## Scrapy改进部分1

#### 1改进说明

##### 1.1 Scrapy分布式爬虫实现。

Scrapy是基于Twisted异步框架的单机爬虫，目前只能在一台机器上进行爬取，而且受限于Python的GIL，在多核CPU上也只能利用到一个CPU核心。所以，我们决定将scrapy改造成一个多机分布式爬虫，实现多台机器对同一个爬取任务的共同完成。

##### 1.2 Scrapy两级过滤器改进与实现。

Scrapy采用Python内置的集合（set）实现内存过滤器，同时提供文件过滤器让用户选择。两者并不排斥，文件过滤器的设置是为了可以停止爬虫（结束爬虫进程），然后在之后的某一个时间内重新接着上次的爬取­­继续时，可以直接从文件中读取已经爬取的url。实际运行期间的过滤都是在内存过滤器（set集合实现的过滤器）进行的。

默认的Python集合需要存储网页本身的url，同时为了实现管理集合以及其他一些针对集合的操作，Python需要保存的内容除了URL之外还有许多，因此，当爬取大量网页的时候占用内存会比较大。而分析发现，我们实际上只需要判断一个url是否已经出现过，而至于url是什么或者何时出现并不在乎。其次，我们也不会对已经爬取的url进行索引等操作，所以Python内置的set集合本身的许多操作相对来说是冗余的。

结合我们决定要实现的Scrapy分布式爬虫，我们决定结合布隆过滤器实现两级过滤器。单机爬虫首先在本机的布隆过滤器中对新生成的url进行过滤，过滤失败则在继续分布式的过滤器中进行过滤。

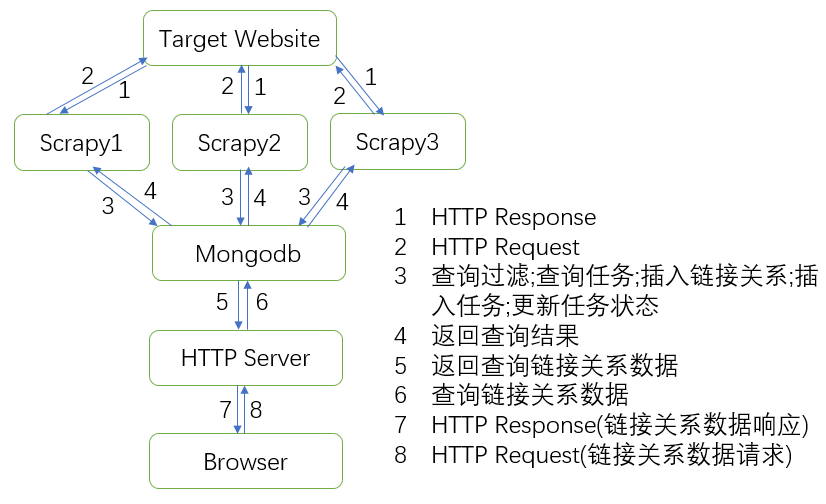
#### 2实现方案

##### 2.1分布式Scrapy实现方案。

2.1.1简要说明

一个完整的分布式爬虫涉及到包括通信、日志系统等许多方面，目前我们决定只实现一个多机协作的分布式Scrapy爬虫。具体实现方案是，使用mongodb文档作为分布式的任务队列以及分布式的过滤器。不同机器的爬虫生成的新的Request加入到分布式mongodb文档中。每个爬虫都从分布式的mongodb文档中取任务。此外，同样使用一个mongodb文档作为过滤器，实现多机协同的过滤器。在每个爬虫本地，实现一个布隆过滤器，用于进行本地过滤。

2.1.2架构图



2.1.3Mongodb任务队列和过滤器文档结构

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| \_id | string | url的MD5哈希值作为唯一标识 |
| url | string | url |
| req | string | Request信息 |
| status | string | Request状态（new|finished|downloading） |
| priority | int | Request优先级 |
| enqueue\_date | date | 入队时间 |

2.1.4Mongodb链接关系文档结构

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| \_id | string | url和target拼接后的MD5哈希值 |
| url | string | 源URL |
| target | String | 外链指向URL |

##### 2.2两级过滤器实现方案。

2.2.1简要说明

两级过滤器为本地的布隆过滤器和分布式mongodb文档。爬虫首先在本地的布隆过滤器中进行查找，如果发现已经下载过该url内容，则直接过滤掉。如果查找失败，则到分布式文档中进行查找。作为过滤器的分布式文档与作为任务队列的分布式文档使用同一个文档，其中由status字段的信息来表明Request目前的状态。如果是new，则表明需要出于任务队列中未爬取Request；如果是downloading，表示该任务为活跃任务，正在进行爬取；如果是finished则表明该任务已经爬取结束。

2.2.2布隆过滤器介绍

布隆过滤器（英语：Bloom Filter）是1970年由布隆提出的。它实际上是一个很长的[二进制](https://zh.wikipedia.org/wiki/%E4%BA%8C%E8%BF%9B%E5%88%B6)向量和一系列随机[映射函数](https://zh.wikipedia.org/w/index.php?title=%E6%98%A0%E5%B0%84%E5%87%BD%E6%95%B0&action=edit&redlink=1)。布隆过滤器可以用于检索一个元素是否在一个集合中。它的优点是空间效率和查询时间都远远超过一般的算法，缺点是有一定的误识别率和删除困难。