轻量型搜索引擎项目文档说明

目 录

1 项目说明

使用 C++语言，实现轻量型搜索引擎的后台服务器；

2 开发环境

Linxu : ubuntun 15.04
g++: version 4.9.2

3 系统目录

3.1 文件夹

Src - 存放系统的源文件（.cc）
Inc - 存放系统的头文件（.h）
Bin - 存放系统的可执行文件
Conf - 存放系统所需配置信息
Data - 存放系统所需数据
    Data/cache.lib - 查询模块缓存文件
    Data/inverted_index.lib - 查询模块索引文件
    Data/ripepage.lib - 网页库
    Data/offset.lib - 记录网页在网页库中偏移的文件
Lib - 存放系统所需开源文件等
Raw - 存放未经处理的原始网页文件
Client - 存放客户端文件

3.2 文件

server.sh - 服务器程序执行脚本
client.sh - 客户端程序执行脚本
Makefile - 服务器程序编译文件

4 系统运行流程

4.1 总体流程图

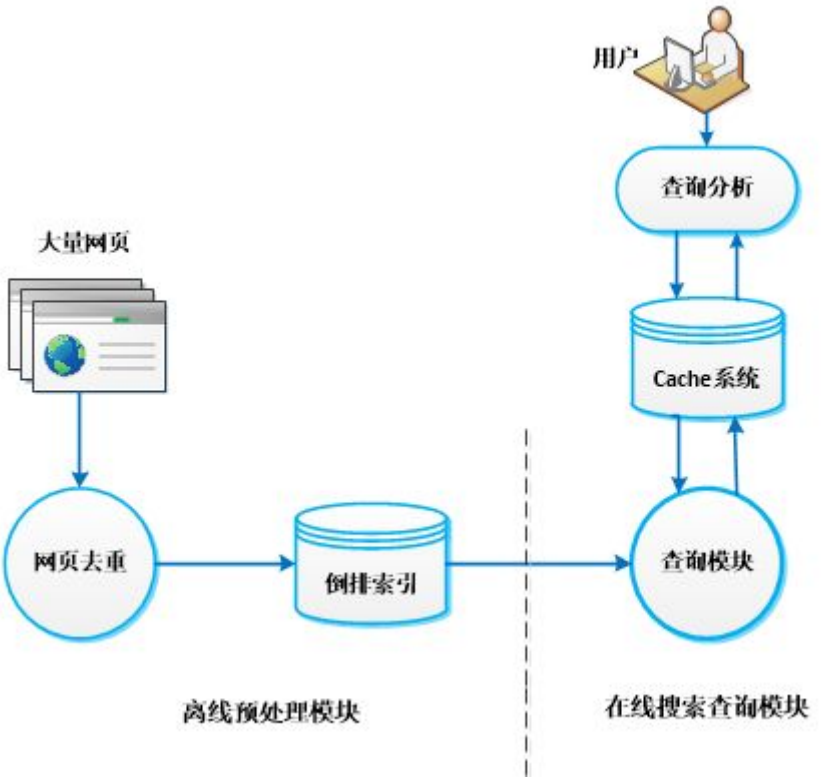

图 1

4.2 子系统流程图

4.2.1 离线预处理模块

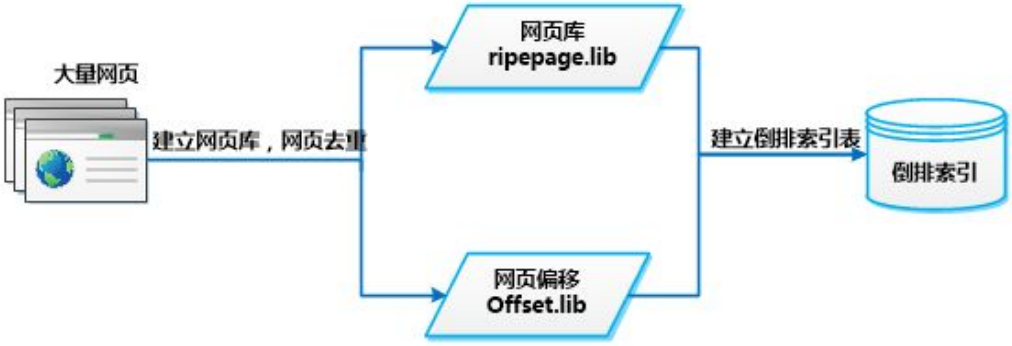

图 2

子任务：提取网页库，网页去重，建立倒排索引；

4.2.2 在线搜索查询模块

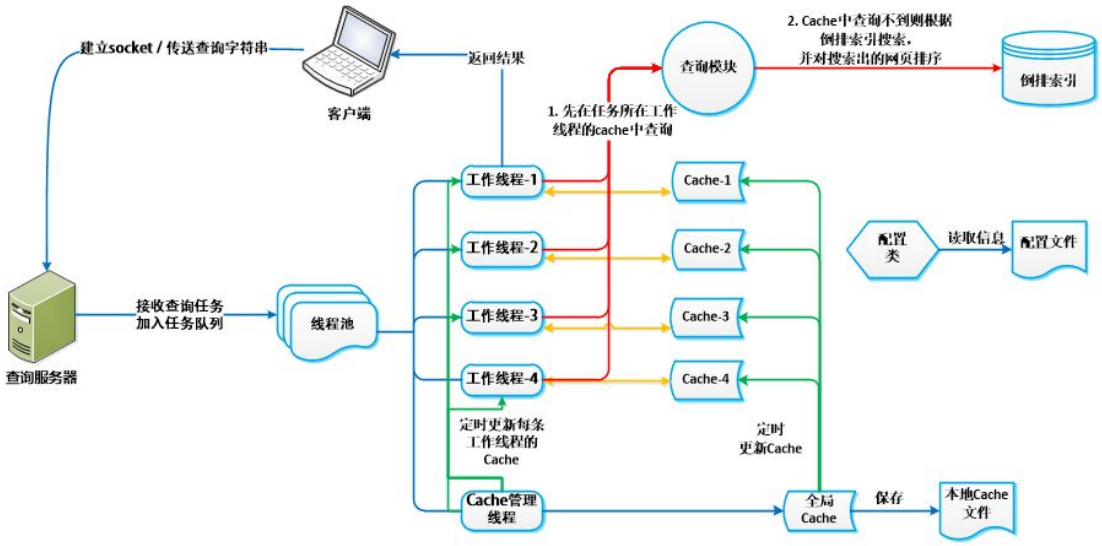

图 3

子任务：
服务器-客户端框架（建立 Socket 连接）；
线程池构造；
查询模块（Cache 中搜索/倒排索引中搜索+网页排序）；
将结果包装成 Json 字符串并返回客户端；

5 功能说明

5.1 主线程

5.1.1 离线预处理部分

选择是否需要更新网页库和索引文件：需要则更新文件；不需要则进入“在线部分”；

5.1.2 在线部分

（1）获取配置文件/索引文件/网页库/网页偏移文件/缓存文件中的信息；初始化线程池/定时器/服务器；
（2）依次启动线程池/定时器/服务器；
（3）监听并接收客户端发来的查询字符串；
（4）当有请求从客户端传来时，将该请求封装成任务类并放入线程池中处理；
（5）任务处理完毕则将结果返回给客户端；

