爬虫：github 教程

写在前面——爬虫前一定要遵循网站的 robots 协议

- **抓包——（获得服务器数据包进行分析）**

Python：Urlib、Request 包模拟浏览器（包装请求头、定义请求参数）访问服务器并获得数据包。

Websocket 协议爬取动态消息如直播、弹幕等

Fiddler 软件抓包——能进行截取并修改后返回，手机网络设置与 PC 相同的代理后便能抓取手机的包

- **分析数据——正则表达式、html 解析**
- **正则表达式——字符串匹配**

通过 Python 的 re 模块的**正则表达式**对数据（网页代码 html）进行分析提取相关数据；

Beautifulsoup 模块解析 html 并直接返回标签内容，方便。

- **PC 浏览器自动化模拟——selenium、phantomjs**

Selenium 编写脚本使浏览器自动化操作——自动化爬虫、模拟登录、验证码识别、滑动验证码识别等等。

Phantmojs 隐式操作

- **提高爬虫速度**

多线程（资源共享，考虑上锁）、多进程（资源不共享）

使用线程池、进程池的方式避免重复的线程、进程创建、回收问题

- **IP 池**

爬取免费 IP 代理网站构建 IP 池。使用 IP 代理爬取网站防止被封 IP

- **手机模拟——Appium**

Python 编写脚本，通过 appium 实现手机的自动化操作。如微信等服务器返回的包数据难以解析，故可以直接爬取手机上经解析后的数据。

基于安卓开发环境

- **数据存储：CSV、Mysql（数据库）、json**
- **数据可视化：matplotlib、seaborn、pyecharts（各种绘图工具）**
- **实战源代码**

不基于 Scrapy 框架的多线程爬虫

基于 Scrapy+Mogodb 数据库的爬虫

- **爬虫框架：scrapy**

Scrapy 框架即封装好的用于爬虫的类

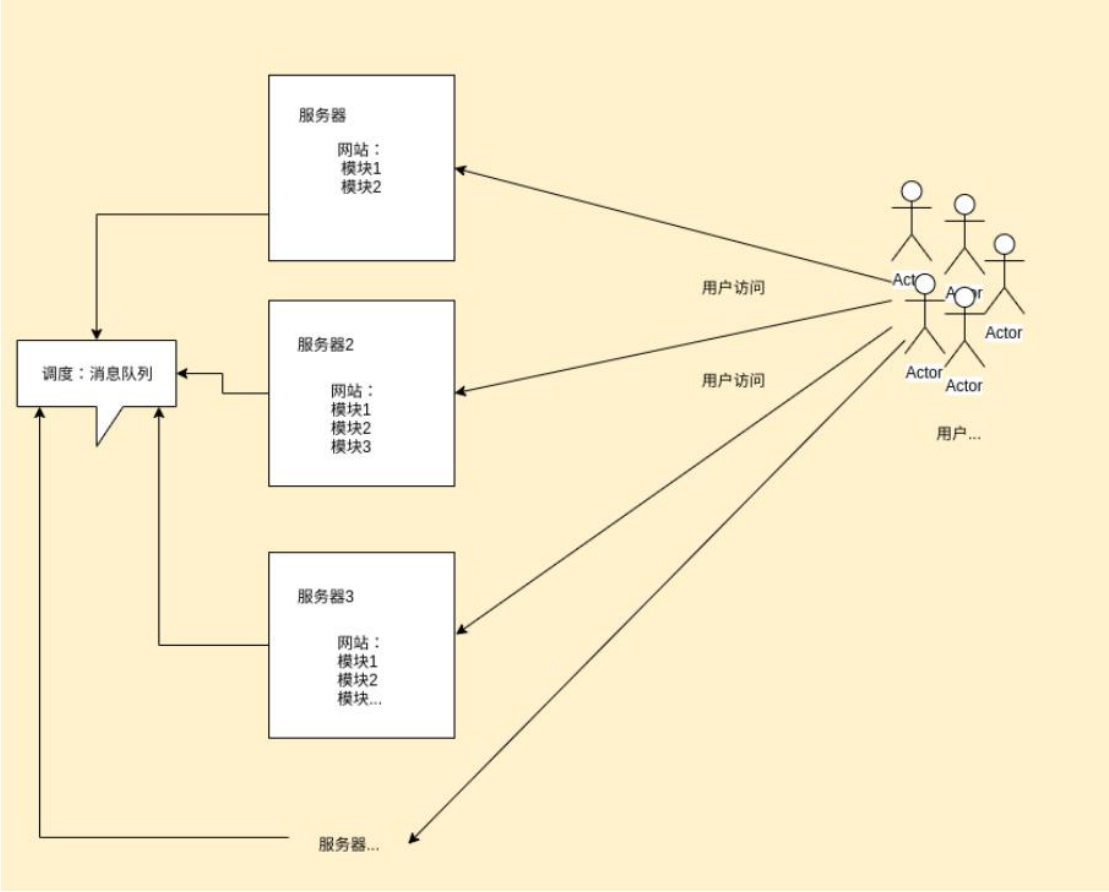
- **实战——糗事百科实战——爬虫+数据库 MongoDB**
- **可用 python 编写脚本的实现类似于 fiddler 抓包的软件——mitmproxy**
- **Chrome 爬虫插件**
- **爬虫技巧**
- **分布式**

