基于HADOOP的倒排索引实现

目录

[1 应用背景介绍 2](#_Toc514101734)

[1.1 背景介绍 2](#_Toc514101735)

[1.2 意义介绍 3](#_Toc514101736)

[2 技术简介 4](#_Toc514101737)

[2.1 Hadoop 4](#_Toc514101738)

[2.2 MapReduce 4](#_Toc514101739)

[3 数据概述 4](#_Toc514101740)

[3.1 数据来源 4](#_Toc514101741)

[3.2 数据存储 6](#_Toc514101742)

[3.2.1 HDFS 6](#_Toc514101743)

[3.2.2 MongoDB 6](#_Toc514101744)

[4 系统设计 7](#_Toc514101745)

[4.1 总体架构设计 7](#_Toc514101746)

[4.2 算法思想 7](#_Toc514101747)

[4.3 集群配置 8](#_Toc514101748)

[5 系统实现 9](#_Toc514101749)

[5.1 算法概述 9](#_Toc514101750)

[5.2 结果展示 10](#_Toc514101751)

[6 项目中遇到的问题 12](#_Toc514101752)

[6.1 集群参数配置 12](#_Toc514101753)

[6.2 Mapreduce程序编写 12](#_Toc514101754)

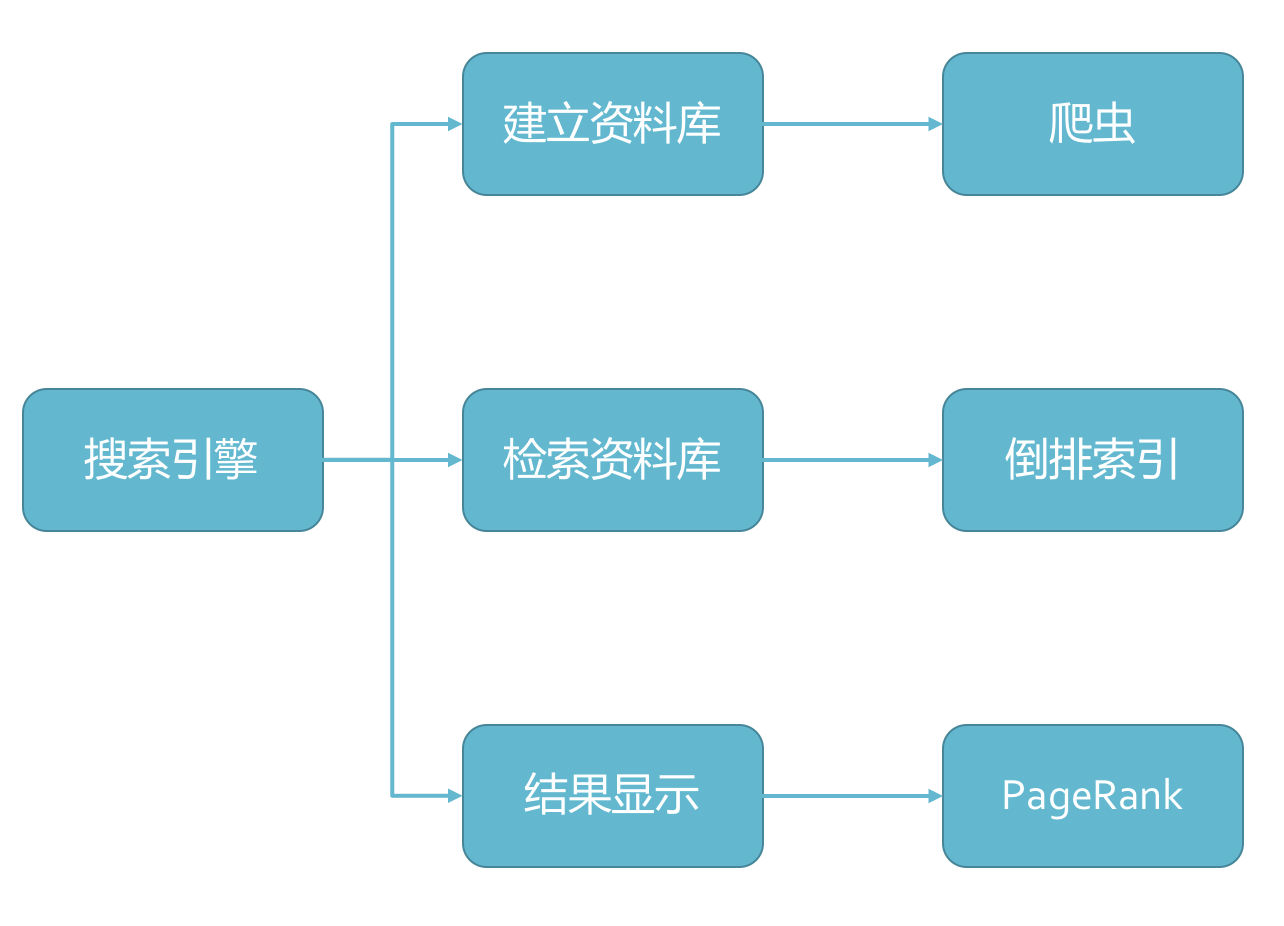
[7 算法思想拓展 12](#_Toc514101755)