5.1.3 主线程流程图

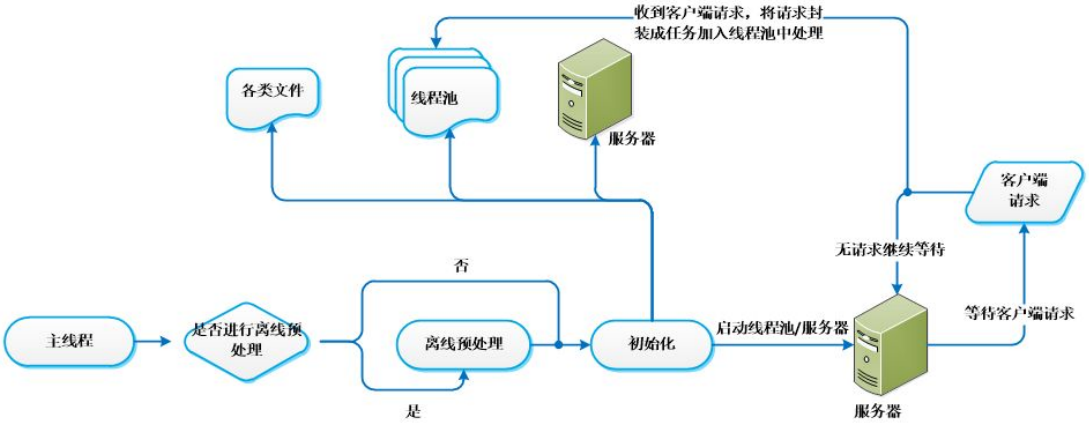

图 4

5.2 工作线程

5.2.1 功能

（1）初始化线程内的 cache 对象；

（2）循环从线程池中的任务队列中取任务；

（3）处理任务：若缓存中有结果则直接返回给客户端；否则进行搜索操作，再将结果返回给客户端并保存在本线程内的 cache 对象中；

5.2.2 工作线程流程图

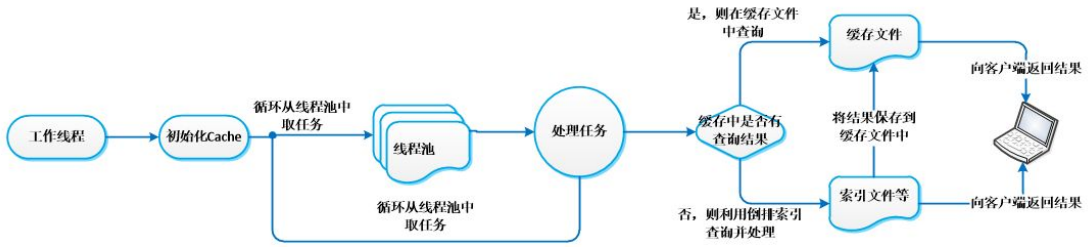

图 5

5.3 Cache 管理线程

5.3.1 功能

（1）服务器启动时读取本地 cache 文件到内存中；

（2）线程池启动时传入全局 cache 对象；每个线程启动时拥有自己的 cache 对象；

（3）定时更新线程池中的全局 cache 对象和本地 cache 文件；再利用全局 cache 对象更新每个线程中的 cache 对象；

5.3.2 Cache 管理线程流程

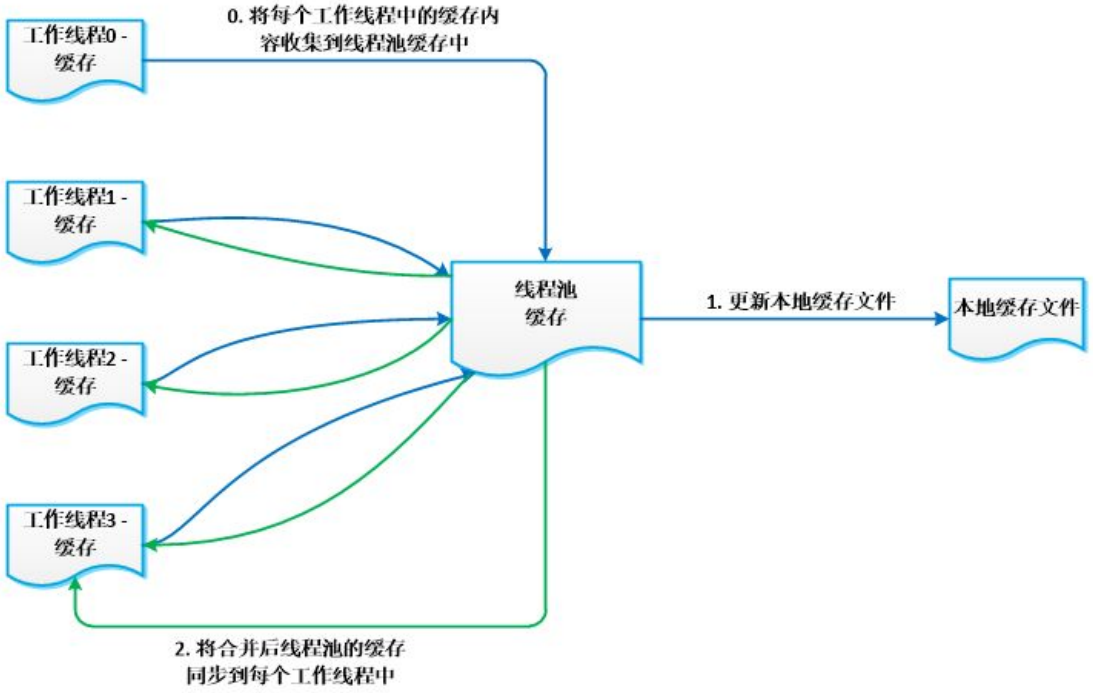

图 6

6 算法和数据结构

6.1 网页去重

在使用爬虫时会提取到重复内容的网页，所以在预处理建立网页库阶段需要进行网页去重；

（1）数据结构

需要对网页抽象出一个 class MyPage,其中包含 docid/url/title/content 等数据成员；

将所有的网页存入 vector<Mypage>中；

（2）算法

思路：确定两篇网页是否相同，需要对网页提取特征，该特征能够代表该网页

