

学士学位论文

中期检查报告

|  |  |
| --- | --- |
| 论 文 名 称： | **网站访客唯一性识别和智能流控方案的设计与实现** |
| 姓 名： | **江一帆** |
| 学 号： | **17373281** |
| 学院指导老师： | **吕云翔** |
| 企业指导老师： |  |
| 企业老师单位： | **清华大学出版社** |

北京航空航天大学软件学院

2021年 4 月 8 日

目 录

1 课题背景与意义 1

1.1 课题来源 1

1.2 课题目标与任务 1

1.3 主要技术与成果 2

2 课题任务进展 3

2.1 需求分析 3

2.2 系统设计 7

2.3 主要工作内容 11

2.4 任务完成情况 12

3 后期拟完成的研究工作及进度安排论文目标 13

4 参考文献 14

# 课题背景与意义

## 课题来源

论文课题由北京航空航天大学软件学院吕云翔老师提出，课题来源于吕老师于清华大学出版社联系到的对于识别用户唯一性、限制恶意爬虫爬取网站内容的实际需求。

## 课题目标与任务

随着互联网的发展与移动设备的普及，互联网用户及终端设备数量呈现爆发式的增长。一位用户可能使用多台不同的设备或经由不同的浏览器访问相同的网站。

除此之外，出于对获取各类网站上大量有价值信息的需要，网络爬虫也被大量地用于获取网络数据。根据统计，2019年，恶意爬虫的流量占所有互联网流量的24.1%，上升到有史以来最高的百分比，而37.2%的互联网流量是非人为带来的[1]。对于网站而言，恶意爬虫可能会导致网站核心内容被复制、注册用户被扫描、网站带宽负担加重等问题。

清华大学出版社是教育部主管、清华大学主办的综合性大学出版社，现年出版图书、音像制品、电子出版物等近3000种。作为国内领先的综合性教育与专业出版机构，清华大学出版社的网站在用户的唯一性识别以及反恶意爬虫等方面都有较高的需求。

本课题的目标是根据清华大学出版社提出的保障网站安全性与性能的实际需求，结合国内外研究现状和发展状态，设计并实现一个识别用户身份与爬虫，并进行网站流量控制的方案，希望通过对访客进行标记以及对恶意爬虫进行识别，对恶意用户与爬虫设置访问限制，达到智能流量控制的目的。

本课题主要包括如下几个任务：

1. 分析清华大学出版社提出的网站访客识别和智能流控具体需求。

2. 进行需求建模，设计方案的架构与数据库。

3. 对网站获取到的访客数据进行提取，整理系统需要的信息。

4. 在提取的信息中进行访客身份的识别，并在数据库中标记。

5. 设计合适的规律判断方法，对爬虫进行识别。

6. 根据流量控制要求，对频繁访问的接口进行请求次数或频率上的限制。

7. 交付清华大学出版社运行测试。

8. 根据测试情况完善方案，总结并撰写论文。

## 主要技术与成果

本课题的主要技术来自网站获取用户数据、用户识别、爬虫识别、数据整理、访问限制几方面。

* 网站获取用户数据： 使用browser对象中的window、navigator、screen对象获取用户的内核数量、屏幕等数据。使用第三方依赖获取fingerprint、字体、时区、语言、ip地址等数据。
* 网站前端数据库：在选择前端数据存储时考虑了LocalStorage、WebSQL与IndexedDB。虽然LocalStorage是用key-value键值的模式存储数据，但它都是通过字符串形式进行数据的存储，而且LocalStorage的容量 仅在 2.5MB 到 10MB 之间，并且不提供搜索功能，不能建立自定义的索引，因此对于数据的查找并不方便。此外WebSQL也已经停止维护，只有Chrome还支持WebSQL，因此采用了更为兼容的IndexedDB。IndexedDB为Nosql型数据库，支持查找、建立索引等。由于IndexedDB的API十分复杂，并且异常处理并不完善，因此使用dexie封装的API来简化对IndexedDB的操作。
* 版本控制：使用Github用于管理项目的代码和说明文档。
* 网站后端数据库：为了与前端数据库IndexedDB，在后端数据库上采用了同为Nosql型数据库的MongoDB，通过Node.js控制对MongoDB的增删改查等操作。
* 依赖管理：使用CDN与npm进行第三方的依赖管理。

