**文献综述**

# 引言

随着信息社会的高速发展，移动个人数据、社交网站、科学计算、证券交易、网站日志、传感器网络等产生的数据量迎来爆发式增长。单机设备已经无法处理庞大的甚至TB、PB级别的数据[1]。相对于传统的数据处理方法，大数据处理方法采用多机器、多节点来处理海量数据。该方法特点是随着数据量的不断加大，可以增加机器数量，水平扩展。一个大数据系统，可以多达几万台机器甚至更多。

高校学者是全球科研力量的最重要组成部分，他们的科研成果推动着人类科技的进步。近年来，新兴领域和学科大量涌现，科技界呈现出百花齐放的繁荣景象。高校学者的科研方向也日趋多元化和精细化。借助于大数据的分布式存储技术，我们可以对学者信息进行高容错性、高扩展性地存储；借助于大数据的分布式计算技术，我们可以对高校学者的研究方向以及相关科研领域的热门程度进行统计分析。统计分析的结果对于人们了解当前科研趋势以及相关人员选择科研方向具有重要的参考价值。

# 大数据

大数据（big data），指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合，是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产。

## 大数据的发展阶段

1. 第一个阶段是萌芽期[1]，这一阶段随着数据挖掘理论和数据库技术的逐步成熟，一批商业智能工具和知识管理技术开始被应用，如数据仓库、专家系统、知识管理系统等，企业、机构对内部数据进行统计、分析、挖掘利用。
2. 第二个阶段是成熟期[1]，这一阶段非结构化数据大量产生，传统处理方法难以应对，带动了大数据技术的快速突破，大数据解决方案逐渐走向成熟，形成了并行计算与分布式系统两大核心技术，谷歌的GFS和MapReduce等大数据技术受到追捧，Hadoop平台开始大行其道。
3. 第三阶段是大规模应用期[1]，大数据应用渗透各行各业，数据驱动决策，信息社会智能化程度大幅提高，同时将出现跨行业、跨领域的数据整合，甚至是全社会的数据整合，从各种各样的数据中找到对于社会治理、产业发展更有价值的应用。

## 大数据的特点

1. 体积量大[2]：伴随着各种随身设备、物联网和云计算、云存储等技术的发展，人和物的所有轨迹都可以被记录，数据因此被大量生产出来。移动互联网的核心网络节点是人，不再是网页。人人都成为数据制造者，短信、微博、照片、录像都是人创造出的数据产品。
2. 多样性[2]：海量数据的产生来自于不同的途径，不同方式随着传感器、智能设备以及社交协作技术的飞速发展，组织中的数据也变得更加复杂，因为它不仅包含传统的关系型数据，还包含来自网页、互联网日志文件（包括点击流数据）、搜索索引、社交媒体论坛、电子邮件、文档、主动和被动系统的传感器数据等原始、半结构化和非结构化数据。
3. 低价值密度[2]：有价值的数据和整个数据总量比，可能只占很小的一部分，如何通过强大的机器算法更迅速地完成数据的价值“提纯”成为目前大数据背景下亟待解决的难题。比如炒股的人，在微博、贴吧等互联网上会对股票涨跌，各种内幕来做评论，但是这些评论数据有多大的参考价值呢。我们就需要从这海量的评论中挖掘出真正有价值的数据，比如庄家发表的评论。
4. 处理速度快[2]：在数据处理速度方面，有一个著名的“1秒定律”，即要在极短的时间范围内给出分析结果，超出这个时间，数据就失去了价值。越来越多的数据挖掘趋于前端化，这里的前端化，就是对客户服务的时候，就不断在收集数据的同时，展开分析、挖掘，并给出个性化的服务。例如，对绝大多数商品来说，找到顾客“触点”的最佳时机并不是在顾客结完账以后，而应该是在顾客还提着篮子逛街的时候。比如你逛淘宝，淘宝会从你的点击流、浏览历史、还有放入购物车、收藏的这些动作、这些痕迹中实时发现你的购买意图和兴趣，并且马上推送对应的商品，这就是“处理速度快”的价值。

# 网络爬虫

网络爬虫是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本[3]，它们被广泛用于互联网搜索引擎或其他类似网站，可以自动采集所有其能够访问到的页面内容，以获取或更新这些网站的内容和检索方式。

## 爬虫原理

网络爬虫系统的功能是下载网页数据，为搜索引擎系统提供数据来源。很多大型的网络搜索引擎系统都被称为基于 Web数据采集的搜索引擎系统，比如 Google、Baidu[3]。由此可见Web 网络爬虫系统在搜索引擎中的重要性。网页中除了包含供用户阅读的文字信息外，还包含一些超链接信息。Web网络爬虫系统正是通过网页中的超连接信息不断获得网络上的其它网页。正是因为这种采集过程像一个爬虫或者蜘蛛在网络上漫游，所以它才被称为网络爬虫系统或者网络蜘蛛系统，在英文中称为Spider或者Crawler[3]。

