基于大数据微服务框架的网站数据统计分析与可视化

王晶 王科

四川大学锦城学院 计算机与软件学院 四川 成都 611731

【摘 要】：随着互联网时代的发展，人们的生活都跟网络息息相关，网站产生的用户数据量越来越大，传统分析方法已经不能满足运营需求，这时就需要引进大数据相关的处理分析技术来对数据进行实时或离线的统计与分析。本文使用了大数据实时分析处理框架来对网站实时产生的数据进行统计分析并通过前端页面可视化展现分析成果。

【关键词】：大数据；微服务；网站数据；实时分析

在这个网络高速发展的时代，大批量数据的产生已经屡见不鲜，一些互联网科技大型企业一天生成的数据量甚至可以达到 EB 级别，这种规模的数据处理的工作量早已超过传统的人工极限。大数据应运而生，市面上出现了各种各样开源的大数据分析框架，也迎来了大数据时代的飞速发展。如何正确高效地使用大数据框架来进行数据处理与分析，成为了大数据时代所面临的最基本也是最重要的问题。

本文通过使用大数据微服务框架来实现对爬取数据地实时处理并对数据进行可视化分析。

1 项目架构及项目流程介绍

1.1 项目架构

本项目使用了 Python 进行数据地实时爬取，数据传输及处理框架使用了 Flume-Kafka-Spark streaming 确保了数据的实时处理性，Spring Boot-Dubbo-Zookeeper 作为后端服务器框架，用来保障数据地正确展示以及数据的存储入库，存储用的数据库使用了 MySQL，缓存数据库选择了 Redis，整个项目运行在 Linux 系统中。

其中 Flume 是一种信息采集、筛选过滤、传输的框架；Kafka 是一种基于分布式架构的消息队列；Spark Streaming 是 Spark 框架所提供的一个实时流处理数据的模块,Spring Boot 是一个用来简化以前 ssm 框架的一套全新框架，Dubbo 是一套 RPC 通信框架，Zookeeper 作为 Dubbo 的注册中心，便于管理各个服务提供者和消费者，从而实现微服务架构，MySQL 是现如今比较热门的关系型数据库之一，Redis 则是 NoSQL 中的重要成员。

1.2 项目流程

整套流程的系统流程图如图 1 所示：

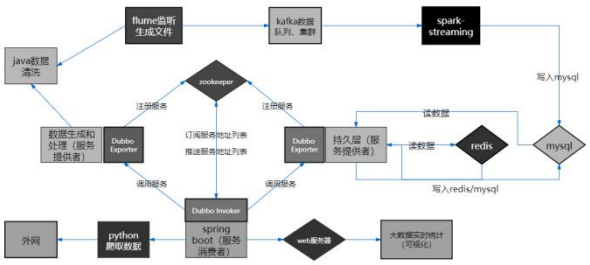

图 1 系统流程图

Fig.1 System flow chart

通过 Python 编写爬虫代码，实时爬取网站数据并在后台生成文件目录，Flume 通过监听该文件目录的文件更新实时将更新的数据发送至 Kafka Sink 对应的 Kafka，再由 Kafka 作为对接 Spark Streaming 的中间件，完成数据的第一次清洗，并将清洗后的数据发送给 Spark Streaming 进行实时的数据清洗并对清洗完成的数据进行统计分析再将统计结果暂存 Redis,最后交由 Redis 写入 Mysql。与此同时后台服务器（Spring Boot + Dubbo + Zookeeper）会通过 ajax 请求实现对数据库的监听，将新入库的统计数据通过请求返回装填至 Echarts 图表用来给前端进行展示，当后台 Spark 程序计算完成时会反馈给 ajax 一个信号用来更新前端进度条，给予用户更好的体验。

其中区别于传统大数据开发框架而言，微服务架构是一套很适合团队开发的架构，它将一个应用程序划分成很多个小的模块，每个模块之间通过轻量级的通信协议来进行沟通交流并达成协作，例如 Dubbo 的 RPC 通信，这样做的好处就是可以降低各个模块之间的耦合度并且拥有非常强大的可扩展性，例如你的程序需要添加一项新的功能，传统架构可能对程序的改动会非常的大，但是在微服务的架构里面，你只需要增加一个单一服务将这个服务注册到注册中心就可以很轻易的实现添加功能。我这里使用的是 Spring Boot + Dubbo + Zookeeper 搭建的简易微服务架构，读者若对微服务架构有兴趣的话我比较推荐 Spring Cloud 所搭建的微服务架

