Computer Science and Application 计算机科学与应用, 2020, 10(11), 1927-1937
Published Online November 2020 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2020.1011203

Hans汉斯

多种技术融合的API接口数据采集策略

杨　光，吴明芬*

五邑大学智能制造学部，广东　江门
Email: 1691327650@qq.com, *mfwu89@foxmail.com

收稿日期：2020年10月18日；录用日期：2020年11月2日；发布日期：2020年11月7日

摘　要

数据开放共享在大数据时代变得越来越重要，API接口在数据共享上扮演着重要的角色，如何从开放共享的API接口快速、高效、便捷地获取数据是迫切需要解决的问题。本文从实用性的角度出发，融合了自动化测试技术、最优线程、Python和ETL技术等，构建了一种基于API接口的数据采集策略，该策略采集速度快、操作简单、线程可控制并推导出数据采集时间公式，该公式在5个线程以上准确率达90%以上，在7~8个线程准确率达97%，在9~10个线程准确率可达99%，在采集之前就可通过该公式以最合理的线程计算出最合理的采集时间，极大地节省采集时间。

关键词

数据开放共享，数据采集，API接口，自动化测试，最优线程

Data Acquisition Strategy of API Interface Based on Multi-Technology Integration

Guang Yang, Mingfen Wu*

Intelligent Manufacturing Department, Wuyi University, Jiangmen Guangdong
Email: 1691327650@qq.com, *mfwu89@foxmail.com

Received: Oct. 18th, 2020; accepted: Nov. 2nd, 2020; published: Nov. 7th, 2020

Abstract

Data open sharing is becoming more and more important in the era of big data. API interfaces play an important role in data sharing. How to quickly, efficiently and conveniently obtain data from open and shared API interfaces is an urgent problem to be solved. From the perspective of practi-

*通讯作者。

文章引用: 杨光, 吴明芬. 多种技术融合的 API 接口数据采集策略[J]. 计算机科学与应用, 2020, 10(11): 1927-1937.
DOI: 10.12677/csa.2020.1011203

cability, this article integrates automated testing technology, optimal threading, Python and ETL technology, etc., and constructs a data collection strategy based on API interface, which has fast collection speed, simple operation, thread control and deducing and formulating the formula of data collection time. This formula has an accuracy rate of more than 90% for more than 5 threads, an accuracy rate of 97% for 7 - 8 threads, and an accuracy rate of 99% for 9 - 10 threads. Through this formula, the most reasonable acquisition time is calculated with the most reasonable thread, which greatly saves the acquisition time.

Keywords

Open Data Sharing, Data Collection, API Interface, Automated Testing, Optimal Thread

1. 引言

随着我国物联网、5G、区块链[1]、新基建[2]、数据湖[3]和大数据技术的发展，据 IDC (International Data Corporation)预测全球数据圈 2025 年将增至 175 ZB (1 泽字节相当于 1 万亿 GB)，接近于 2018 年 33 ZB 的 5.3 倍[4]，并且到 2025 年我国将成为数据量最大的区域，如今的经济高度依赖数据，并且随着各公司在其供应链[5]的每一步都对数据进行采集、编目和利用，对数据的依赖在未来只会不断加强。

目前数据采集主要来自于自身内部数据，或者网上数据也就是网络爬虫，自身内部的数据我们无法改变，但在互联网上人们几乎可以获取任何想要的数据，从互联网上获取数据一般分为从网页上获取数据或者通过数据 API (Application Programming Interface，应用程序编程接口) [6]获取数据，与网页爬虫不同的是，数据 API 的设计更为简单高效，这个接口已经存储好大家所需要的数据，不需要再花过多的精力去解析网页。并且网页数据爬取往往会对服务器造成压力，如果代码没有设置好合理的类人的网页浏览频率，会有 IP 被封的风险[7]。目前，API 接口支持以下数据格式 XLSX、JSON、XML、CSV、RDF 等，在本文中选择了一种易于读写并广泛应用于 Web 的轻量级数据格式 JSON，它可以被快速、高效地解析，也易于存储。

获取到 API 接口先不用忙着怎么去将数据采集下来，可以先对 API 接口的服务文档、响应示例、并发数、响应时间等进行分析，这些准备工作对大量 API 接口数据的采集是非常有必要的。服务文档可以帮助我们获取到网站的 URL 以及访问方式；响应示例可以看到接口响应的数据格式、内容、编码等；并发数可以看到 API 接口每秒并发运行的次数，接近这个次数网站就会预警，超过这个次数程序就会报错；接口响应时间是数据采集快慢的关键，虽然程序可以多线程或者多进程，但一次接口响应时间就花了近 1 秒，这个采集速度怎么能快呢！自动化测试的目的就是为了测试接口的响应时间，网上的 API 接口很多不会像“大厂”，比如百度 API、阿里 API、腾讯 API、微博 API 那样规范稳定，而且接口响应速度还不会太快。百度 API 的单个响应接口速度可以达 60 ms (1 s = 1000 ms)左右，多个线程一起跑响应速度成倍数下降，但像国家的政务查询接口，响应速度都在 1 秒左右，此速度是通过测试广东常住人口接口和国家企业信息接口而来，速度慢的原因应是数据量过大，查询费时多。