[8 小组任务分工 13](#_Toc514101756)

# 应用背景介绍

## 背景介绍

在互联网搜索引擎中，主要存在着三个方面的难题：资料库的建立、资料库的检索以及检索结果的展示。狭义的搜索引擎架构也就是由这三部分组成，解决了这三个难题，也就解决了互联网在海量数据环境下的搜索问题。如下图显示，在分布式计算基本成型的今天，三大难题都已经有了主流的解决方案。其中资料库的建立主要由爬虫程序来实现，资料库的检索过程主要是倒排索引数据结构的建立，最后一步的结果显示主要由PageRank算法来实现。



在互联网级别的检索中，搜索引擎面对的是海量数据，像Google、百度这样大型的商业搜索引擎索引都是亿级甚至百亿级的网页数量 ，面对如此海量数据 ,数据库系统很难有效的管理。于是倒排索引作为一种能够快速检索包含关键字文档的技术应运而生，它主要解决了互联网级别的海量文档检索慢的问题。提到检索关键字，最简单的检索逻辑肯定是逐篇文档检索关键字集合，如果全部包含则归入结果文档中。这虽然是最简单的方法，但也属于暴力破解法，原因在于：首先我们要遍历海量文档数据，其次我们要在每篇文档中遍历所有的关键字，这是一个近似复杂度的问题。这样的搜索技术无疑会给用户带来十分漫长的等待时间，导致用户体验性极差。倒排索引是相对于正排索引的一种数据存储结构，能够在海量文档的情况下以极快的速度返回包含用户检索关键字集合的网页集合，受到了搜索引擎的青睐。

## 意义介绍

我们所熟知的正排索引是文档到单词的索引，即某篇文档中都包含了哪些词。这样的数据结构是不利于对关键字进行检索的，试想对每一片文档进行一次遍历从而判断该文档是否为检索结果文档在互联网的海量数据下是工作量多大的一件事。倒排索引是将文档到词的映射转换为词到文档的映射，即这个词都在哪些文档中出现过，这样的数据结构十分有利于关键词的检索——将所有关键字的匹配文档序列做一个交集即可得出同时包含这些关键字的文档集合。

小组选择该项目的出发点主要由两个：一是想要通过一个实际的案例来对Hadoop技术进行一次从零开始的实践，在项目过程中完成课堂理论到实践知识的飞跃，从而切实地对Hadoop相关的技术有一个更深层次的了解。另一个是考虑到在互联网级别的海量数据中，搜索引擎所使用到的技术是最基本的检索技术之一，对其他领域的软件开发从编程思想上也有一定程度的启发作用，从它的技术角度出发进行实践是一个不错的选择。

# 技术简介

## Hadoop

面对互联网级别的海量数据，分布式并行处理必然是不可或缺的。Hadoop是一个开发和运行处理大规模数据的分布式并行平台，可以使用大量计算机组成集群对海量数据进行分布式计算，使得计算效率能够成倍地提升。

## MapReduce

MapReduce是一种分布式编程模型，是一个基于集群的高性能并行计算平台，它允许使用大量廉价的计算机而不是高性能计算机组成集群来进行并行计算。它的计算方法借鉴了分而治之的思想，通过Map（映射）和Reduce（归约）两个步骤即可完成，用一句话来概括其思想就是“一个人完成不了的事就多找几个人来一起做”。

# 数据概述

## 数据来源

维基百科是一个网络百科全书项目，特点是自由内容、自由编辑。它目前是全球网络上最大且最受大众欢迎的参考工具书，名列全球十大最受欢迎的网站。虽然维基百科不算是一个搜索引擎，但是却作为最火热的参考工具被广大网友所接受。用户在使用维基百科进行词条搜索时通常不能够精确得给出想要检索的关键字，而是一些较为接近的模糊信息，所以系统需要对包含关键字的所有词条进行处理后返回最接近的那一个。维基百科的数据是对外开放的，我们可以很容易得下载到。

项目所使用的数据大小为60G，包含某个时期内维基百科的词条数据，并且以XML的格式进行存储。如下图所示，每一个词条包含在一个<page>标签块内，其中对于项目有价值的信息为<title>标签中的词条标题和<text>标签中的词条内容，项目对词条生成了内容到标题的映射。