## 爬虫的工作流程

1. 选取一部分精心挑选的种子URL。
2. 将这些URL放入待抓取URL队列。
3. 从待抓取URL队列中取出待抓取在URL，解析DNS，并且得到主机的ip，并将URL对应的网页下载下来，存储进已下载网页库中。此外，将这些URL放进已抓取URL队列。
4. 分析已抓取URL队列中的URL，分析其中的其他URL，并且将URL放入待抓取URL队列，从而进入下一个循环。

# 关键技术

## Python

Python是一种动python是一种动态的、面向对象的脚本语言，最初被设计用于编写自动化脚本，随着版本的不断更新和语言新功能的添加，越来越多被用于独立的、大型项目的开发。

python语言简洁，语义清晰，符合一般人的思维。很多人选择python作为他们的编程入门语言。

python拥有胶水语言的特性，因此它的第三方库非常丰富。在软件开发领域中，有pyqt库来开发桌面应用程序，有flask库开发web后台，有scrapy开发爬虫，有scapy做服务器运维等等。在科研领域中，有numpy、pandas库做数值计算，有matplotlib库做数据可视化，有opencv库做视觉处理，有tensorflow、pytorch库训练模型等等。本系统就用到了requests、redis、pymysql、flask、beautifulsoup等这些库。

python可移植行强，如果你在windows机器上开发好了代码，那么你的python程序无需修改就可以放到服务器上运行。一套代码，多端共用。

综上而言，对于编写爬虫脚本，python是最佳选择。Python提供了丰富的爬虫库，例如scrapy、beautifulsoup、selenium等等。这些库功能强大、文档齐全、api简洁友好，几行代码即可达到爬取页面的效果。

## Docker

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中[3]，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。

通过docker，我们可以将网络爬虫的python脚本封装到容器中。宿主机只需安装docker客户端，然后拉取封装了爬虫的容器，这样就可在本地运行爬虫镜像。利用docker，我们屏蔽了各个宿主机运行环境的差异，为网络爬虫提供了统一的运行环境。

## Redis

Redis 是一个开源的，内存中的数据结构存储系统。它可以用作数据库、缓存和消息中间件。它支持多种类型的数据结构，如字符串，散列，列表，集合，有序集合与范围查询，bitmaps，hyperloglogs和地理空间索引半径查询。Redis 内置了复制，LUA脚本，LRU驱动事件，事务和不同级别的磁盘持久化，并通过Redis哨和自动分区提供高可用性。

由于Redis的数据是存放在内存里的，因此可以将网络爬虫的页码和偏移量存放在Redis中，这样就可以快速获取页码和偏移量。另外，Redis读写速度快，很适合做session共享。

## CentOS

CentOS是Community Enterprise Operating System的缩写。是企业Linux发行版领头羊Red Hat Enterprise Linux的再编译版本。RHEL是很多企业采用的Linux发行版本。CentOS的开发者们使用Red Hat Linux的源代码创造了一个和RHEL近乎相同的Linux[4]。但是一切和RedHat有关的商标都被去除了，因为RedHat不允许他们这样做。CentOS是免费的，你可以使用它像使用RHEL一样去构筑企业级的Linux系统环境，但不需要向RedHat付任何的费用。CentOS的技术支持主要通过社区的官方邮件列表、论坛和聊天室。

相对于图形化的windows，CentOS是无图形化界面的。由于图形渲染要占用一定的CPU和内存资源，无图形化的CentOS保持了最简洁的微内核形式，从而使得系统性能优越、运行非常流畅。这也是CentOS称为服务器领域霸主的最重要原因。我们的网络爬虫脚本即是放在CentOS上运行的。

## HDFS

Hadoop Distributed Filesystem：Hadoop分布式文件系统(HDFS)被设计成适合运行在通用硬件(commodity hardware)上的分布式文件系统[1]。HDFS有着高容错性的特点，并且设计用来部署在低廉的硬件上[5]。它提供高吞吐量来访问应用程序的数据，适合那些有着超大数据集的应用程序。HDFS放宽了POSIX的要求这样可以实现流的形式访问（streaming access）文件系统中的数据。

HDFS是通过增加副本的数量来提高它的容错性的。如果某一个副本丢失，HDFS机制会复制其他机器上的副本。文件一旦写入，不能修改，只能增加。这样可以保证数据的一致性。