定义：将功能划分为不同模块并运行在不同服务器上，服务器间消息队列需要进行消调度以保证运行有条不紊。即多个人同时干，分工

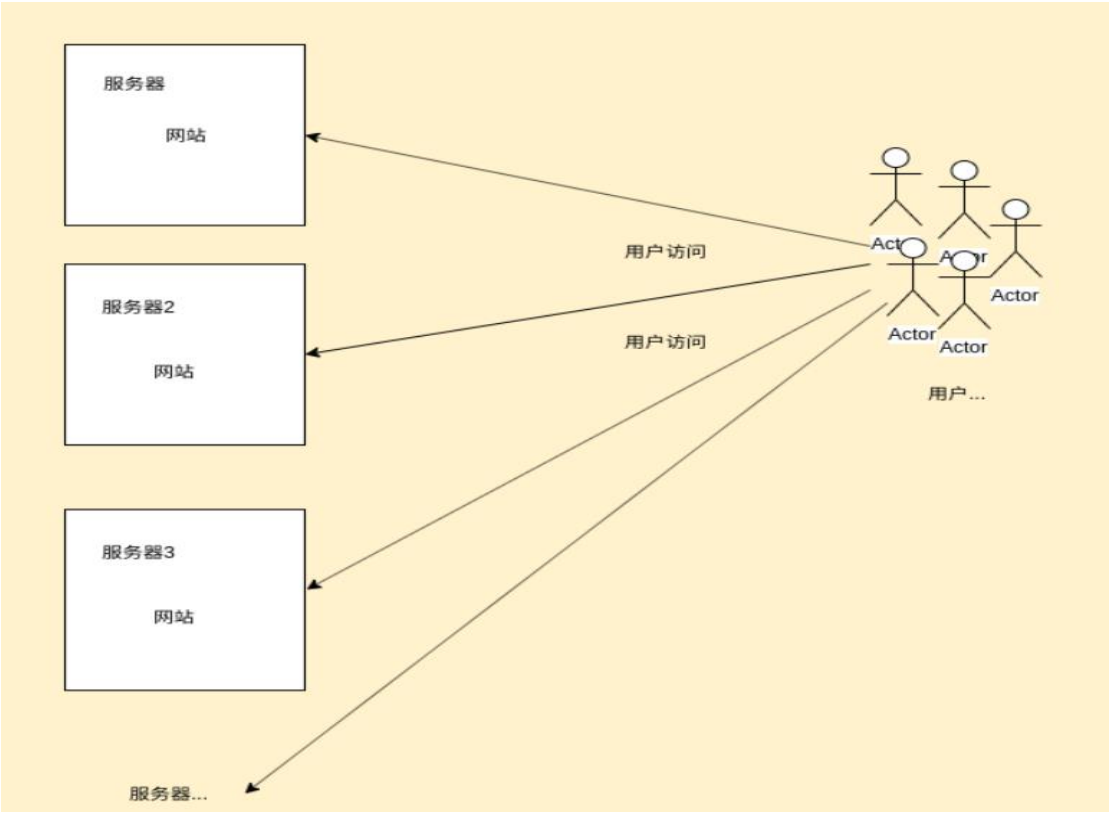
集群：多个人同时干，但每个人干同样的活

分布式爬虫：若干台服务器进行爬虫，爬虫服务器通过部署在一台服务器上的 redis 进行请求消息队列调度，并连接到部署在一台服务器上的数据库将爬取数据存入数据库。

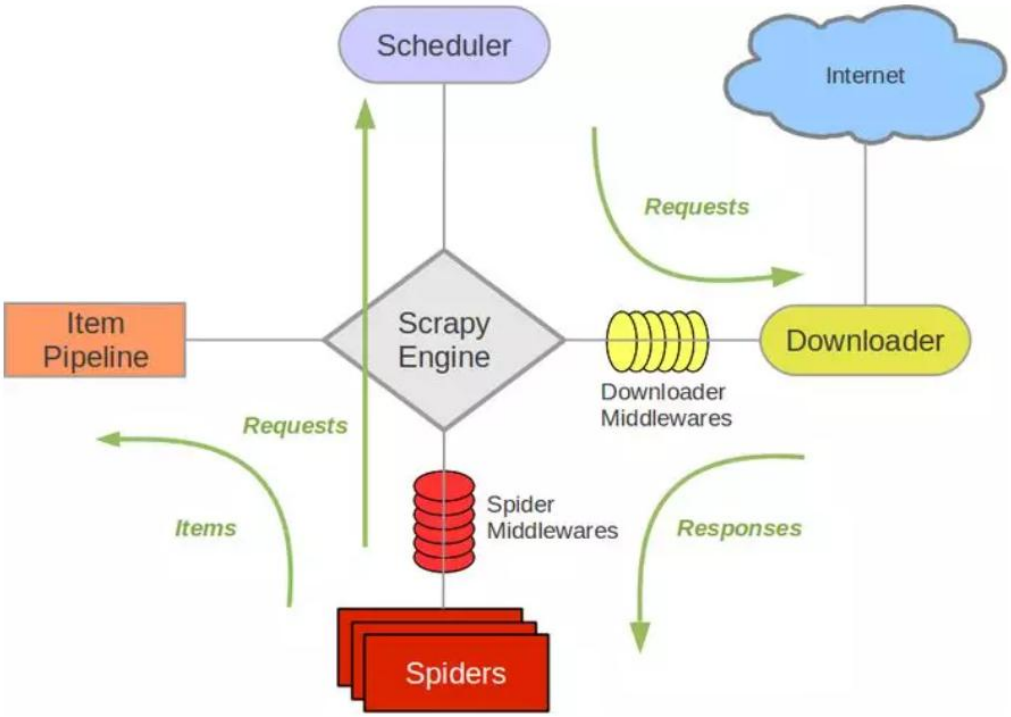
分布式爬虫实战


服务器
网站：
模块1
模块2
调度：消息队列
服务器2
网站：
模块1
模块2
模块3
服务器3
网站：
模块1
模块2
模块...
服务器...
用户访问
用户访问
Actor
Actor
Actor
Actor
Actor
用户...

分布式


服务器
网站
服务器2
网站
服务器3
网站
服务器...
用户访问
用户访问
Actor
Actor
Actor
Actor
Actor
用户...

集群式



分布式爬虫框架

一台用来搭建 redis

一台用来搭建数据库

再搞三台来运行爬虫

服务器买起

slave-02	68.183.185.255
slave-01	68.183.188.184
slave-03	68.183.181.83
redis	68.183.180.0
database	68.183.180.71

分布式服务器部署

- **反爬虫：**

1. CSS 加密

网站为了反爬虫，自定义字体对关键数据加密，使用编码代替数据且每次返回的编码不同。

破解：根据某一次返回的编码，根据编码对应的内容找出其映射关系。大部分网站使用的是同一字体定义但编码不同，所以根据这次映射关系构建编码映射字典。每次爬取时，首先逐个对比新编码与旧编码直至编码对应内容相同，则将旧编码对应数字映射给新编码从而完成解密。

2. JS 加密——请求参数加密

对请求的 form data 加密。

破解：debug 调试源代码获得加密逻辑，python 模拟加密形成 form data 从而正确请求。

3. 网站反 debug

返回的源代码里进入调试模式便递归调用 debug，从而使得我们无法 debug。