构。

使用 Dubbo 构建的微服务架构就像组装电脑，各环节我们的选择自由度很高，但是最终结果很有可能因为一条内存质量不行就点不亮了，总是让人不怎么放心；而 Spring Cloud 就像品牌机，在 Spring Source 的整合下，做了大量的兼容性测试，保证了机器拥有更高的稳定性，但是如果要在使用非原装组件外的东西，就需要对其基础有足够的了解[1]。

2 数据爬取与生成

2.1 数据的爬取

在 PyCharm 上使用 python 爬取数据。使用的爬取框架是 Python 自带的 BeautifulSoup。爬取数据使用的 ip 主要使用了本机 ip 爬取数据、网站（xici）的免费 ip 爬取数据、收费的 ip 代理池爬取数据。本机的 ip 爬取数据适用于测试，稳定性高，但 ip 被封得不偿失。收费的 ip 代理池爬取数据适用于爬取大批量的数据。免费 ip 稳定性极差，10 条 ip 可能只有 1 条能用，不推荐使用。

2.2 数据的生成

由于设备及网络原因，单纯的爬取数据速度过慢，且访问 ip 及其容易被屏蔽，为了体现大数据框架的分析能力以及更加贴近于真实场景，我选择了使用在 IDEA 上搭建 spring boot 框架生成数据。前端页面由用户自定义爬取规则以及爬取数据条数，后端通过获取前端的规则并按此规则生成模拟真实数据。在数据生成的位置调用 Python 数据爬取的文件，实现数据生成和数据爬取同时执行。与此同时，后端将进度发送给前端，前端就能通过进度条观察实时进度。其中，数据爬取及生成规则的用户自定义页面如图 2 所示：

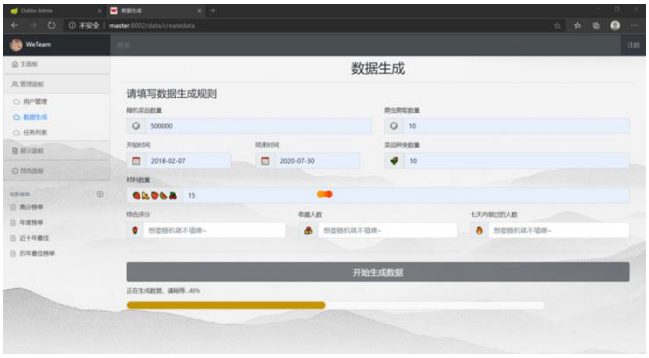

图 2 用户自定义生成和爬取规则页面展示图

Fig.2 User custom generated and crawled rule page display diagram

3 数据清洗与数据分析

3.1 数据清洗

数据清洗(Data cleaning)是对数据进行重新审查和校验的过程，目的在于删除重复信息、纠正存在的错误，并提供数据一致性[2]。

数据清洗在本项目中一共有两个环节。第一个清洗环节是 Kafka 收到 Flume 传递过来的数据之后通过重写 Kafka 的消费者并对数据进行二次清洗，将清洗完成的数据放入另一个 Topic 供 Spark Streaming 来消费。第二个环节是由 Spark Streaming 消费 Kafka 初步清洗的数据并对其进行第二次清洗。：

Flume 监听文件使用了 TailDir Source，可以用来同时监控生成文件目录下的多个数据文件，这里的多个文件指的就是生成数据的文件以及爬取数据的文件，Sink 使用的是 Kafka Sink，一旦这些文件内数据有更新 Flume 就会将其传递给 Kafka 对应的 Topic，并交由 Kafka 执行下一步清洗操作。

3.2 数据分析

数据分析采用的是将编写好的 Spark Streaming 程序提交到 Linux 上的 Spark 集群上运行，将实时分析结果存入到数据库中，其中 Spark Streaming 使用了 map 算子来对数据进行拆分成数组，再用 flatMap 将数据扁平化后使用 filter 去除空格等空白字符，再用 map 生成新的流数据（键值对形式），后再使用将其根据其键分类将对应键的值进行累加得到最终结果再用 map 生成新的流数据（配合上 uuid 用于区分每次任务的结果），处理数据的整体思想就是先将数据进行拆分然后对你想要处理的数据进行对应操作再将得到的结果重组成新的流数据进行写文件或者写数据库操作；Spark Streaming 的数据入库操作需要建立一个 MySQL 的连接池。