## 数据存储

### HDFS

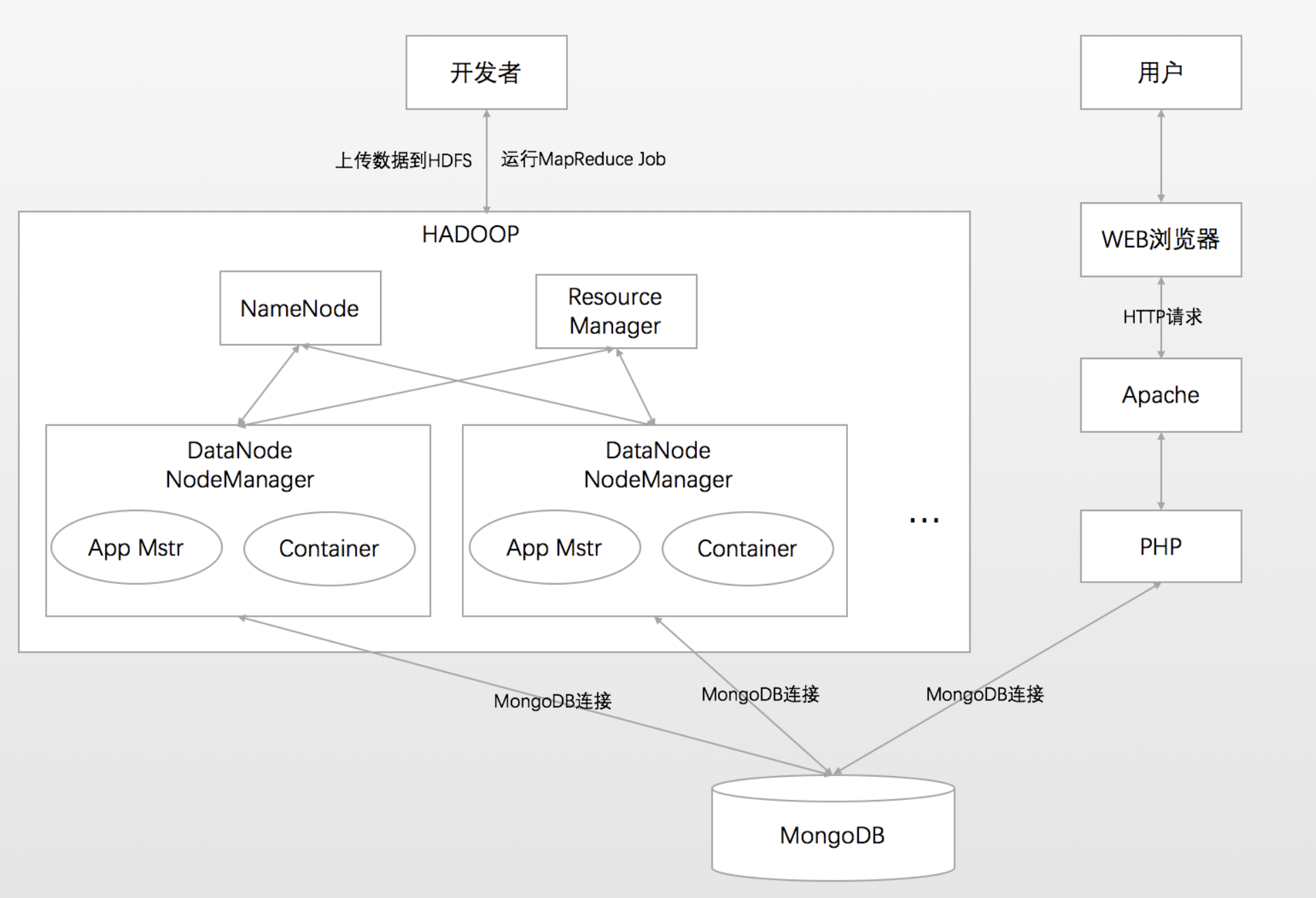
该项目将XML数据文件存储在HDFS中。Hadoop分布式文件系统(HDFS)被设计成适合运行在通用硬件上的分布式文件系统。它和现有的分布式文件系统有很多共同点。但同时，它和其他的分布式文件系统的区别也是很明显的。HDFS是一个高度容错性的系统，适合部署在廉价的机器上。HDFS能提供高吞吐量的数据访问，非常适合大规模数据集上的应用。