破解：使用中间代理如 mitmweb 抓包截取返回源代码进行修改，再返回即可。

- **Python 学习心得：**

1. Yield 关键字出现的函数为生成器，没有返回值，不会返回，而是 yield 生成指定的东西。

2. Python 可读写 excel、数据可视化等等，for 循环的区间左闭右开

3. Python 没有类似于 C 的块结构（即｛｝括起来），取而代之使用对齐（对齐方式一致的为一块），可以用制表符 TAB

4. 正则表达式就类似于字符串匹配，用于提取信息。Python 的 re 模块可以很好实现

5. Python 中 in 的用法

in	如果在指定的序列中找到值返回 True，否则返回 False。	x 在 y 序列中，如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x 不在 y 序列中，如果 x 不在 y 序列中返回 True。

is	is 是判断两个标识符是不是引用自一个对象	**x is y**, 类似 **id(x) == id(y)** , 如果引用的是同一个对象则返回 True，否则返回 False
is not	is not 是判断两个标识符是不是引用自不同对象	**x is not y** ， 类似 **id(a) != id(b)**。如果引用的不是同一个对象则返回结果 True，否则返回 False。

1. 在 for 循环中，获取列表或者元组的每一项：

```
for item in list:
```

2. 判断左边的元素是否包含于列表，类似java中的List的contains方法

```
if 1 in aa:
  print 'Very Good'
else:
  print 'Not Bad'
```

这里是判断 1 是否在 aa 内部

3. 可以用来判断字符串是否包含某一串,可以用来筛选文件使用

```
if 'a' in 'qa'
  print 'ok'
```

4. 比如判断project_admin是否是数字1或者字符串“1”

```
if project_admin in (1, "1")
```

主意：如果是想比较两个字符串是否相等，还是需要用 == 或者 !=