Kaito

爬虫代理服务

2015-11-02 | 5026

> 由于最近一直在做爬虫相关的事情，那肯定少不了跟 代理IP 打交道，这篇文章用来记录如何实现爬虫代理服务，主要以讲解思路为主。

起因

做过爬虫的人应该都知道，抓的网站和数据多了，如果爬虫抓取速度过快，免不了触发网站的防爬机制，几乎用的同一招就是**封IP**。解决方案有2个：

- 同一IP，放慢速度(爬取速度慢)
- 使用代理IP访问(推荐)

第一种方案牺牲的就是时间和速度，来换取数据，但是一般情况下我们的时间是很宝贵的，理想情况下是用最短的时间获取最多的数据。所以第二种方案是推荐的，那么从哪里能找到这么多代理IP呢？

寻找代理

程序猿不懂的时候就去寻找嘛，google、度娘，输入关键字： 免费代理IP ，前几页几乎都是提供代理IP的网站，一一打开后观察发现，几乎都是一个列表页，展示少则几十、多至几百个IP。

但是仔细观察你就会发现，每个网站提供的免费IP是有限的，拿来用几个就会发现，有的也已经失效了。当然，他们更倾向于你购买人家的代理，人家就靠这个赚钱嘛。

身为狡猾的程序猿，当然不能因为这点困难就跪了，仔细想一下，既然搜索引擎能搜到这么多提供代理的网站，**每个网站提供几十或几百个，假如有10家网站，那加在一起也有几百到几千个IP。**

那么好了，你要做的事情就是，把这些网站记录下来，用程序把IP抓过来就好了，想想是不是很简单？

测试代理

通过刚才的方式，应该可以获得几百或上千的代理IP了。

等等，这么多IP，难道别人真的就免费送给你了么？当然不是，前面也提到过，这些代理中，有很大一部分已经是失效的了。那么怎么办？如何知道哪些代理是有效，哪些是不可用的呢？

很简单，挂上这些代理，访问某一个稳定的网站，然后看是否能正常访问，可以正常访问的就是可用的，不能访问的不就是无效的嘛。

最快速的，用 `curl` 命令就可以测试某个代理是否可用：

```
# 使用代理 48.139.133.93:3128 访问 网易首页
curl -x "48.139.133.93:3128" "http://www.163.com"
```

当然，这种方式只是为了演示方便，实际最好的方式是：
用多线程方式，使用代理去访问某个网站，然后输出可用的代理。
这样做能最快速的找出可用代理。

使用代理

现在已经可以通过上面的方式，找出可用的代理了，如果应用到程序中，应该不用我多说，大部分都应该会用了。
例如，刚才把可用的代理输入到某个文件中，每一行是一个代理，那么就可以这样使用：

- 读取代理文件
- 随机选择代理IP，发起HTTP请求

这样，如果代理有几百个，基本上可以保持一段时间抓取某个网站的数据了，抓个几千几万条数据不成问题。

但是，如果我想**持续不断**的从某个网站获取数据，或者是抓取上百万甚至上亿的网页数据，那这样肯定是不行的。

持续不断供应代理

刚才的方式是，一次性抓取某几个代理网站，然后通过程序测试每个代理是否可用，得到可用的代理列表。但是这只是一次性的，而且代理量往往很少，在持续抓取中肯定无法满足需要。那么怎么能持续不断的找到可用代理呢？

- 找到更多的代理网站（数据基础）
- 定时监控这些代理网站，获取代理
- 拿到代理IP后，程序自动检测，输出可用代理（文件或数据库）

- 程序加载文件或数据库，随机选取代理IP发起HTTP请求

按照上面的方式，可以写出一个自动采集代理的程序，然后爬虫端就可以定时去文件/数据库中获取然后使用就可以了。但是有一个小问题，怎样知道每个代理的质量如何？也就是说，代理的速度怎么样？

- 在检测代理时，记录请求响应时间
- 响应时间从短到长，加权重值，响应短的使用率高一些
- 限制某段时间内最大使用次数

前面几点只是基础，这3点可以进一步优化你的代理程序，输出有优先级的代理列表，爬虫端根据权重和最大使用次数使用代理。这样做的好处：**保证使用高质量代理，同时防止某一代理频繁使用防止被封。**

服务化

上面经过一系列的完善和优化，已经搭建好了一个可用的代理服务，只不过是基于文件系统或数据库的。

爬虫端要想使用这些代理，只能是读取文件或读取数据库，然后根据某种规则选择代理使用，这样做比较繁琐，能不能让爬虫端使用代理变得简单一些？那么就需要把代理访问做成服务化。

有个大名鼎鼎的服务器软件 `squid` ，利用它的 `cache_peer` 邻居代理机制，就可以帮这个事情做的很完美。

把代理列表的代理，按照 `squid` 的 `cache_peer` 机制按照一定格式，写在配置文件中即可。

`squid` 是个代理服务器软件，一般情况下是这样使用的，假如爬虫在机器A， `squid` 安装在机器B，需要爬取的网站服务器是机器C，代理IP是机器D/E/F...

- 不使用代理：爬虫机器 `A` 请求 -> 网站机器 `C`
- 使用代理：爬虫机器 `A` -> 代理IP机器 `D/E/F/...` -> 网站机器 `C`
- 使用 `squid` ：爬虫机器 `A` -> `squid` (机器 `B` ， `cache_peer` 机制管理调度代理 `D/E/F`) -> 网站机器 `C`

这样做的好处就是：**爬虫端不用考虑如何加载和选择可用代理，给出一个代理列表给 `squid` ，按照配置文件的规则，它就可以帮你管理和调度选择代理。最重要的是，爬虫端使用代理只需访问 `squid` 的服务端口就可以了！**

进一步整合

现在服务化也搭建完成了，唯一差得一步就是整合：

- 定时监控代理源网站(30分/1小时都可)，解析出所有代理IP，入数据库

- 从数据库中取出所有代理，访问某个固定的网站，找出访问成功的代理，更新数据库可用标记和响应时间
- 从数据库中加载所有可用代理，通过某种算法，根据响应时间计算使用权重和最大使用次数
- 按照 `squid` 的 `cache_peer` 格式，写入配置文件
- 重新加载 `squid` 配置文件，刷新 `squid` 下的代理列表
- 爬虫指定 `squid` 的服务IP和端口，进行纯粹的爬取操作

一个完整的代理服务通过这样的方法就可以搭建完成，定时输出高质量代理。爬虫端不用关心代理的采集和测试，只管使用 `squid` 的统一服务入口爬取数据即可。

> 以上只是提供一个代理服务完成的思路，有此需求的同学可根据此思路写出一个完整的代理采集程序。有时间会写一下具体的实现。

如果此文章能给您带来小小的工作效率提升，不妨小额赞助我一下，以鼓励我写出更好的文章！

微信打赏

支付宝打赏

代理服务

‹ 科技与人文

iphone6s使用感受 ›