## MapReduce

MapReduce是一种编程模型，用于大规模数据集（大于1TB）的并行运算。概念"Map（映射）"和"Reduce（归约）"[6]，是它们的主要思想，都是从函数式编程语言里借来的，还有从矢量编程语言里借来的特性。它极大地方便了编程人员在不会分布式并行编程的情况下，将自己的程序运行在分布式系统上。当前的软件实现是指定一个Map（映射）函数，用来把一组键值对映射成一组新的键值对，指定并发的Reduce（归约）函数，用来保证所有映射的键值对中的每一个共享相同的键组[6]。

## MySQL

MySQL是一个关系型数据库管理系统。关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库[7]。目前MySQL8.0已经支持文档存储、事务性数据字典、通用关系表达式等符合现代化需求的功能。

相对于Oracle和Sql Server，MySQL是一个轻量级的数据库，可定制性强。所以MySQL受到了Alibaba、Facebook、Google等大公司的青睐。支持MySQL的第三方客户端工具也很多，例如Navicat、DataGrip、Workbench，这些工具界面友好、操作简单，非常适合DBA用来数据库开发。

## Node.js

Node.js采用Google开发的V8运行代码，使用事件驱动、非阻塞和异步输入输出模型等技术来提高性能，可优化应用程序的传输量和规模。这些技术通常用于数据密集的即时应用程序[8]。Node.js大部分基本模块都用JavaScript语言编写。在Node.js出现之前，JavaScript通常作为客户端程序设计语言使用，以JavaScript写出的程序常在用户的浏览器上运行。Node.js的出现使JavaScript也能用于服务端编程。Node.js含有一系列内置模块，使得程序可以脱离Apache HTTP Server或IIS，作为独立服务器运行[8]。

Node给前端带来了春天，使得前端有了明确的定义。借助于Node，前后端分离成为现实。在传统的开发模式中，前后端代码是存放在一个项目中，前后端开发界限不清晰。在传统的项目部署中，整个项目是部署在一台服务器上，如果这台服务器宕机，那么整个项目便无法访问。前后端分离后，在开发上，前端和后端程序员各司其职，前后端通过网络来通信，这样就极大地提高了开发效率；在部署上，前后端分别部署在不同的服务器上，即使后端服务器宕机，前端页面仍可访问，这样就提高了用户体验。

## Express

Express 是一个简洁而灵活的 node.js Web应用框架, 提供了一系列强大特性帮助你创建各种 Web 应用，和丰富的 HTTP 工具。使用 Express 可以快速地搭建一个完整功能的网站。Express 框架核心特性：可以设置中间件来响应 HTTP 请求。定义了路由表用于执行不同的 HTTP 请求动作[9]。可以通过向模板传递参数来动态渲染 HTML 页面。

# 总结

对于单机无法解决的问题，综合利用多台普通机器要比打造一台超级计算机更加可行[10]，这就是大数据技术的处理思路。大数据技术已经成为当今数据处理挖掘行业广泛使用的主流技术。

网络爬虫是一种获取外界数据的有效手段。对于缺乏真实数据的学生而言，爬虫是最佳选择。但随着各大网站反爬措施的加强，爬取这些网站的难度越来越大，这对学生的编程技能提出了更高的要求。

通过阅读这些文献，极大地拓宽了我的知识面。也让我对毕设工程的实现有了清晰的思路：利用网络爬虫爬取高校学者的数据，然后使用MapReduce对数据进行统计分析，最后以web形式将这些结果可视化。

# 参考文献

1. 余明辉, 张良均. Hadoop大数据开发基础[M]. 北京: 人民邮电出版社, 2018.
2. 安俊秀, 王鹏, 靳宇倡. Hadoop大数据处理技术基础与实践[M]. 北京: 人民邮电出版社, 2015.
3. 崔庆才. Python3网络爬虫实战[M]. 北京: 人民邮电出版社, 2018.
4. Silberschatz, Galvin, Gagne. 操作系统概念[M]. 北京: 高等教育出版社, 2007.
5. 林大贵. Python+Spark2.0+Hadoop机器学习与大数据实战[M]. 北京: 清华大学出版社, 2017.
6. J. Dean and S. Ghemawat.MapReduce: simplified data processing on large clusters[C]. volume 51, pages 107–113, New York, NY, USA, 2008. ACM.
7. Hector Garcia-Molina, Jeffrey D.Ullman, Jennifer Widom. 数据库系统实现[M]. 北京: 机械工业出版社, 2010.
8. Marc Harter. Node.js实战[M]. 北京: 人民邮电出版社, 2014.
9. 谢希仁. 计算机网络[M]. 北京: 电子工业出版社, 2017.
10. Thomas H.Cormen, Charles E.Leiserson, Ronald L.Rivest Clifford Stein. 算法导论[M]. 北京: 机械工业出版社, 2013.