最优线程[8]是为了使数据采集速度达到最优化而加入的，既然 API 接口的响应速度无法改变，就通过运行多个线程变相地降低响应时间，但线程不能是无限增加的，所以计算出最优线程可以节约计算机

的性能，还能使数据采集更加快。Python 的简单，以及支持一系列的第三方库，使得 Python 在数据采集方面用途很广，但 Python 的速度慢，以及线程支持不友好，使得对有一定速度要求的数据采集接口变得不是很实用，而基于 Java 的 ETL 工具 Kettle [9]，可以自由调节线程数量，而且易于实现，故选择了它作为数据采集的工具，通过实验对比这两种数据采集的方式，可以看出基于 Java 的 ETL 工具 Kettle 采集速度更加快，还可以定时数据采集，本文提出的基于 API 接口的数据采集策略，符合常规的数据采集逻辑，而且数据采集快，易于发现数据采集中哪个环节出现问题。

2. 相关技术介绍

本文主要涉及到 API 接口、自动化接口测试、最优线程、Python、Kettle 等技术，故在此进行一些介绍为后续的工作做好铺垫。

API 是应用程序接口(Application Programming Interface)的简称，API 是一些功能、定义或者协议的集合，提供应用程序或者程序开发人员基于软件访问一组例程的能力，对外封装完善，调用时无需学习 API 内部源码，依据 API 文档功能说明书来使用即可[10]。针对 API 接口数据采集也是网络爬虫的一种，但从 API 中获取的数据格式更加规范，数据质量更加好，可以相对的避免复杂的数据清洗过程，随获随用，极大的节省时间，因为 API 设计的目的就是为了实现数据的共享，开放数据。

目前 API 接口支持 XLSX、JSON、XML、CSV、RDF 等数据格式，其中 JSON 和 XML 是主流的数据格式，几乎所有 API 接口都支持这两种数据格式，JSON (JavaScript Object Notation)是一种轻量级的数据交换格式，具有良好的可读和便于快速编写的特性，可在不同平台之间进行数据交换[11]。XML 是扩展标记语言(Extensible Markup Language)，用于标记电子文件使其具有结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。JSON 与 XML 相比是一种更加轻量级的数据格式，而且更加易于解析，支持多种语言，这使得 JSON 在大数据时代备受欢迎，而且随着应用程序和平台的不断发展，应用程序的功能变得越来越复杂，但为了保证用户体验的优化，需要通过重构代码，将复杂的逻辑封装在内部，保持其对外提供的 API 仍然简洁。JSON 也正因为简洁这一优势逐渐超越了 XML，成为了应用间的首选数据交换格式[12]。

自动化接口测试是对接口测试的辅助手段，借助工具和测试用例达到模拟手工测试的效果，测试过程中无需人工干预[13]。自动化接口测试，在很大程度上可以减轻测试人员的压力，大大提高了软件测试工作的效率。同时，避免了人为操作带来的失误，一定程度上提高了测试的准确性[14]。在本文中自动化接口测试一方面是为了检测接口是否可用，另一方面是为了测试接口的响应速度，不同的 API，由于性能和功能的不同响应速度相差很大，像类似于百度 API 的响应时间在 60 ms 左右，而像目前的政务信息资源共享的接口，目前响应速度在 1 s 以上，这个速度是通过多次测试广东常住人口接口和国家企业信息接口得来的，从省级接口和国家级接口采集数据的目的是为了完善某地市的自然人和法人库，使得数据可以互联互通互享，打破信息孤岛，使得人民群众，去政府办事真正做到只跑一次，提高政府的办事效率，真正的实现数字政府。

目前的自动化测试工具有很多，比如 Postman、Appium、Selenium、QTP、Loadrunner、RobotFramework 等，在本文中选择使用 Postman 作为接口自动化测试的工具，一是因为它提供功能强大的 Web API 和 HTTP 请求调试，二是因为开源，三是 Postman 可以很好的模拟并向 API 发送 Request 请求，接收 API 发回的 Response，并可以自动生成 Js 函数对返回结果做断言，接口自动化测试实现比较容易。