最终成果应该是一款可以接入原有网站，通过一定的浏览器与HTTP数据对用户与爬虫进行识别并进行针对性访问限制的系统。此外，还需要编写项目的说明文档，提供代码以便后续维护。

# 课题任务进展

## 需求分析

本项目的需求主要集中在数据的收取与处理方面，可分为如下几部分：记录用户浏览器数据、标记用户、记录用户请求数据、标记爬虫以及截至接口访问限制。项目的用例图如图2-1所示。



图 2‑1 用例图

记录用户浏览器数据的用例说明见表2-1。

**表2-1 记录用户浏览器数据**

|  |  |
| --- | --- |
| **用例名称** | 记录用户浏览器数据 |
| **简要描述** | 该用例描述了系统记录用户浏览器数据的过程 |
| **参与者** | 网站管理员 |
| **涉众** | 用户 |
| **前置条件** | 用户进入到网站 |
| **后置条件** | 记录成功后进行用户标记 |
| **基本事件流** | |
| 1. 用户进入网站 2. 系统收集用户浏览器等数据，包括useragent、ip地址、fingerprint等。（A-2） | |
| **备选事件流**  A-2 存在无法获取的数据  1.以undefined代替 | |
| **补充约束**  （暂无） | |
| **待解决问题**  （暂无） | |

标记用户的用例说明见表2-2。

**表2-2 标记用户**

|  |  |
| --- | --- |
| **用例名称** | 标记用户 |
| **简要描述** | 该用例描述了系统根据获取的数据标记用户的过程 |
| **参与者** | 网站管理员 |
| **涉众** | 无 |
| **前置条件** | 已经获得用户浏览器等数据 |
| **后置条件** | 标记成功后保存到数据库中 |
| **基本事件流** | |
| 1. 传入收集到的用户浏览器等信息 2. 根据数据区分用户后，存入IndexedDB中，并同步到MongoDB。（B-2） | |
| **备选事件流**  B-2 同一用户已存在   1. 检查数据是否重复 2. 添加不重复的新数据 | |
| **补充约束**  （暂无） | |
| **待解决问题**  （暂无） | |

记录用户请求数据的用例说明见表2-3。

**表2-3 记录用户请求数据**

|  |  |
| --- | --- |
| **用例名称** | 记录用户请求数据 |
| **简要描述** | 该用例描述了系统记录用户请求数据的过程 |
| **参与者** | 网站管理员 |
| **涉众** | 用户 |
| **前置条件** | 用户进入到网站 |
| **后置条件** | 记录成功后进行爬虫标记 |
| **基本事件流** | |
| 1. 用户进入网站 2. 系统收集用户请求数据，包括请求的header、请求的频率等。（C-2） 3. 计算请求的频率 | |
| **备选事件流**  （暂无） | |
| **补充约束**  （暂无） | |
| **待解决问题**  （暂无） | |

标记爬虫的用例说明见表2-4。

**表2-4 标记爬虫**

|  |  |
| --- | --- |
| **用例名称** | 标记爬虫 |
| **简要描述** | 该用例描述了系统根据获取的数据标记爬虫的过程 |
| **参与者** | 网站管理员 |
| **涉众** | 无 |
| **前置条件** | 已经获得用户请求数据 |
| **后置条件** | 标记成功后保存到数据库中 |
| **基本事件流** | |
| 1. 传入收集到的用户请求信息 2. 根据信息进行频率、合法性等检查以区分爬虫 3. 区分爬虫后，存入MongoDB中。（D-2） | |
| **备选事件流**  D-2 同一爬虫标记已存在   1. 检查数据是否重复 2. 更新请求频率、访问限制等信息 | |
| **补充约束**  （暂无） | |
| **待解决问题**  （暂无） | |

设置访问限制的用例说明见表2-5。

**表2-5 设置访问限制**

|  |  |
| --- | --- |
| **用例名称** | 设置访问限制 |
| **简要描述** | 该用例描述了系统根据爬虫标记限制访问的过程 |
| **参与者** | 网站管理员 |
| **涉众** | 无 |
| **前置条件** | 已经完成爬虫标记 |
| **后置条件** | 无 |
| **基本事件流** | |
| 1. 传入需要限制的用户数据。 2. 根据数据对请求进行限制。（D-2） | |
| **备选事件流**  D-2 同一用户已被限制   1. 更新限制信息 | |
| **补充约束**  （暂无） | |
| **待解决问题**  （暂无） | |

