復旦大學

本科生课程项目报告

课程名称：分布式系统

姓　　名：刘卓珉　　学　　号：15307130223

学　　院：计算机学院　　专　　业：计算机科学与技术

分布式系统课程项目报告

——Wikipedia-Index

一．简介

Wikipedia-Index 是对 Wikipedia（12Gb）文件进行倒排索引的项目，基于 Hadoop File System 和 MapReduce 框架，实现了可视化界面，统计了每篇文章所有单词的 TF-IDF 值，支持对每篇文章 TF-IDF 值最大的三个单词进行搜索，并且查看指定文章的内容。

项目开源代码地址：https://www.github.com/lzmhhh123/Wikipedia-Index

Web 端校内网络访问地址：http://10.131.235.86:8080

二．项目依赖

本项目依赖于一些开源项目，具体依赖如下：

- *Hadoop File System (http://hadoop.apache.org/)*
- *Wikipedia (https://dumps.wikimedia.org/enwikisource/latest/)*
- *Node.js (https://nodejs.org/)*
- *yarn (https://yarnpkg.com/)*

三．技术栈

- *Java MapReduce Framework*
- *后端——node.js*
- *前端——Javascript React Framework*

四．实现过程

1. 本项目全程在 Windows 10 环境下开发，使用 Windows PowerShell 作为命令行，Java 部分使用 Eclipse IDE，Web 端部分使用 Atom 编辑器，首先基于网络教程，在本地搭建了一个虚拟 Hadoop File System 以方便调试。
2. 从 Wikipedia 中抽取小部分 Xml 数据（大概几十 Kb 左右），以方便调试。
3. 进入 Eclipse 进行倒排索引的计算代码编写，实现一个 XmlInputFormat 类继承于 TextInputFormat 类，主要作用是把 MapReduce 原本基于行 map 的行为改变为基于<page></page>的标签进行 map。
4. 统计每文每词的 TF-IDF 值，并把它们导出到文件中，此过程比较复杂，在项目用法中会详细说明。
5. 统计每篇文章的内容出现在 Wikipedia 文件多少个 bytes 处，也统计每篇文章的长度是多少个 bytes，用(ID, Offset, Length)的形式记录下来，输出文件（约 50Mb），方便 Web 后端处理。
6. 提取每篇文章 TF-IDF 最大的三个单词，用作每篇文章的 Web 端搜索关键词，同样以文件的格式记录，文件大小约 90Mb。
7. 实现 Web 后端与前端，启动后端的过程大概需要 15s 左右（用于处理前面所叙述的两个几十 Mb 的文件），因为处理了每篇文章在 Wikipedia 的 Offset 和 Length，所以从 Web 端访问文章内容，用户方面几乎感受不到延迟。值得说明一下，这个架构我思考了两种办法，一种是我想把每篇文章都导出一个单独的文件，以${Id}.txt 的格式命名，但是这种方法 Web 端的静态资源服务器启动速度将会非常慢，因为静态资源会有接近 10Gb，所以我采用了第二种方法，这样大大降低了后端的启动速度，也不影响用户的访问速度。

五．文件架构

```
WikipediaIndex
├── bin                         //Java class
│   ├── DF.class
│   ├── ...
│   └── XmlInputFormat.class
├── src                         //MapReduce source file
│   ├── DF.java                 //count document frequency
│   ├── ExtractPage.java        //extract each page to a independent file
│   ├── IdOffset.java           //count each page's size & offset in Wikipedia
│   ├── IntArrayWritable.java   //A class extends ArrayWritable
│   ├── MaxThreeLabel.java      //count the biggest three TF-IDF words for each page
│   ├── PageWordCount.java      //count the number of words for each PageWordCount
│   ├── TextArrayWritable.java  //A class extends ArrayWritable
│   ├── TF_IDF.java             //count term frequency-inverse document frequency
│   ├── TF.java                 //count term frequency
│   └── XmlInputFormat.java     //A class extends TextInputFormat
├── view
│   ├── bin
│   │   └── www                 //start node.js server
│   ├── public                  //frontend static file
│   │   ├── favicon.ico
│   │   ├── ...
│   │   └── manifest.json
│   ├── src                     //frontend source file
│   │   ├── App.css
│   │   ├── App.js
│   │   ├── index.css
│   │   ├── ...
│   │   └── logo.svg
│   ├── app.js                  //main node.js file
│   ├── package.json
│   └── yarn.lock
├── LICENSE
├── preview.jpg
├── preview2.jpg
└── README.md
```

六．项目部署用法

1. `git clone https://github.com/lzmhhh123/Wikipedia-Index & cd Wikipedia-Index`
2. 用 `Eclipse` 导出`.jar`包
3. `hadoop fs -put ${WikipediaFilePath} ${YourHadoopFileSystemPath}` 将 Wikipedia 放到 Hadoop File System 中

4. `hadoop jar LzmWikiIndex.jar LzmWikiIndex.TF ${WikipediaPathOnHadoop} ${TFOutputPath}` 计算每个单词的 TF 值
5. `hadoop jar LzmWikiIndex.jar LzmWikiIndex.DF ${TFOutputPath}/part-r-00000 ${DFOutputPath}` 计算每个单词的 DF 值
6. `hadoop jar LzmWikiIndex.jar LzmWikiIndex.TF_IDF ${DFOutputPath}/part-r-00000 ${TFOutputPath}/part-r-00000 ${TF-IDFOutputPath}` 计算每个单词的 TF-IDF 值
7. `hadoop jar LzmWikiIndex.jar LzmWikiIndex.MaxThreeLabel ${TF-IDFOutputPath}/part-r-00000 ${MaxThreeWordOutputPath}` 统计每篇文章 TF-IDF 最大的三个词
8. `cd view & yarn build` 打包前端
9. `echo ${WikipediaFilePath} > WikipediaPath`
10. `hadoop fs -get ${MaxThreeWordOutputPath}/part-r-00000 ${ThisRepoPath}/view/`
11. `java -classpath LzmWikiIndex.jar LzmWikiIndex.IdOffset ${WikipediaFilePath} ${ThisRepoPath}/view/IdOffset`
12. `node bin/www` 启动后端，在本地 http://localhost:8080 访问

七. 项目预览