### MongoDB

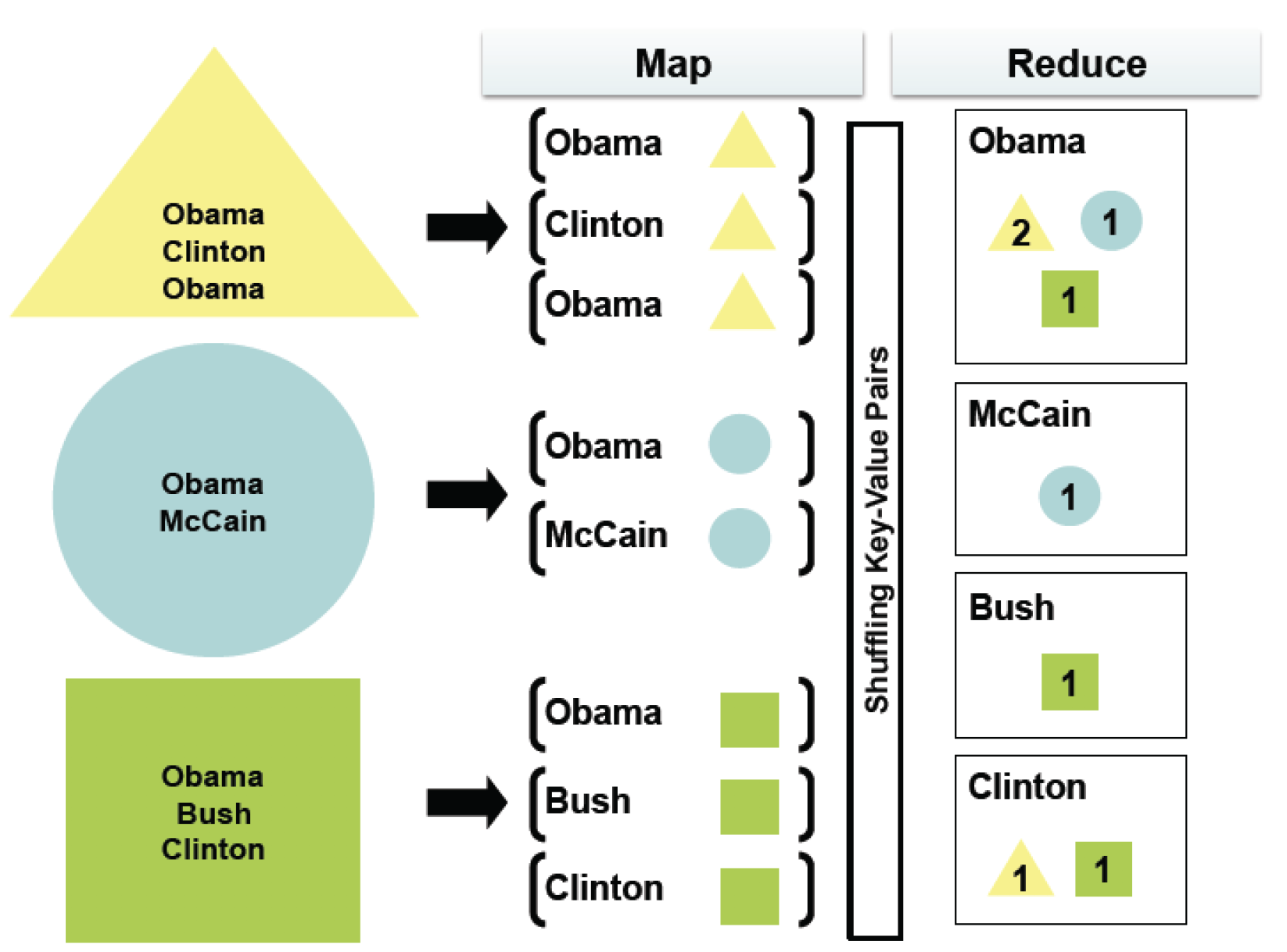
该项目将分布式计算结果存储在MongoDB数据库中。MongoDB是一个介于[关系数据库](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93)和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。他支持的数据结构非常松散，是类似[json](https://baike.baidu.com/item/json)的[bson](https://baike.baidu.com/item/bson)格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是他支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立[索引](https://baike.baidu.com/item/%E7%B4%A2%E5%BC%95)。

# 系统设计

## 总体架构设计



## 算法思想



倒排索引的算法思想可以用上面这张图来表示：首先我们有三个存储了单词内容的文本，在map阶段，算法首先将每个文本中的单词切分出来，形成一个key-value结构，其中key为(word+text)，例如(Obama+text3)，表示单词所在的文本；value值为1，表示该词在该文本中出现了1次。在combine阶段，算法将所有key相同的k-v对汇总到一起并将value进行累加，表示该词在该文本中总共出现的次数，此外在此阶段将k-v对的结构进行重组，key为word表示词，而value为(text+count)表示该词在该文本中出现的次数。在reduce阶段，算法将key相同的k-v对汇总到一起即计算出了一个词在哪些文本中出现了多少次，即可完成该计算步骤。