通过接口自动化测试，可以知道 API 接口返回的数据内容以及格式，并且可以发现影响 API 数据采集快慢的关键，响应时间。接口的响应时间在数据采集时，接口就已经设定好了，采集方是不能进行修改，为了解决接口响应速度慢以及采集速度慢，引入了线程这个概念，本来采集时是一次采集一条，引

入线程后一次可以采集多条，相当于成倍的缩短响应时间，进而提高采集的速度。通过设置多个线程可以加快采集速度，但过多的线程会造成阻塞和资源浪费，过少的线程达不到速度的优化，故需要计算采集的最佳线程[15]使得采集速度达到最佳，最佳线程计算公式如下：

$$N_{threads} = N_{cpu} * \frac{T_{waiting} + T_{cpu}}{T_{cpu}} \tag{1}$$

$N_{threads}$ 指最佳线程数量，N_{cpu} 指 CPU 的数量，$T_{waiting}$ 指线程等待时间，T_{cpu} 指线程 CPU 时间。在平常的使用中用 CPU 的数量，即可达到较优的速度。

本文用到的采集工具是 Python 和 Kettle，Python 是目前最流行的爬虫语言，Python 从本质上来说属于一种开源编程语言，其具有功能强大、语法简便、适用性良好等优势特性[16]。Kettle 是一款国外开源的 ETL 工具，使用图形界面进行可视化 ETL 操作，支持广泛的数据库类型与文本格式的输入与输出，同时支持定时和循环，具有跨平台、可拓展、可集成、高性能等优点[17]。ETL 是用来描述将数据从来源端经过抽取(extract)、转换(transform)、加载(load)至目的端的过程[18]，本文创新性的用 Kettle 做数据采集工具，与 Python 编写的程序作对比，主要原因是 Kettle 数据采集相对简单，不需要复杂的代码进行调优，而且线程数量可控，采集比较稳定，如果采集的数据质量不好还能直接进行清洗，但在程序灵活度方面不如 Python。

采集完成的数据需要进行保存以方便后续的使用，常用的保存格式有文本格式和数据库格式，文本格式主要有 CSV、TXT、EXCEL，数据库格式又分为关系型数据库和非关系数据库(NoSQL)，关系型数据库主要有 MySQL、SQL server、Oracle 等，非关系型数据库主要有 MongoDB、Redis、Hbase 等，因为采集下来的数据是 JSON 格式，选择非关系型数据库 MongoDB，不需要解析 JSON 数据，直接存储进数据库里面，极大的方便了存储工作，而且 MongoDB 性能高效查询速度快，使用 JSON 风格进行存储易于理解和掌握。

上面是对本文用到的技术进行介绍，下面介绍具体的数据采集过程。

3. 数据采集思路及实现

3.1. 数据采集思路

好的数据采集思路，可以使目标明确，有条理，及时发现哪一步出现问题，有针对性的解决，本文设计如图 1 的数据采集思路流程图。

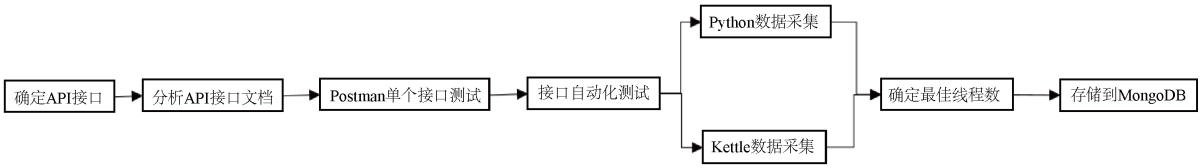

Figure 1. Data collection ideas
图 1. 数据采集思路

数据采集思路解释如下：

1) 确定 API 接口，根据需求确定采集的 API 接口，一般 API 接口都会有密钥或者发送方签名(appKey)，这些在确定采集 API 接口的时候需要获取到，否则无法访问 API 接口。

2) 分析 API 接口文档，API 接口文档会对 API 接口进行详细的介绍，包括请求参数、返回结果参数、服务状态和实例结果等，通过分析 API 接口文档，就可以对 API 接口有宏观的了解。

3) Postman 单个接口测试，通过单个接口测试可以测试 API 是否可用，查看返回结果，除了返回结果还会返回状态码以及是否成功的信息，返回结果可以选择 JSON、XML、HTML、TEXT 等格式，默认是选择 JSON 格式，JSON 格式简洁而且有着清晰的层次结构易于阅读。

4) 接口自动化测试，随机选择一批数据做接口自动化测试，并设定验证函数，以判断接口是否请求成功，导出自动化测试结果，在结果中可以看到接口访问成功数量以及每个接口的响应时间和接口的平均时间。