两种方法：

a. topK 方法（本项目中使用）

对每篇网页进行词频统计，取网页的前 n 个词频最高的词 来代表该网页；

流程：

分词 => 去停用词（停用词：“的，得，地，如果，但是，还”等出现频率很高，但没有区分度的词语）=>取两篇文章的 top n 词汇进行比较，根据策略判断两篇文章是否相等；

b. 寻找公共子序列 LCS（精确度更高）

取文章中每个‘，’的前 n 个字节和后 m 个字节分别拼成字符串，将这些字符串拼接成长字符串，作为该网页的特征；

=》比较两个网页的长字符串，获取公共子序列，将该公共子序列与较短的网页字符串进行比较，根据相同的比例判断两篇文章是否相等；

删除重复网页：
当从 vector 中删除网页时，vector 底层是线性数组，在 erase 时，会将后面数据整体向前移动，使得效率下降；
=>删除时默认删除 docid 较大的文档，将该文档和尾部的文档交换位置，再将该文档 pop_back 出去即可，这样就不涉及移动数据的操作，效率较高；

6.2 建立倒排索引

（1）数据结构

数据结构	时间复杂度	本项目中是否使用
std::map	O(log(n))	否
std::unordered_map	O(1)	是

（2）算法
倒排索引即通过词汇检索文章即

```
hash_map<string, set<pair<int, double> > >;
```

```
word -> docid1 freq1 weight1, docid2 freq2 weight2, ...
```