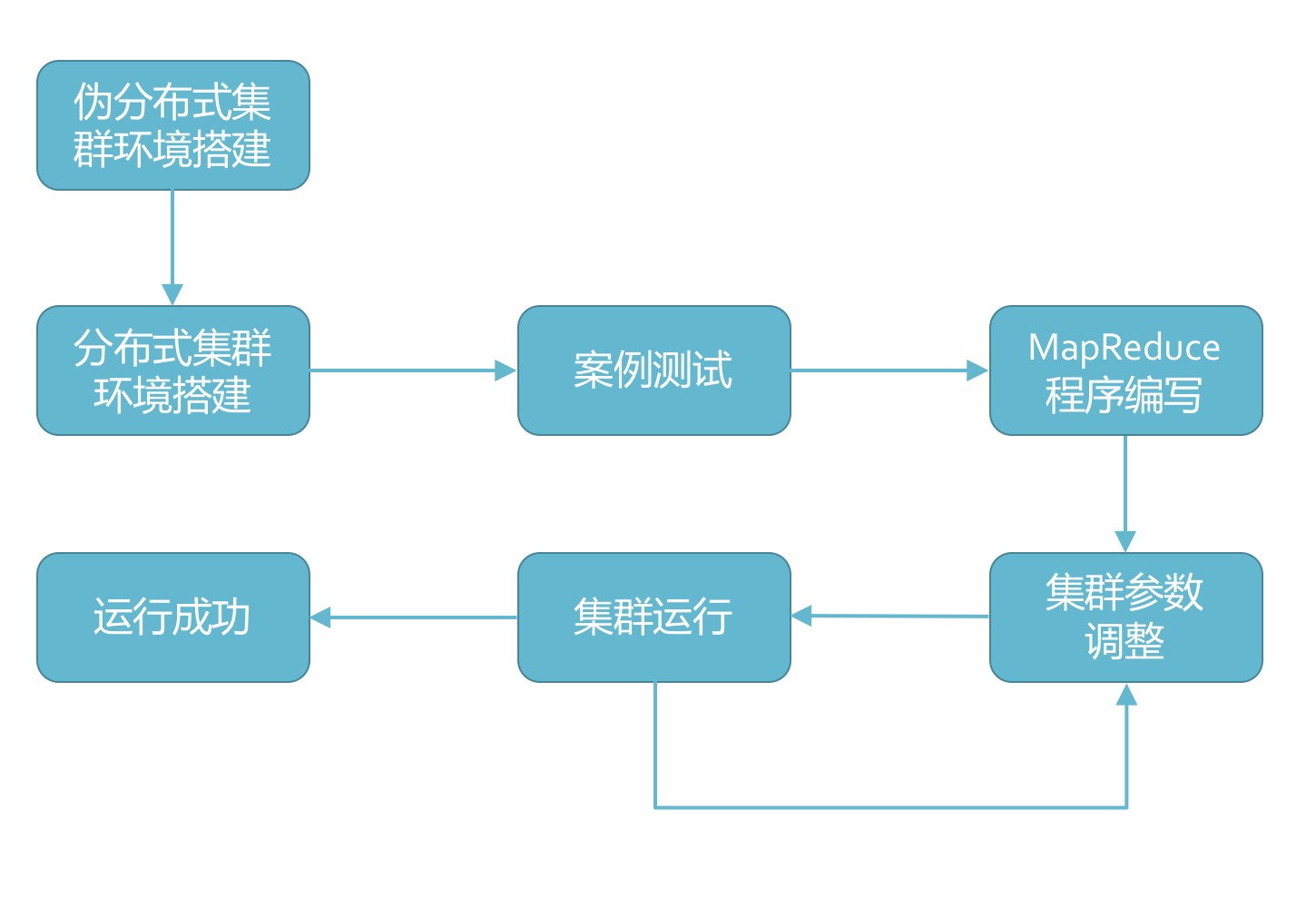
## 集群配置

|  |  |  |  |
| --- | --- | --- | --- |
| **主机** | **CPU及内存** | **硬盘** | **功能** |
| NameNode | 1核1G | 20G+80G | NameNode结点 |
| Rscmng | 1核1G | 20G+80G | ResourceManager结点  以及MapReduce JobHistory Server结点 |
| Datanode1 | 1核4G | 20G+80G | 作为DataNode结点，NodeManager结点，用于hdfs的存储以及mapreduce任务的执行 |
| Datanode2 | 1核4G | 20G+80G | 同上 |
| Datanode3 | 1核4G | 20G+80G | 同上 |
| Datanode4 | 1核4G | 20G+80G | 同上 |
| MongoDB | 1核1G | 20G | 作为MongoDB结点存储数据 |