5) Python 数据采集和 Kettle 数据采集，这是两种数据采集的方式，通过两种方式进行比较，选择较好的方式进行数据采集工作。

6) 确定最佳线程，在数据采集时，不使用线程数据采集相对较慢，使用线程可以使数据采集速度成倍的增加，但速度不能一直增加，在保持数据采集稳定以及占用资源相对合理的情况下，确定最佳的线程可以极大的提高采集的速度。

7) 存储到 MongoDB，与其他数据库相比，比如 MySQL，采集下来的 JSON 数据存到 MySQL 数据库里面，需要对 JSON 数据进行解析，获取相应的值存到 MySQL 数据库里面，而存到 MongoDB 就不需要对 JSON 数据进行解析，可以直接进行存储，极大的提高了存储效率还保持了 JSON 数据的层次结构。

3.2. 数据采集实现

主要从三个方面介绍数据采集的实现过程，分别是自动化测试、Python 数据采集和 Kettle 数据采集，其中 Python 数据采集和 Kettle 数据采集是做对比实验。本实验数据选用百度天气查询 API 接口，也可以选用其他接口，比如政务数据接口，但不如百度 API 具有代表性。

3.2.1. 自动化测试实现

自动化测试实现的流程图如图 2，结合流程图详解 API 接口自动化测试实现流程图。

Figure 2. Flow chart of automated testing
图 2. 自动化测试实现流程图

1) 确定接口，这里选择的是百度国内天气查询服务 API 接口，接口地址如下：
http://api.map.baidu.com/weather/v1/?district_id=222405&data_type=all&ak=你的 ak，其中 district_id 是区县的行政区划编码，ak 是百度开发者密钥，单个接口测试的返回结果如图 3，政务接口返回的结果也类似于这样的 JSON 数据。

2) 随机选择测试数据，因为需要自动化测试多个接口所以需要随机的选择区县的行政区划编码，将接口地址的 district_id 改为随机变量，这时的接口变成这样
http://api.map.baidu.com/weather/v1/?district_id={{district_id}}&data_type=all&ak=你的 ak，这里随机选择了 102 条数据作为测试数据。

3) 验证返回结果，自动化测试不需要人工再去验证接口返回结果是否正确，在 Tests 里写一个 JS 判断，就可以自动判断返回结果是否正确。

4) 分析结果，图 4 和图 5 分别是自动化测试运行图和自动化测试数据结果图，在图 4 中可以看到每个接口的 URL 和返回结果，还有运行时间以及返回结果的大小，图 5 是将这些结果导出，导出的结果会有运行的平均时间，成功多少失败多少，通过图 5 可以看到每条接口的平均响应时间是 64 ms，成功了 102 条数据，也就是全部接口都成功了。

