网络爬虫设计文档

2018.4.11

王焕锋

# 一、网络爬虫概述

网络爬虫是把网页信息自动爬取下来，并进行解析、过滤、整理后，保存到本地或数据库。

# 二、整体技术框架

**网络爬虫设计到多线程、http请求、队列、url去重、爬虫策略、数据库等技术。**

## 1、多线程

运用多线程技术可以并发爬取网页信息，大大提高爬取的速度。建立线程池可以减少创建线程和销毁线程的次数，线程池中的线程可被重复利用，执行多个任务。

## 2、http请求

http请求能够下载网页信息。服务器会将大规模的不加保护的爬取请求识别成爬虫程序从而对其拦截，所以爬取的时候有必要设置http请求头，例如user agent、cookie、referer等，这样可以模拟真实的浏览器请求，减小被反爬的几率。

## 3、队列

爬虫程序一般有种子url队列、待爬取url队列、已爬取url队列，种子url队列是程序的入口点，待爬取url是爬取过程中产生的列表url、内容url，爬取过后的url对存到已爬取url队列中，避免重复爬取同一url。

## 4、url去重

url去重可以避免重复爬取同一url，造成内存的浪费。最简单的方法就是给url地址指定唯一key，对于get方法请求的地址，因为url地址是唯一的，所以直接将url作为key，对于post请求的地址，由于url地址不唯一，在指定key的时候可以把请求参数追加到url后面形成一个唯一的key。

## 5、爬虫策略

爬虫策略包括广度优先搜索和深度优先搜索，目前主流的爬虫一般采用广度优先搜索的策略。广度优先搜索策略是指在爬取过程中，在完成当前层级的搜索后，才进行下一层级的爬取，可以通过设置待爬取url的层级来实现。

## 6、数据库

在大型爬虫应用中，爬虫过程中使用的url队列、缓存信息通常很庞大，直接存到服务器的内存中会导致服务器宕机，这时候利用redis、mongodb等内存数据库可以缓存信息，还可以结合同步机制实现分布式爬取。

下载的网页信息经过必要的处理后，通常需要持久化到数据库中。对于数据量不大的爬虫，mysql、sql server可以满足业务的需求，如果数据量特别巨大，可以应用hbase等大型的分布式数据库。

# 三、爬虫架构图

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

种子url队列 已爬取url队列

下载网页

解析出来的url

|  |
| --- |
|  |
|  |
|  |

待爬取url队列 已下载网页库

# 四、爬虫流程图

否

数据持久化

将URL加入到待爬取URL队列

是否重复URLL

解析网页，获取URLs

下载网页

创建爬虫线程

获取待爬取队列的URL

选取种子URLs

将种子URLs加入待爬取URL队列

# 五、爬虫性能

不同的爬虫框架，应用到的技术会存在一定的差异，在不同业务环境下性能的差别会比较大。一些爬虫框架可能比较适合爬取数据量小的需求，在爬取大量数据的时候可能因为庞大的待爬取队列等因素导致程序变得异常缓慢；一些爬虫框架适合爬取大数据量的需求，爬取过程稳定且高效，但是在爬取小数据量时，程序启动时需要启动相应的数据库、支持软件，爬取过程中需要处理爬虫队列，可能程序花在维护的时间都比下载时间大，这时程序的性能就表现得很差了。

所以，需要结合实际的业务需求，选择合适的爬虫框架或者搭建一套自己的爬虫框架，才能使爬虫程序的性能表现得更优。

下面列出java爬虫框架webcolletor的一些性能数据。

本地测试：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| URL数量 | 1 | 100 | 10,000 | 1,000,000 |
| 爬取总时间（秒） | 3 | 11 | 8分9秒 | 13时53分8秒 |

阿里云服务器测试：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| URL数量 | 1 | 100 | 10,000 | 1,000,000 |
| 爬取总时间 | 3秒 | 5秒 | 50秒 | 29分26秒 |

上面测试的差距主要是由服务器内存、带宽等硬件引起的，所以除了爬虫框架本身的性能，硬件也是影响爬虫程序性能的一大因素。