## 系统设计

用户识别部分拟采用IP地址与浏览器fingerprint相结合的方式。

Fingerprint是通过获取浏览器具有辨识度的信息，进行计算得出的哈希值。特征值可以是User Agent、时区、地理位置或者是使用的语言等。这些特征值具有不同的信息熵，信息熵大的特征值对于唯一确定一位用户更有作用，往往在计算fingerprint时会有的更大的权重。

将特征值与权重结合计算后可以生成一个Fingerprint哈希值，用于标记不同的用户，这里采用了ClientJS提供的计算方法，通过clientjs.getFingerprint()方法获取到fingerprint的哈希值。采用的特征值包括：user agent、screen print、color depth、current resolution、available resolution、device XDPI、device YDPI、plugin list、font list、local storage、session storage、timezone、language、system language、cookies、canvas print。

Fingerprint在跨浏览器识别用户时，相比于User Agent等也更具参考价值，可以更准确的区分不同的用户设备。但受限于其精度，本项目将IP地址与fingerprint结合作为用户判断的依据。在同一个fingerprint下保存用户的不同IP地址。同时，用户的可用核心、User Agent、时区等其他信息也被保存在数据库中，便于需要时进行进一步的区分。数据以JSON的形式保存在NoSQL型数据库中，前端为IndexedDB，后端为MongoDB。

在后端采用node.js，对MongoDB的增删改查操作进行了封装，便于数据的整理。数据库表内容的设计如表2-6所示。

**表2-6 数据库设计**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 表名 | 数据说明 | 功能说明 |
| 1 | users | 存储fingerprint与ip | 将fingerprint作为用户身份的标记 |
| 2 | fingerprints | 存储fingerprint、User Agent、核心数、时区、语言、字体、分辨率等 | 存储其他表征用户的特征值 |
| 3 | userfreqs | 存储用户在时间段内的访问次数、上一次访问时间等 | 用于计算用户访问频率等，判断是否爬虫 |
| 4 | Blocks | 存储用户与限制时间 | 用于限制用户请求 |
| 5 | Whilelists | 存储白名单用户 | 用于放行可能会被识别为爬虫的白名单用户 |

对于爬虫检测的部分，采用设置阈值、HTTP请求检测的方式。在Node.js端对容易被频繁起请求的接口处接入爬虫检测。通过记录时间段内的请求次数以及最近的访问时间，将访问过于频繁的用户进行标记。对于请求中User Agent、platform等信息与记录不匹配的用户也进行标记。将受标记的用户与白名单用户进行匹配，去除需要放行的用户后，对于其他受标记的用户进行频率与流量速率限制。