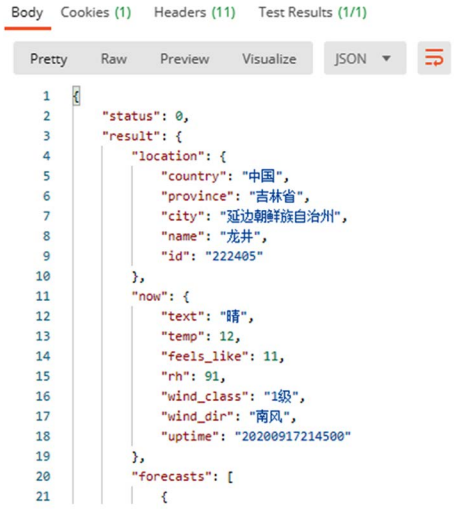

Figure 3. Return result of a single interface test
图 3. 单个接口测试的返回结果

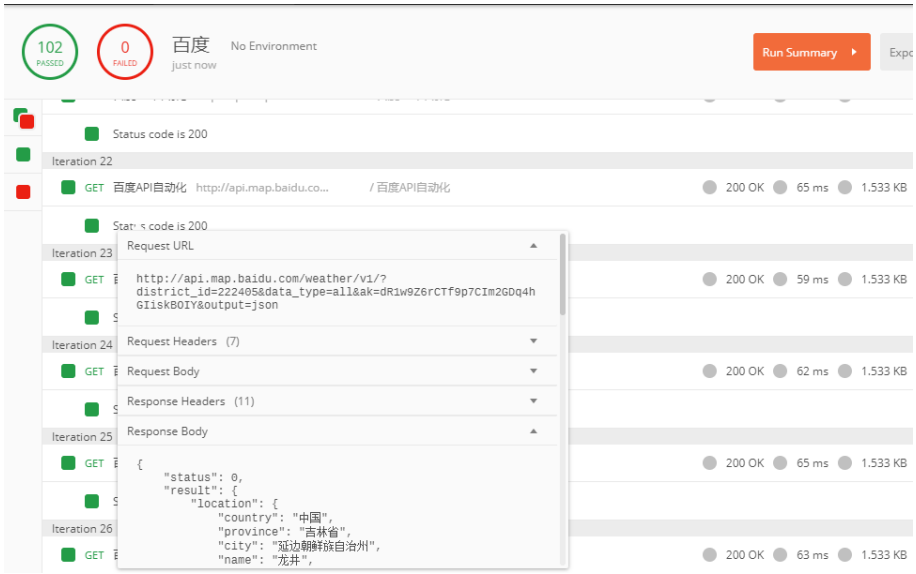

Figure 4. Automated test run
图 4. 自动化测试运行

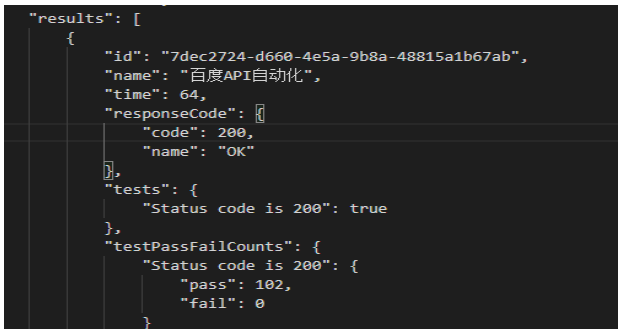

Figure 5. Automated test output results
图 5. 自动化测试输出结果

上述完成了 API 接口自动化测试，下面介绍两种具体的数据采集方式。

3.2.2. Python 数据采集实现

Python 有着丰富的内置库和外部库，内置库可以直接引用，外部库需要安装才能使用，安装方式也比较简单，在命令窗口输入 pip install 库名即可安装。本文主要用到的库有 requests、json、pymongo、csv、datetime、threading、multiprocessing，其中 requests 是一个 Python 的 HTTP 客户端库，用来发送请求与接收响应；json 是一种轻量级的数据交换格式，json 库将已编码的 json 字符串解码为 Python 对象；pymongo 是 MongoDB 数据库的驱动，用来连接 MongoDB 数据库；csv 是用来存储或者打开表格数据的，这里是用来打开 csv 文件；datetime 是一个时间库这里是用来计算数据采集用的时间；threading 是多线程库，用来给程序开多个线程；multiprocessing 是多进程管理包，它和 threading 一样都是为了加快采集的速度。

Python 数据采集的伪代码如下：

输入：API 接口 URL 和密钥

输出：接口返回结果，并存到 MongoDB，程序运行时间

1. 引入 Python 库，requests、json、pymongo、csv、datetime、threading 等；
2. 程序开始时间；
3. 引入线程和线程池；
4. 循环读取区县行政区划编码；
5. 采集 API 接口数据；
6. JSON 数据解析；
7 保存进 MongoDB 数据库；
8. 程序结束时间；
9. 程序运行时间。

采集到的数据存到 MongoDB 数据库的结果如图 6。

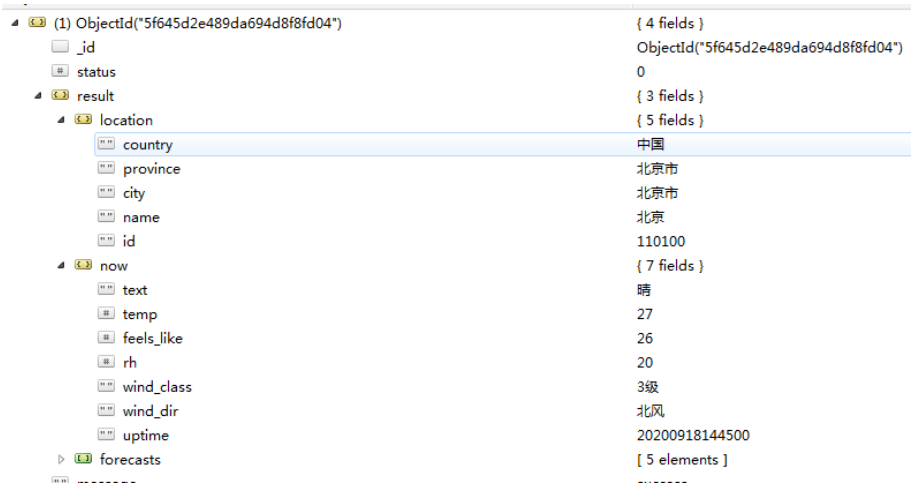

Figure 6. Mongo DB storage results
图 6. MongoDB 存储结果

3.2.3. Kettle 数据采集实现

Kettle 里面有着丰富的控件，用到哪一个可以直接拖拉出来，这里共选择了五个控件，分别是 CSV 文本输入、字段选择、JavaScript 代码、HTTP Client 和 MongoDB output 控件，CSV 文本输入控件是用来

