算法（pc算法 原创） 4](#_Toc534494755)

[需求背景 4](#_Toc534494756)

[算法思路 5](#_Toc534494757)

[其他补充说明 6](#_Toc534494758)

[其他实现细节说明 6](#_Toc534494759)

[Spark Streaming的foreach操作 6](#_Toc534494760)

[Redis的存取序列化操作 6](#_Toc534494761)

[“相关搜索”功能中获取关键词示例图片的实现 6](#_Toc534494762)

[程序结构说明 7](#_Toc534494763)

[运行说明 7](#_Toc534494764)

[UI展示 7](#_Toc534494765)

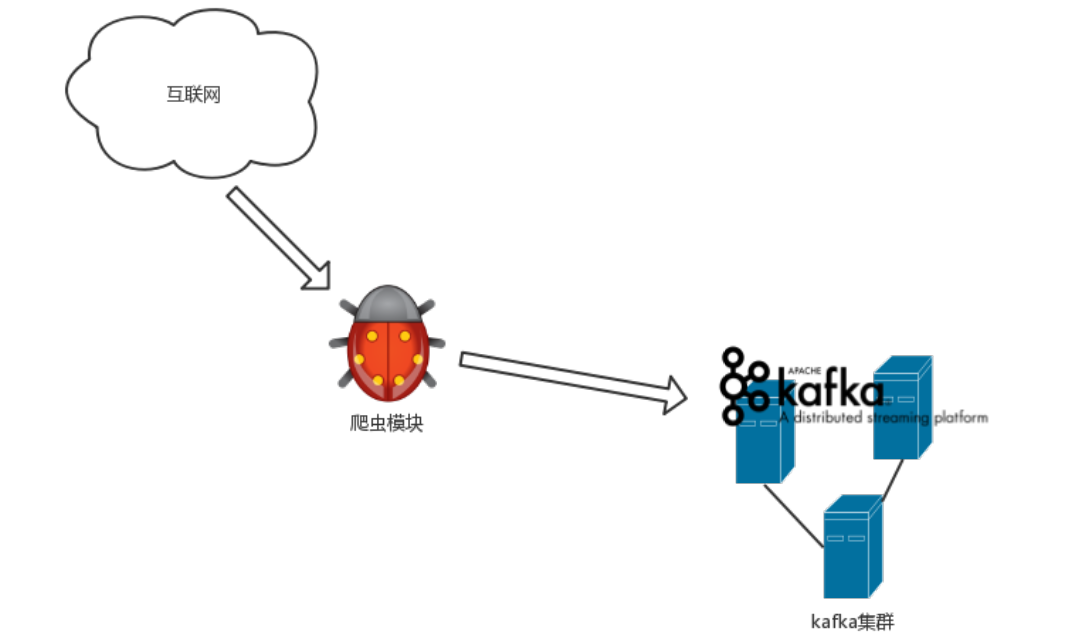
[个人感悟 7](#_Toc534494766)