为了加快请求响应速度与访问限制的灵活性，将影响严重的用户加入nginx的blockip配置文件中，并适时进行更新。

在访问限制部分，过时的用户代理、过时的浏览器版本、不合理的海外IP、不合理的Referer都会被特别标记，在需要时可以在这些方面设置更为严格的限制策略。

系统的流程图如图2-2与图2-3所示。

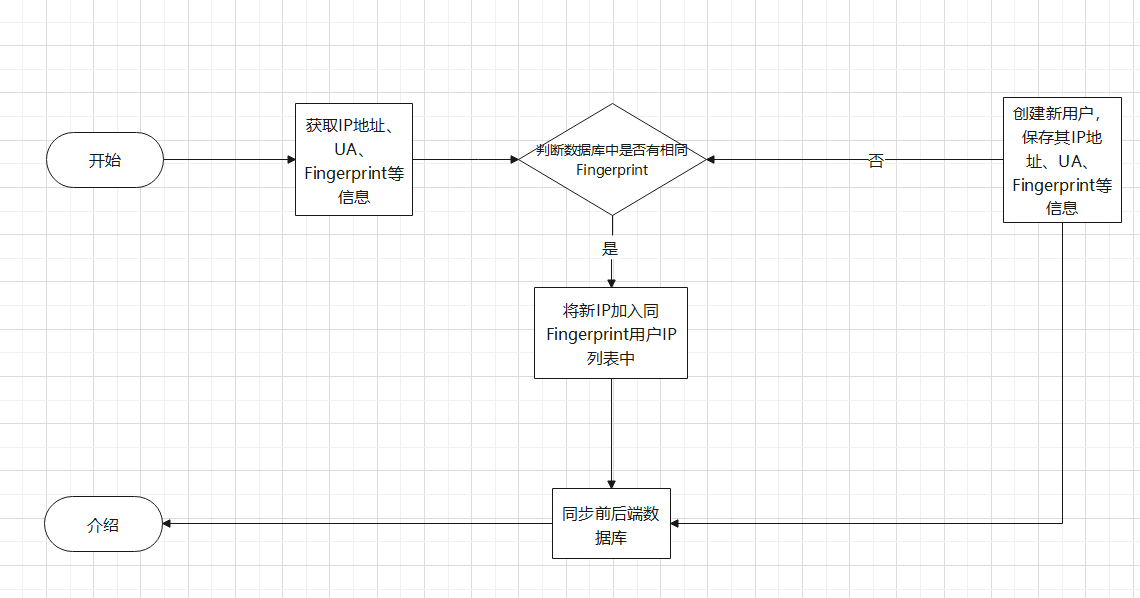


图 2‑2 用户识别流程图

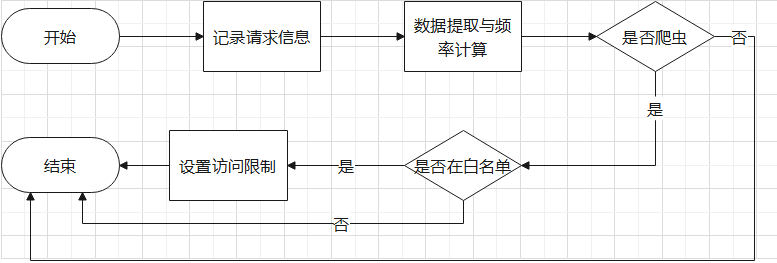


图 2‑3 爬虫识别流程图

系统架构图如图2-4所示。

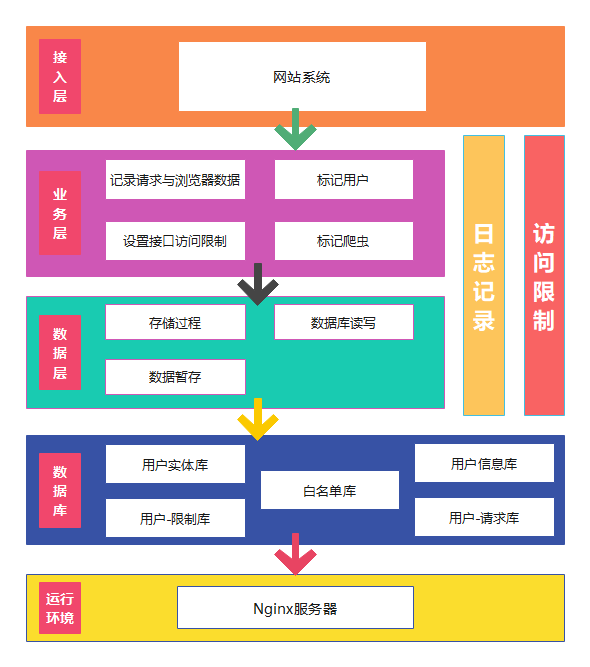


图 2‑4 架构图

## 主要工作内容

项目的主要工作是完成网站访客唯一性识别和智能流控方案的设计与实现，包括系统设计，数据库设计，接口实现等。具体工作内容如表2-7所示。

**表2-7 具体工作内容**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 任务 | 完成过程 | 完成情况 | 可能存在的改动 |
| 1 | 需求分析 | 与清华大学出版社负责人进行交流 | 完成 | 部分需求需细化 |
| 2 | 系统整体设计 | 根据需求划分了了不同的接入形式 | 完成 | 各部分之间衔接可以优化 |
| 3 | 数据库设计 | 根据需求设计数据库 | 完成 | 部分表可能修改 |
| 4 | 用户标识信息收集 | 编码完成用户标识信息收集部分 | 完成 | 无 |
| 5 | 用户标记与存储 | 编码完成用户标记与存储部分 | 完成 | 数据存储可以优化 |
| 6 | 用户请求信息收集 | 编码完成用户请求信息收集部分 | 完成 | 无 |
| 7 | 爬虫标记与存储 | 编码完成爬虫标记与存储部分 | 完成 | 无 |
| 8 | 限制请求次数 | 编码完成限制请求次数部分 | 完成 | 限制可以逐级加强 |