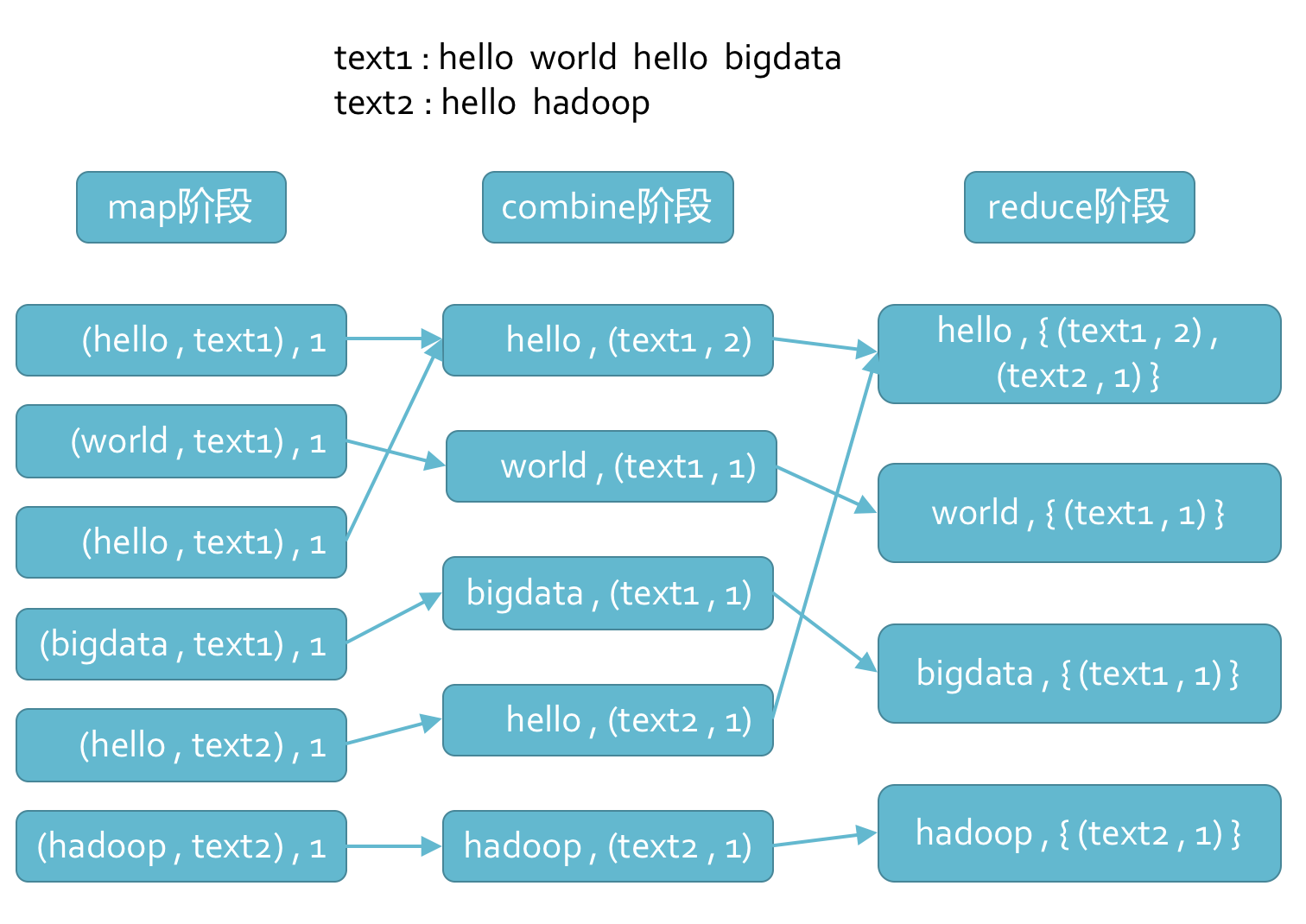
# 系统实现

## 算法概述

项目的主要实现经历了如下图所示的过程。

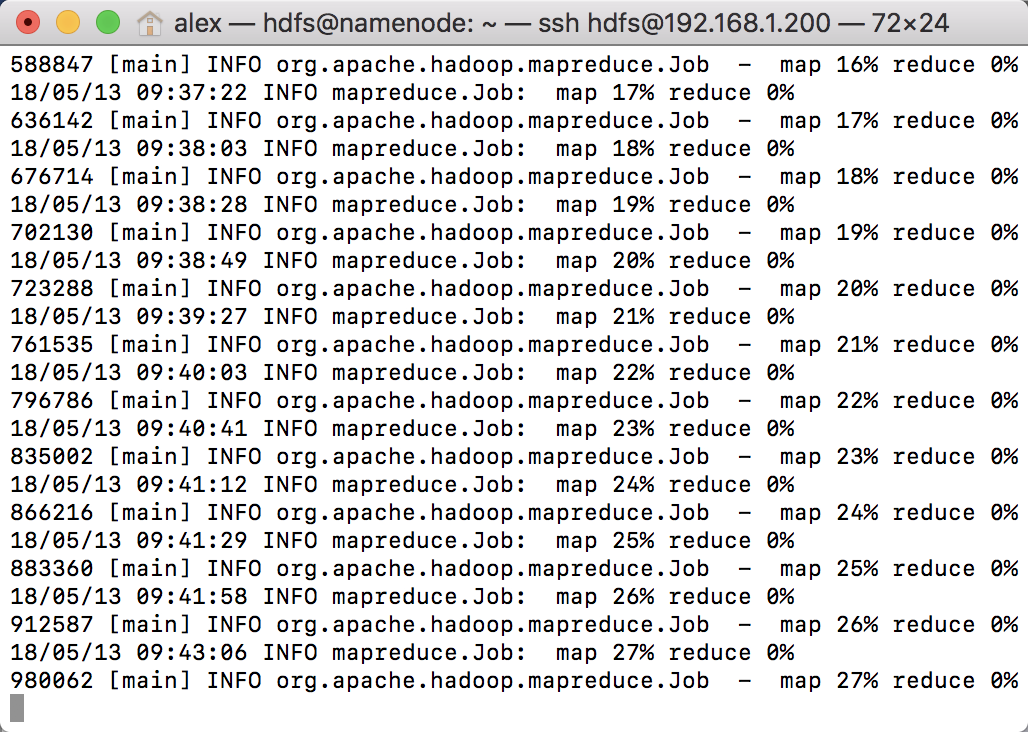


下图展示了一个简单数据的算法流程。对于数据文件text1和text2，在map阶段首先将单词和文本作为一个key并将value设置为1。在combine阶段将输入数据按照相同的key值进行汇总，累加出现次数后将key分解为单词，而将文本和出现次数拼接为复合值作为value。在reduce阶段将输入数据按照相同的key值进行汇总便算出了每个单词在每个文本中出现的次数。此处还可以有优化的方法，单词出现次数只是一个不那么重要的信息，而如果将这个数据进行变换得到一些能够对结果集进行排序的权重信息就能使该算法得到进一步的提升。