# 集群说明

## 模块功能简介

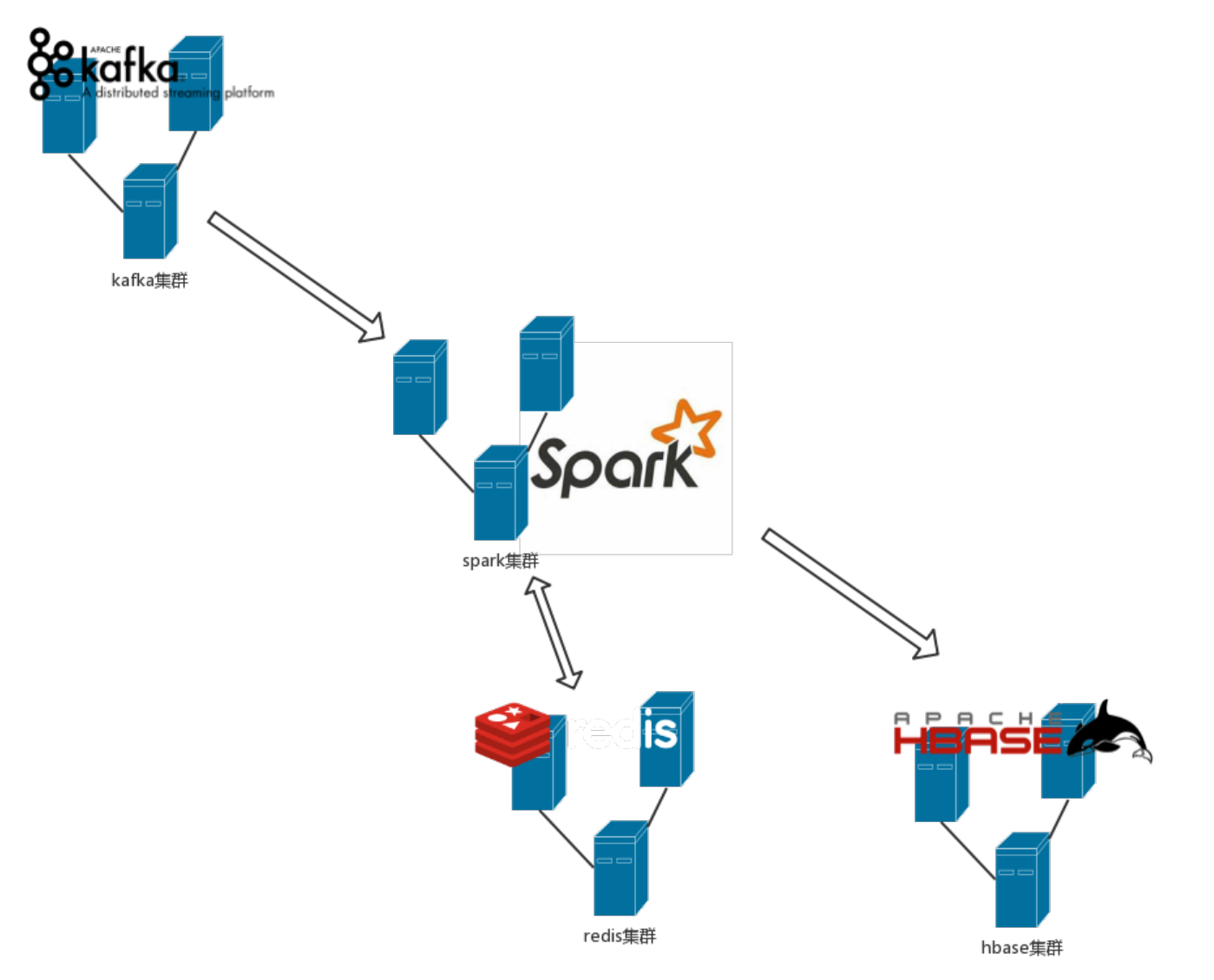
### 爬虫模块

爬取网页，作为kafka的生产者，将 (网页url,网页原html) 的键值对传向kafka。



### 索引计算模块

使用spark streaming，作为消费者从kafka中获取 (网页url,网页原html) 的键值对，计算得出与该网页相关的索引值，在系统中体现为tf数值与pc数值，将相关索引存入hbase中。而且如果更新了某词的索引，而该词的旧索引已经缓存在redis中，则更新redis中缓存的索引，维护索引的正确性。