数据库连接池是负责分配、管理和释放数据库连接，它允许应用程序重复使用一个现有的数据库连接，而不是再重新建立一个连接[3]。

4 与用户的交互——前端的实现

4.1 前端登录页面及其他的一些功能页面的实现

为了本程序有一个更加清晰和直观的界面方便用户使用以及帮助用户对数据的分析结果有一个更加深刻的认知，一个好的前端界面用来与用户的交互是必不可少的。

登陆页面的业务逻辑使用了 java 的安全框架 shiro，通过配置框架的一些接口来实现跟数据库中用户名密码的匹配校验从而判断是否登录成功，值得一提的是，安全框架的使

用可以让位于数据库中的密码不再是明文密码，而是一串经过加密之后的字符，我在这里使用了一个简单的 MD5 加密。具体代码及加盐加哈希算法如下：

先是加盐的代码：

```
public static String getSalt(int n) {
        StringBuffer stringBuffer = new StringBuffer();
        for (int i = 0; i < n; i++) {
            char aChar = chars[new
Random().nextInt(chars.length)];
            stringBuffer.append(aChar);
        }
        return stringBuffer.toString();
    }
```

通过对函数传参 n 来限定盐的长度，再由循环来随机生成一串盐追加到可变字符串 stringBuffer 中并将其作为函数的返回值返回给 Md5Hash 加密算法。

以下代码是实现 Md5 加密的过程：

```
public void addUser(User user) {
        //生成随机盐
        String salt = SaltUtils.getSalt(8);
        user.setSalt(salt);
        Md5Hash md5Hash = new
Md5Hash(user.getPassword(), salt, 1024);
        user.setPassword(md5Hash.toHex());
    }
```

最后将加密后的用户信息写入数据库，并将原用户信息从 Redis 的缓存中删除。

其中哈希加盐中的盐值是一组随机的字符串，通过插入在口令后进行 HASH 算法，这样即使是相同的口令，插入不同的盐值后生成的 HASH 值也是不相同的[4]。

为了更好的与用户交互，我还增加了一个历史任务的列表，方便历史统计数据的存档以及多次任务的比较分析。实现效果如图 3 所示：

图 3 效果图

Fig.3 Rendering

4.2 核心功能——统计与分析结果展示页面

这一部分是本文核心的功能以及重要成果展示，后端处理完成后的数据结果会被 ajax 发送异步请求从数据库中拿取出来装填到前端的图表插件 Echarts 中，方便用户及管理员能对数据有一个清晰直观的认证，而不是局限于一堆数字表格，这也是大数据时代非常重要的一个模块——数据可视化。最终效果图如图 4 所示：

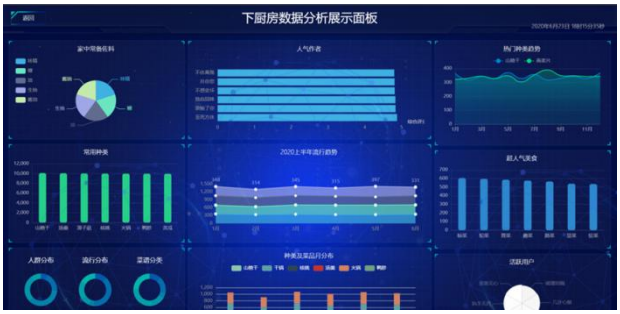

图 4 最终效果图

Fig.4 Final rendering

这是一个动态的数据可视化页面，他会随着数据库数据的更新不断局部更新页面的可视化图表，是一个完全实时的可视化。这部分功能的实现我主要是依赖于 ajax 以及前端的定时器，间隔请求数据库中的新数据，装填到 Redis 中做好缓存，然后再从 Redis 中拿取数据更新前端的数据再把新数据局部刷新页面，就可以达到此动态效果。

5 总结

随着大数据时代的来临，人们的生活越来越离不开数据化，但是随着数据产生越来越多，数据量越来越大，单纯的数字数据以及传统的数据分析手段已经不能满足社会需求，必须通过相应的大数据分析技术，大数据可视化技术，将数据交由计算机处理并将最终结果反馈成图表的形式来方便管理人员以及用户浏览，在优化用户体验的同时又能清晰准确的表达数据所代表的含义以及通过这些数据的规律对未来的趋势做一些预测。

本文所实现的项目充分结合了大数据分析技术和大数据可视化技术，在大批量准确的处理数据的同时给予用户更加准确、更加迅速的友好反馈。

参考文献：

[1] https://blog.csdn.net/Jonny_jun_gao/article/details/91049778

[2] https://www.zhihu.com/question/375452162/answer/1047257354

[3] https://www.cnblogs.com/cocoxu1992/p/11031908.html

[4] https://blog.csdn.net/acm_yuuji/article/details/27642221