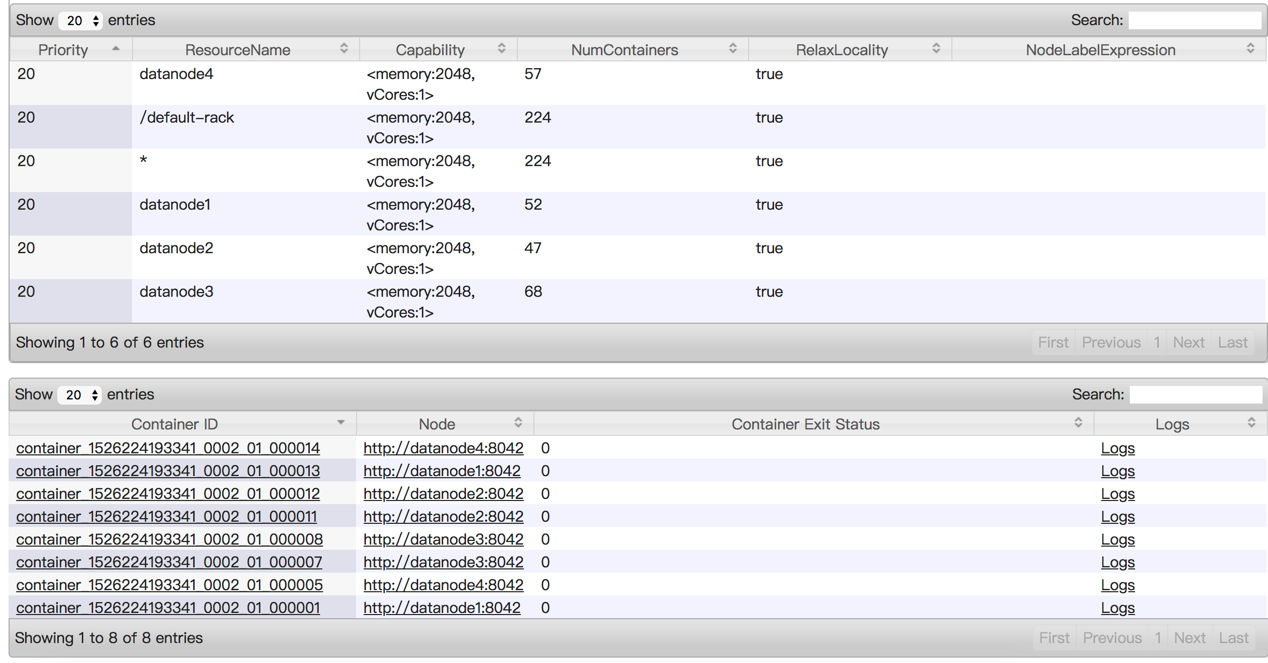


## 结果展示

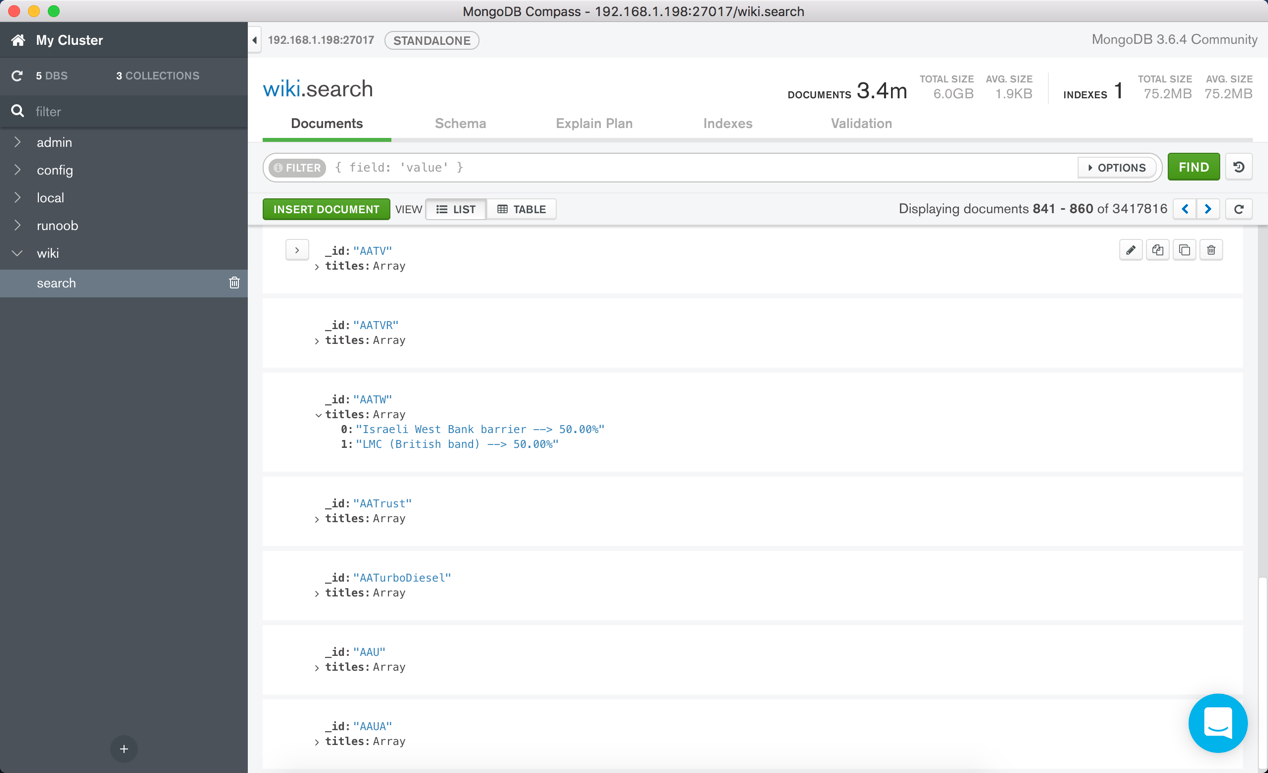
下图是程序运行过程中的进度展示图。



下图是运行过程中的container展示图



下图是运行结果存储在MongoDB中的展示，其中每条数据中的id表示单词，titles表示该单词出现的文章集合。



# 项目中遇到的问题

## 集群参数配置

在hadoop的配置文件中有四个需要手动进行各种参数的配置，其中core-site.xml文件主要是hadoop的全局配置，hdfs-site.xml主要是hdfs的局部配置，yarn-site.xml主要是yarn的局部配置，mapred-site.xml主要是mapreduce的局部配置。在这四个配置文件中，最容易出现问题的是yarn和mapreduce有关内存方面的配置，如map进程允许拥有的最小/最大内存、yarn资源管理器允许拥有的最小/最大内存等等。项目在进行过程中因为这些参数的配置出现了很多问题，如内存溢出、内存配置不一致等。解决这些问题的唯一办法就是弄明白和内存有关的任何配置参数的意义，并从实际物理机器的角度去进行配置。