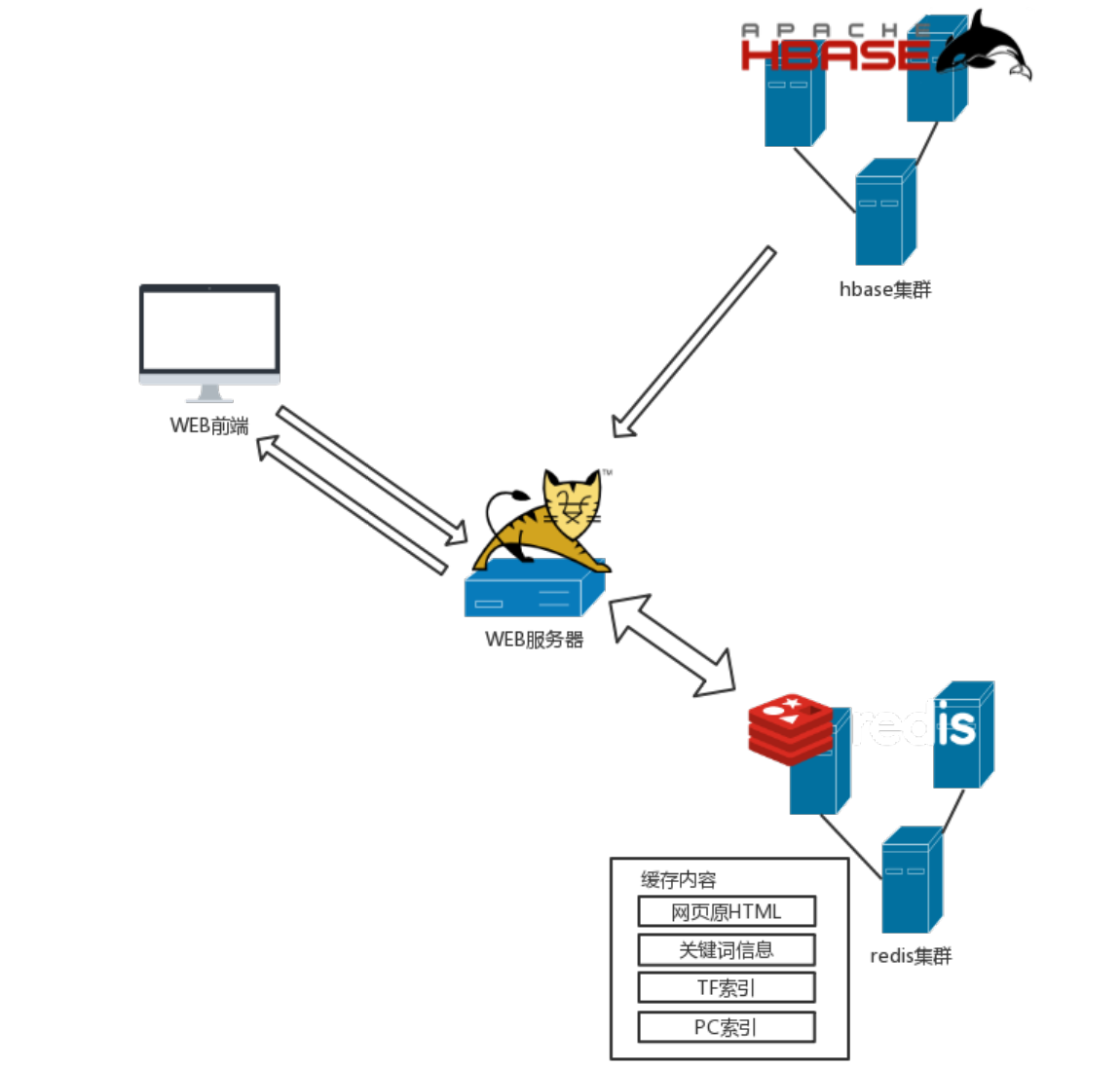


### 搜索模块

**首先访问redis，尝试得到关键词相关的索引值**，如果待搜索的关键词的相关索引不在redis中则访问hbse，得到索引值并更新redis中的缓存。

然后用得到的索引值计算出搜索结果。

最后搜索结果要用于前端展示，还需要得到一些与搜索结果相关的信息。获取信息的过程中需要使用网页原html等内容，这部分内容，也会使用redis进行缓存。



# 算法说明

## 网页相关度计算算法（tf-idf的改良）

### 索引建立过程

采用了tf-idf算法，使用spark streaming来计算网页与待搜索关键词的相关度。

文档分词后，计算每个词在其出现的文档中的词频（tf值），建立索引。建立的索引格式如下。

假如总共有两篇文档，文档地址为“./1”和“./2”，内容分别为“i am mike”和“ooh mike ooh”，建立后的4条索引记录，Json格式下应为：



Redis与hbase中各自有对该索引的储存格式。建立索引后存入hbase中，并更新缓存。

### 搜索过程

根据idf的平滑后的公式：

其中为系统中的总文档数目，为出现了词的文档数目。

而在该搜索引擎系统中，得到系统中的总文档数目需要计算或维护的过程较为复杂，效率较低，所以在执行一个语句的搜索过程时，将的语义改为——含有该语句分词后得到的一个或多个关键词的文档总数。

新语义的计算较为简便，改良了计算效率。而且通过对算法的分析，更改为新语义后对数值排序的准确程度影响不大。

搜索过程首先对待搜索的语句分词，循环取出关键词的tf索引值，计算出对应文档的tf-idf数值并累加排序，得到搜索结果。

## 关联词相关度计算算法（pc算法 原创）

### 需求背景

在搜索一个语句时，将语句分词，得到一组关键词。我们希望得到和这组关键词相关联的一些相关关键词，并根据相关度排序。

截图来自演示视频，右侧的相关搜索条目即为该部分要实现的需求。



### 算法思路

Pc算法即为pair count算法，从所有文档中计算一对单词共同出现的次数并将其作为。单词相关度的评判标准。

首先，我认为两个词语在同一篇文章中出现，说明两个词语具有一定的相关性。所以如果我们建立一个格式如下的索引，然后再根据其数值倒序排序即可。



然而如果对所有文档的内容建立pc索引，算法的时间复杂度为。其中为文档数目，为平均每篇文档的词数。不但时间复杂度较高，空间复杂度也可能会较高，而且出现许多pc数值极小，基本无效的索引值。