图 7

weight 权重的计算（0-1 之间）：
A.
tf - term frequency，word 在文章中出现的次数；
df - document frequency，word 在所有文章中出现的次数即包含该词的文档书；
idf - inverse document frequency（逆文档频率），表示该词对于该篇文章的重要性；

$$\mathrm{idf} = \log(N / df)$$

B.词的特征权重（即该词对于该篇文章的重要性）的计算：

$$\mathrm{w} = tf * idf$$

C. 最后需要进行归一化处理得到最终的权重值：

$$\mathrm{w}eight = w / sqrt(w1^2 + w2^2 + ... + wn^2)$$

6.3 网页查询和排序

在网页查询模块首先包含所有关键字的网页，再对查询到的网页进行排序返回客户端；

（1）算法
A. 将客户端传给服务器的查询词当作一篇网页，并提取这个网页的特征来代表此网页；

eg：

中关村 在线 论坛 => a = (0.2, 0.6, 0.3)

向量中每个坐标值代表对应词汇在该网页中的 weight 权值；

B. 通过倒排索引表查找包含所有关键字的网页；

eg：

```
    ->                                          i
中关村  set_iterator[0] ->  (1  2  3  5  6  7  8)
    ->                                      j
在线   set_iterator[1] ->  (1  2  5  6  8  9)
    ->                                  k
论坛   set_iterator[2] ->  (1  3  6  8  10  11)

1  6   8 (3个关键词均包含的网页id)

set_iterator[3];
```

图 8

C. 利用“余弦相似度”对查找到的网页进行排序

eg：

a = (0.2, 0.6, 0.3)

	所在文章id	中关村	在线	论坛
b	1	0.1	0.3	0.8
c	6	0.2	0.4	0.6
d	8	0.3	0.1	0.7

图 9

分别计算上图中向量 b,c,d 与向量 a 之间的余弦值，按照夹角的大小对网页进行排序；

7 操作说明

7.1 配置文件

Raw:

参考语料库地址："http://pan.baidu.com/s/1i3XwMBN"

直接将上述文件解压到 Raw 文件夹下即可

Conf/server.conf:

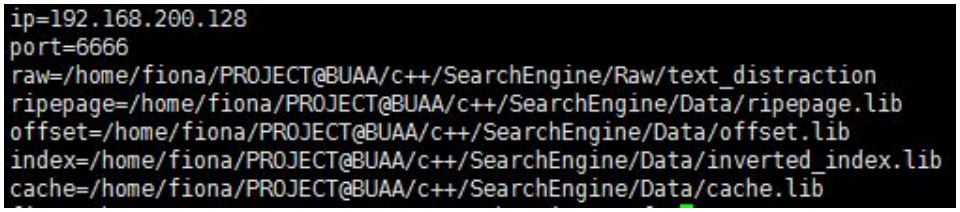

图 10

ip - 服务器的 ip 地址；
port - 服务器监听客户端连接的端口号；
raw - 存放原始所有网页文件的文件夹；
ripepage - 经过处理后的网页库文件夹地址；
offset - 记录网页库中网页偏移位置文件的地址；
index - 倒排索引文件夹地址；
cache - 记录查找结果的缓存文件地址；

server.sh:

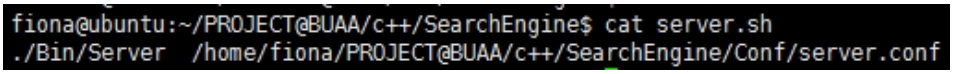

图 11

./Bin/Server 的参数为配置文件的地址；

client.sh:

