作业1：分布式 DBLP 数据查询系统报告

1. 小组成员

2052710 吴笛

2052717 陈晨

2. 项目结构

项目分为client/datanode/common/server四个部分

client

客户端，包括Logger/Main/RequestClient四个类

Main用于启动客户端，Logger维护客户端的日志，RequestClient用于向若干个servers发出查询请求。

datanode

用于数据处理，包括数据集的切分、预处理和负载均衡的实现。

common

用于存储一些公有数据，如服务器ip，访问的端口和数据文件路径等。

server

服务端，包括DataServer/Query/QueryServer/UploadServerThread四个类

DataServer和UploadServerThread用于上传数据，实现服务端的负载均衡

QueryServer接收来自客户端的请求，Query实现具体的查询功能

3. 算法设计与实现

3.1 数据预处理

首先对原xml文件进行分块。由于dblp.xml中，每篇著作的起始标签不同（有article/bool/www等），因此以key来标识一篇著作的起始位置，按行读取并分块。

在实验过程中，我们分别使用了grep/wc和java的正则匹配来直接扫描xml文件，发现查询时间非常长，对所有数据查询一次至少需要5-10分钟。但由于请求的查询是作者（在指定年份）的论文发表数量，因此只需要关注原数据中的author/year信息。所以先对每份xml进行扫描和简化，记录每位作者在每一年的作品数，用json存储简化后的结果，从而避免对xml数据块进行全局扫描。

例如：{"Paul Kocher": {"2018":1, "2020":2}}

3.2 服务端

对处理好的json文件，我们使用了4台云服务器作为存储端，每台云服务器存储一份原件和一份副本，实现了负载均衡和容错功能。

QueryServer开启后则始终等待客户端的请求。查询时，Query检测出文件目录有n个json文件，则会使用n个线程来扫描和查询，分别向客户端返回每个线程的查询结果。消息格式为：[块号]-[副本号]-[查询结果]，例如0-1-2就代表第0块文件的副本1中，查询到的论文篇数为2。

3.3 客户端

服务端和客户端使用java的Socket类建立通信，并用DataOutputStream和DataInputStream等类来实现数据传输。

用户通过console输入想要查询的作者姓名（和年份范围）。客户端在本地存有所有m个存储服务器的地址和端口号。用户发起查询时，会主动通过m个线程去向服务器发出查询请求。

待所有线程结束（所有服务器的查询结果已取回时），客户端根据指定的消息格式汇总查询结果，并在终端输出总的查询结果。

客户端在连接服务器、数据传输等过程中均会输出相应的日志用于调试。在一次查询结束后，输出查询总时间。

4. 总体评价

- 查询结果正确：数据完整、查询算法合理
- 存储负载均衡：每台存储服务器固定存有一份原件和一份副本。
- 查询容错：即使有服务器发生故障，客户端在另外的机器上也可以找到故障服务器副本的查询结果
- 查询时间：优化前为5-10分钟，优化后平均查询时间大约在2-7秒