出于对资源的节省和对性能的要求，该系统实现的pc索引计算，是对所有文档的标题，进行pc索引的建立和持久化。文档的标题概括了文章的主题，具有很高的代表性，在大大节省了资源，提高了效率的同时，对算法的准确性不会造成过大的影响。

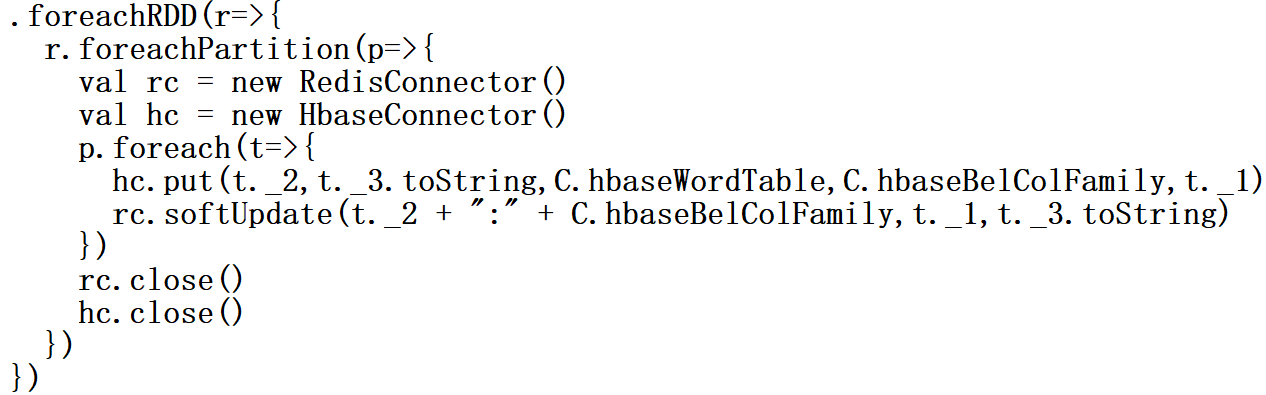
# 其他补充说明

## 其他实现细节说明

### Spark Streaming的foreach操作

在实现过程中了解到，Spark Streaming的foreachRDD运行在Driver端，而foreach和foreachPartion运行在Worker节点。

所以如果在driver上创建与hbase或者redis等的传输数据的连接对象，则很可能由于对象无法序列化而出现异常。而如果为每一条数据都新建一个传输对象，则十分浪费资源，所以系统中采用了较为合理的方法，为每一个partition创建一个对象。代码节选如下。



### Redis的存取序列化操作

由于索引的格式适合于在redis中以hash的形式储存，所以开发初期本打算使用hash格式储存索引，在需要取得索引的时候使用hgetall命令。

但后来了解到，redis的hgetall命令的时间复杂度为。其中为该hash下储存的key的数目，时间复杂度较高，由于redis是单线程的，容易造成redis服务器cpu资源消耗过大。

所以最终，索引的存取格式为Java对象在序列化之后，将字符串作为value，以redis键值对的形式在redis中存取，将解析对象的压力分担到调用redis服务的服务器中。

序列化和反序列化相关的方法，位于serializeutils包下的SerializeUtils类中。

### “相关搜索”功能中获取关键词示例图片的实现

截图来自演示视频，可以看到右侧的相关搜索面板，每一个词条，会配合一个与这个词语相关的图片。



这个图片的获取，我钻研了各种搜图网站以及维基网站的请求机制，最终实现了从关键词获取示例图片url的方法，参见WordMessageCatcher类下getWordMessageImg()方法的API文档与代码。

这个获取过程，本来被归为搜索的一部分，在同一个servlet中与其他信息获取的操作一起顺序执行，然后将完整的信息返回给前端用于展示，但此时发现程序效率极低，因为访问图片获取API需要较长时间，而在同一个请求过程中单线程地获取，速度较慢。

所以我又实现了一个servlet，用于重定向获得示例图片，在该访问该servlet的参数中指定wd参数，servlet接收参数，获取到示例图片的url，重定向到该url即可。

这样，搜索结果的返回速度不会被图片的获取拖慢，大大提升了效率。

## 程序结构说明

可参见项目API文档。

## 运行说明

可参见程序入口说明文档。

## UI展示

可参见视频演示

## 个人感悟

为了做这个项目，三天没沾床，困得不行了才在椅子上睡一会。各种细节我尽量做到我个人能力以内的最好，我真敬业。