读取 CSV 文本文件的，字段选择控件是用来选择需要的字段，JavaScript 代码控件是用来定义变量，HTTP client 是用来获取 API 接口 URL，MongoDB output 是将采集的数据存储到 MongoDB 数据库，数据采集过程流程图如图 7。

Figure 7. Kettle data acquisition process
图 7. Kettle 数据采集过程

数据采集过程解释：

1) CSV 文件输入，因为是以国内天气查询服务 API 接口为例，故 CSV 文件选择了国内城市行政区划代码，若是国家企业信息接口应以统一社会编码或者企业名称作为查询条件，广东省常住人口信息应以身份证号码作为查询条件。

2) 字段选择，因为上一步里面可能有很多字段，有些字段是不需要的，字段选择可以帮我们选择出需要的字段进入下一个流程。

3) JavaScript 代码是用来获取上一步的字段选择变量，并重新定义一个接口 URL 变量。

4) HTTP client 是用来获取上一步的 URL 变量，进而采集数据，这里使用了 10 个线程也就是一次可以采集十条数据，这个线程是可以修改的，在线程使用上 Kettle 比 Python 更灵活更容易实现。

5) MongoDB output 是将数据存到 MongoDB 数据库里面，类似于图 6，但这里没对 JSON 数据进行解析，而是选择了直接存储，如若需要解析引入 json 控件按照 jsonpath 解析即可。

图 8 是数据采集运行图，采集 3400 条数据用时 23.6 s。

#	步骤名称	复制的记录行数	读	写	输入	输出	更新	拒绝	错误	激活	时间	速度(条记录/秒)
1	CSV文件输入	0	0	3400	3401	0	0	0	0	已完成	0.0s	103,061
2	字段选择	0	3400	3400	0	0	0	0	0	已完成	0.0s	91.892
3	JavaScript代码	0	3400	3400	0	0	0	0	0	已完成	0.0s	82.927
4	HTTP client	0	340	340	0	0	0	0	0	已完成	22.6s	15
5	HTTP client	1	340	340	0	0	0	0	0	已完成	22.6s	15
6	HTTP client	2	340	340	0	0	0	0	0	已完成	22.5s	15
7	HTTP client	3	340	340	0	0	0	0	0	已完成	23.2s	15
8	HTTP client	4	340	340	0	0	0	0	0	已完成	22.4s	15
9	HTTP client	5	340	340	0	0	0	0	0	已完成	22.9s	15
10	HTTP client	6	340	340	0	0	0	0	0	已完成	23.2s	15
11	HTTP client	7	340	340	0	0	0	0	0	已完成	22.6s	15
12	HTTP client	8	340	340	0	0	0	0	0	已完成	22.9s	15
13	HTTP client	9	340	340	0	0	0	0	0	已完成	23.6s	14
14	MongoDB output	0	3400	3400	0	3400	0	0	0	已完成	23.6s	144

Figure 8. Data acquisition operation diagram
图 8. 数据采集运行图

4. 数据采集对比及结果分析

本文的数据采集环境为 Windows 7 旗舰版 64 位操作系统，处理器为 Intel(R)CoreTM i7-4790 CPU @ 3.60 GHz (8 CPUs)，~3.6 GHz，内存为 16 G，Python 版本为 3.7.0，Python 编译器为 PyCharm Community Edition 2018.2.4 ×64，Kettle 版本为 8.2，MongoDB 版本为 4.0.2，MongoDB 可视化工具为 Robo 3T 1.2.1。

以采集百度国内天气查询服务 API 接口为例，采集其他 API 接口原理同样，此接口共有 3400 条数

据，主要从 Python 无线程、Python 有线程、Kettle 不同线程等在数据采集时间上对比，以体现线程对数据采集速度提升的重要性并验证最优线程，数据采集时间如表 1，这里选择每组五次实验求平均值，以验证实验结果。

Table 1. Experimental data comparison table
表 1. 实验数据对比表

序号	实验名	第一次时间/s	第二次时间/s	第三次时间/s	第四次时间/s	第五次时间/s	平均时间/s
1	Python 无线程	349	336	312	340	336	334.6
2	Python 有线程	43	43	40	44	40	42
3	Kettle 无线程	202	197	203	205	204	202.2
4	Kettle 3 个线程	76	100	70	80	100	85.2
5	Kettle 5 个线程	43.1	40.3	44.4	47	47.1	44.38
6	Kettle 7 个线程	29.9	29.3	29.9	30.2	30.4	29.94
7	Kettle 8 个线程	27.8	25.1	24.9	24.9	27.9	26.12
8	Kettle 9 个线程	22.1	22.2	23.7	22.5	23.4	22.78
9	Kettle 10 个线程	20.1	20.1	20.5	20.2	21.5	20.48