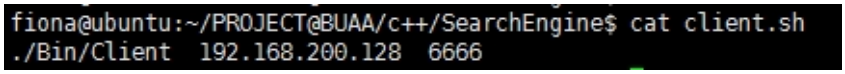

图 12

./Bin/Client 的参数为服务器的 ip 地址 和 服务器的端口号；

7.2 运行效果

执行 make 命令生成可执行程序./Bin/Server；
执行./server.sh 启动服务器：

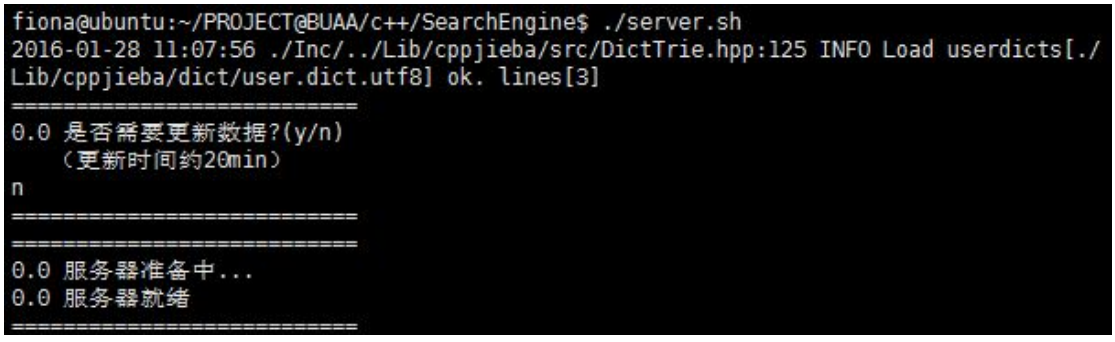

图 13

上图中“是否需要更新数据?” 输入 n 直接进入在线部分，输入 y 则进入离线部分重新建立网页库和索引文件。

执行./client.sh 启动客户端：输入词汇开始查询

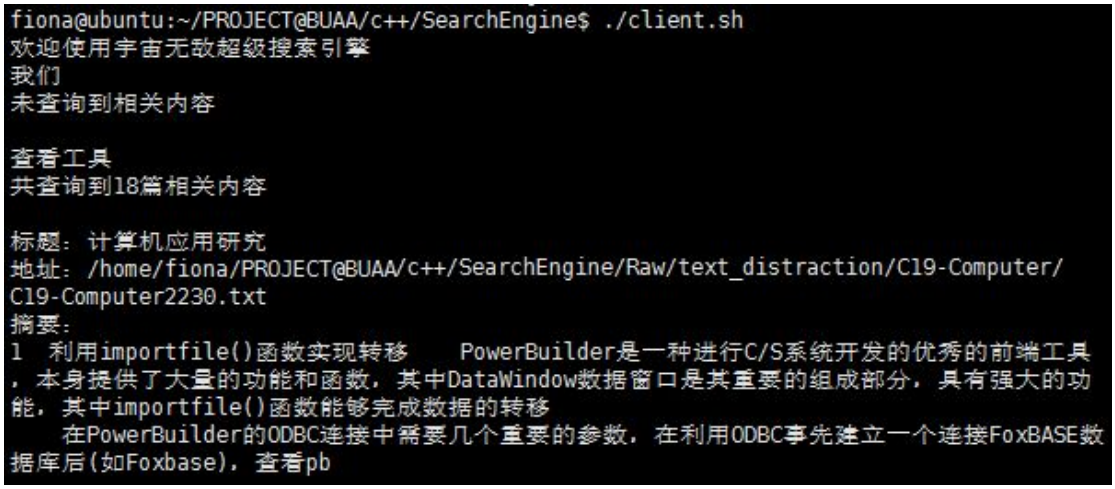
fiona@ubuntu:~/PROJECT@BUAA/c++/SearchEngine$ ./client.sh
欢迎使用宇宙无敌超级搜索引擎
我们
未查询到相关内容
查看工具
共查询到18篇相关内容
标题：计算机应用研究
地址：/home/fiona/PROJECT@BUAA/c++/SearchEngine/Raw/text_distraction/C19-Computer/C19-Computer2230.txt
摘要：
1 利用importfile()函数实现转移 PowerBuilder是一种进行C/S系统开发的优秀的前端工具，本身提供了大量的功能和函数，其中DataWindow数据窗口是其重要的组成部分，具有强大的功能，其中importfile()函数能够完成数据的转移
在PowerBuilder的ODBC连接中需要几个重要的参数，在利用ODBC事先建立一个连接FoxBASE数据库后(如Foxbase)，查看pb

图 14