## 任务完成情况

项目的具体完成情况见表2-8。

**表2-8 计划进度表**

|  |  |  |  |
| --- | --- | --- | --- |
| 开始日期 | 结束日期 | 工作内容 | 完成情况 |
| 2020.11.17 | 2020.11.26 | 与清华大学出版社负责人沟通确认需求，同时进行访客识别与流量控制方法的调研。 | 完成 |
| 2020.11.27 | 2020.12中旬 | 了解国内外研究现状。撰写开题报告，准备开题答辩。 | 完成 |
| 2020.12下旬 | 2020.12.31 | 明确需求细节。搭建开发环境。 | 完成。进度不完全相同，在开发过程中有所改动。 |
| 2021.01.01 | 2021.02.01 | 设计用户唯一性识别与智能流控具体方案。设计数据库。完成初步数据清洗。 | 完成。进度不完全相同，穿插在不同部分的开发中完成。 |
| 2021.02.01 | 2021.03.01 | 完成用户识别部分编码工作。 | 完成 |
| 2021.03.01 | 2021.04.01 | 完成用户识别部分编码工作。 | 完成 |

# 后期拟完成的研究工作及进度安排论文目标

目前本项目的设计与实现已基本完成，后期会完善系统功能整合，优化数据整理等。具体安排见表3-1。

**表3-1 后期任务计划表**

|  |  |  |  |
| --- | --- | --- | --- |
| 开始日期 | 结束日期 | 工作任务 | 完成计划 |
| 2020.04.01 | 2020.04.15 | 完成服务器端请求限制，整合系统功能。 | 编码实现，进行中。 |
| 2020.04.15 | 2021.04.30 | 测试并交付清华大学出版社。完善毕业设计论文。 |  |

# 参考文献

[1] Edward Roberts.Bad Bot Report 2020: Bad Bots Strike Back[R].San Mateo: Imperva,2020.

[2] 林中明.基于Hadoop的Web用户识别与新闻智能推荐算法研究[D].郑州：战略支援部队信息工程大学,2018.

[3] 伏康，杜振鹏.网站反爬虫策略的分析与研究[J].电脑知识与技术，2019(28)：28-30.

[4] 张晔，孙光光，徐洪云，庞婷，曲潇洋.国外科技网站反爬虫研究及数据获取对策研究[J].竞争情报，2020(01)：24-28.

[5] Y. Liu, Z. Yang, J. Xiu, C. Liu. Research on an anti-crawling mechanism and key algorithm based on sliding time window[A]. In: 2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)[C].Beijing: IEEE, 2016: 220-223

[6] W. Zhu, J. Qin, R. Kong, H. Lin, Z. He. A System Framework for Efficiently Recognizing Web Crawlers[A]. In: IEEE SmartWorld 2018 Organizing and Program Committees. 2018 IEEE SmartWorld [C].Guangzhou: IEEE, 2018: 1130-1133

[7] H. Wang, C. Li, L. Zhang, M. Shi.Anti-Crawler strategy and distributed crawler based on Hadoop[A]. In: 2018 IEEE 3rd International Conference on Big Data Analysis(ICBDA)[C]. Shanghai: IEEE, 2018: 227-231

[8] P. Lewandowski, M. Janiszewski, A. Felkner. SpiderTrap—An Innovative Approach to Analyze Activity of Internet Bots on a Website[J]. IEEE Access,2020, 8: 141292-141309

[9] G. Neelima, S. Rodda. Predicting user behavior through sessions using the web log mining[A]. In: 2016 International Conference on Advances in Human Machine Interaction (HMI) [C]. Doddaballapur: IEEE, 2016:1-5

[10] 刘洋. 基于网页浏览行为的反爬虫研究[J]. 现代计算机(专业版)，2019(07)：50-60+70.

[11] Cao, Y., Li, S., & Wijmans, E. (Cross-)Browser Fingerprinting via OS and Hardware Level Features[R]. San Diego: NDSS,2017.