为了更加直观的展示不同采集方式所用的平均采集时间，建了如图 9 的平均采集时间对比图。

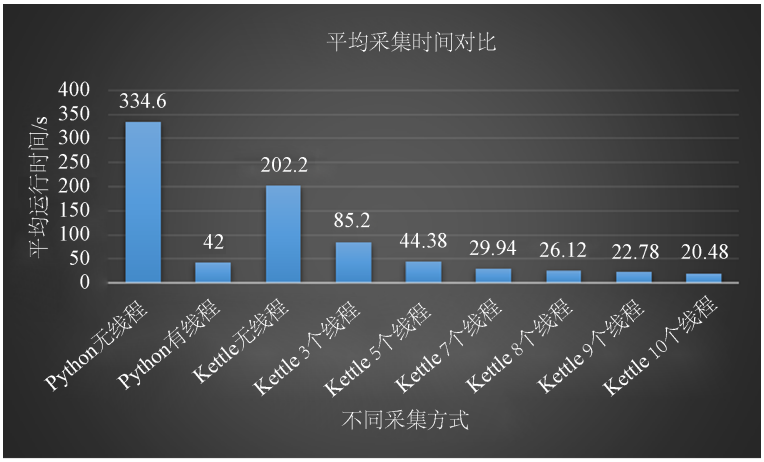

Figure 9. Comparison of average acquisition time
图 9. 平均采集时间对比

图中和表中的无线程并非是没有线程，而是没有人为的设置线程，不能并发的采集数据，可以看到两个无线程的采集方式所用时间都是相对较长的，与有线程的相比，最大相差接近 10 倍，相当于在数据采集时快了十倍。

Python 在设置线程时灵活度不够，因此引入了线程池，通过计算可以看到 Python 无线程和有线程在采集速度上差了 8 倍，而差的这 8 倍正好是计算机的 CPU 逻辑核心数，也是理论上最优线程数，最优线程数并不是指采集速度最快的线程数，而是综合各个方面最优的指数，这一点在 Kettle 数据采集上得到了很好的验证，随着线程数的增加 Kettle 的数据采集速度越来越快，在 7~10 个线程时采集速度逐渐趋于水平，而且在使用 10 个线程进行采集时，API 就开始发布预警通知，并发量已接近约定上限，理论上如

果没有并发量的限制，线程数是可以随意调节的，只不过会占用过多计算机资源，甚至导致程序运行失败。

数据采集时间在理论情况下可以通过以下公式计算：

$$T_{collections} = \frac{N_{collections} * T_{response}}{N_{thread}} \tag{2}$$

$T_{collections}$ 数据采集时间，$N_{collections}$ 数据采集数量，$T_{response}$ 接口响应时间，N_{thread} 线程数量，其中 $T_{response}$ 接口响应时间是可以通过自动化测试计算出来的，N_{thread} 在这里不是指最佳线程数量，而是指实际采集中的线程数量。此公式在 5 个线程以上准确率达 90%以上，在 7~8 个线程准确率达 97%，在 9~10 个线程准确率可达 99%，如表 2：

Table 2. Formula prediction accuracy
表 2. 公式预测准确性

序号	线程数	实际采集时间/s	预测时间/s	准确性
1	Kettle5 个线程	44.38	40.800	91.9%
2	Kettle7 个线程	29.94	29.143	97.3%
3	Kettle8 个线程	26.12	25.500	97.6%
4	Kettle9 个线程	22.78	22.667	99.5%
5	Kettle10 个线程	20.48	20.400	99.6%

这是因为一个线程在数据采集数量已经稳定，一个线程在实验中每秒平均采集 17 条数据，每条约 58 ms，这个采集速度已经接近于接口的响应速度，上述公式也可以改成如下：

$$T_{collections} = \frac{N_{collections}}{N_{per-thread} * N_{thread}} \tag{3}$$

$T_{collections}$ 数据采集时间，$N_{collections}$ 数据采集数量，$N_{per-thread}$ 单个线程采集数量，N_{thread} 线程数量。这两个公式在计算采集时间上误差不是很大，一个适用于知道接口响应时间，一个适用于知道单个线程采集速度。