## Mapreduce程序编写

网上能搜索到的倒排索引的示例程序都是对单行文本做切片并送到map阶段进行处理，所以示例程序中都默认使用了按行分割文本并分词的处理方法进行计算。而针对具体问题时，xml文件中的内容并不是按行分割就能进行处理的。需要自己编写方法匹配到相应的xml标签并且提取标签中的内容，这就需要我们适当对MapReduce程序进行研读，了解其具体的数据处理方式后对相应的方法进行重写以达到预期的处理效果。

# 算法思想拓展

倒排索引是互联网搜索引擎的基础技术之一，但除了全文搜索之外在其他领域它也可以带给我们指导思想，如在数据挖掘中很多算法的优化步骤中几乎都有一项是优化搜索，倒排索引的思想是完全可以用来做算法优化的；在购物网站的商品检索中也可以使用到倒排索引算法；另外它是一个索引算法，在索引方面有着很重要的意义，并且算法自身的优化也有着很重要的意义。

# 小组任务分工

在项目开始之前我们对整个项目进行了任务划分，该项目从整体上可以分为两个部分：搭建分布式环境、编写MapReduce程序以及程序在分布式环境上的运行。在项目的进展过程中，宋昌雄、宋文浩和应正大首先负责分布式环境的搭建，我们每个人都在自己的机器上利用虚拟机进行伪分布式的环境搭建，然后在完成该部分工作后将3台机器的虚拟机构成了一个分布式集群进行测试，保证分布式环境正常运行。其次由贾佩和王会负责MapReduce程序的编写，编写完成之后首先在本地伪分布式上进行测试运行保证程序的正确性。最后我们将两部分合在一起后大家一起做最后一部分的运行测试，并在过程中解决出现的各种问题。每次开组会时我们会互相介绍自己的工作任务，以保证其他没有做该工作的人也能够对此有所了解。