5. 结束语

数据的开放共享有利于打破信息孤岛，让数据真正地为人民服务，目前各地都在建立“数字政府”，目的是使政务数据互联互通互享，真正地做到“一网通办”和企业群众办事“只进一扇门”“最多跑一次”。在目前的数据开放共享中，API 接口是数据开放的主流方式，如何快速高效地从 API 接口采集数据是本文研究的重点。本文融合自动化测试、最优线程、Python 以及 ETL 技术等构建了多种技术融合的 API 接口采集策略，该策略表明在有无线程的时候都比 Python 数据采集高效，而且线程可调节，实现简单，并推导出数据采集时间公式，该公式在 5 个线程以上准确率达 90%以上，在 7~8 个线程准确率达 97%，在 9~10 个线程准确率可达 99%，在采集之前就可通过该公式以最合理的线程计算出最合理的采集时间，极大地节省采集时间。

但目前的 API 接口良莠不齐，规范性不够，使得在数据采集时经常性采集失败，应重点加强 API 接口的管理以及开放接口 API 的安全性，这样才能使开放共享的数据真正为人们服务，而不是为了共享而共享，同时在数据开放共享的时候，应确保数据的质量，开放的数据应经过清洗后才共享的，“脏”数

据共享出来，采集时还得进行复杂的数据清洗过程，这违反了数据开放共享的初衷。

基金项目

国家自然科学基金资助项目(No. 11801081)；广东省自然科学基金资助项目(No. 2018A030313063)；广东省教育厅重大项目(No. 2014KZDXM055)；广东省大数据分析与处理重点实验室开放基金资助项目(No. 2017018)；研究生教育创新计划项目(No. 2016SFKC_42，YJS-PYJD-17-03)；教育部“云数融合科教创新”基金项目(No. 2017B02101)；江门市基础与理论科学研究类科技计划项目(No. 2019030101870008915)；五邑大学教学质量与教学改革项目(JX2020054)。

参考文献

[1] 王震, 周颖, 黄赪东, 等. 面向大数据应用的区块链解决方案综述[J]. 计算机科学, 2019, 46(S1): 6-10.

[2] 李迅雷, 徐驰. 以“新基建”推进国家治理现代化[J]. 人民论坛·学术前沿, 2020(10): 70-74.

[3] 谢裕清, 王渊, 江樱, 等. 便于数据共享的电网数据湖隐私保护方法[J/OL]. 计算机工程与应用: 1-9. http://kns.cnki.net/kcms/detail/11.2127.TP.20200429.0920.004.html, 2020-06-25.

[4] Reinsel, D., Gantz, J. and Rydning, J. IDC-全球白皮书《世界的数字化从边缘到核心》[EB/OL]. 2018-11-08. https://www.doc88.com/p-7876468738713.html, 2020-9-14.

[5] 赵颖, 侯俊杰, 于成龙, 徐皓, 张伟. 面向生产管控的工业大数据研究及应用[J]. 计算机科学, 2019, 46(S1): 45-51.

[6] 李正, 吴敬征, 李明树. API 使用的关键问题研究[J]. 软件学报, 2018, 29(6): 1716-1738.

[7] 妮可的平凡时光. 如何使用 API 爬取数据, 它和网页爬虫有什么区别? [EB/OL]. https://blog.csdn.net/weixin_43944997/article/details/105502469, 2020-9-14.

[8] 王科特, 王力生. 信号实时采集系统的最佳并行线程数的研究[J]. 计算机应用, 2011, 31(10): 2593-2596.

[9] 朱晓姝, 许桂秋. 大数据预处理技术[M]. 人民邮电出版社: 北京, 2019: 14-41.

[10] 王华志. 基于 JSON 的异构数据源数据交换技术研究[D]: [硕士学位论文]. 武汉: 武汉理工大学, 2015.

[11] 忧伤的比目鱼. XML 与 JSON 比较[EB/OL]. https://blog.csdn.net/luoyoub/article/details/80290145, 2020-9-14.

[12] 高升. 基于 JSON 的数据库访问层研究与应用[D]: [硕士学位论文]. 北京: 北方工业大学, 2019.

[13] 张明, 程宝雷, 查伟忠, 等. 面向安卓手机 App 功能测试技术的方法[J]. 计算机工程与设计, 2018, 39(3): 684-689.

[14] 谢业欣. 一个基于数据共享的接口开发平台[J]. 软件, 2020, 41(8): 152-157.

[15] Goetz, B., Peierls, T., Bloch, J., *et al.* (2006) Java Concurrency in Practice. Addison-Wesley Educational Publishers Inc., New York, 182-190.

[16] 杜晓旭, 贾小云. 基于 Python 的新浪微博爬虫分析[J]. 软件, 2019, 40(4): 182-185.

[17] 刘鹏, 张燕. 数据清洗[M]. 北京: 清华大学出版社, 2018: 199-222.

[18] 郭丹, 樊红. 基于 ETL-KETTLE 的贵州卷烟营销大数据分析及可视化[J]. 计算机系统应用, 2017, 26(1): 74-80.