scrapy-redis搭建分布式爬虫环境

scrapy-redis简介

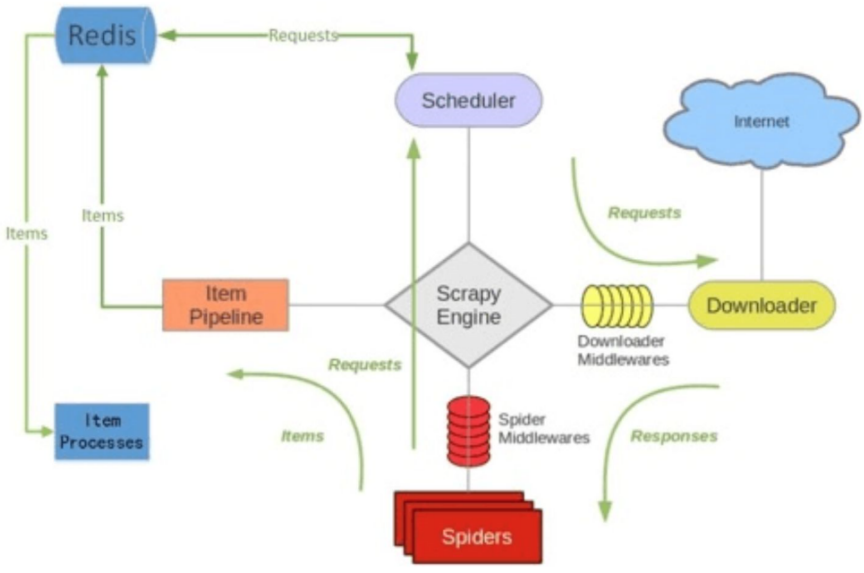
scrapy-redis是scrapy框架基于redis数据库的组件，用于scrapy项目的分布式开发和部署。

- 官方文档：https://scrapy-redis.readthedocs.io/en/stable/
- 源码位置：https://github.com/rmax/scrapy-redis
- 参考博客：https://www.cnblogs.com/kylinlin/p/5198233.html

优势:

- 分布式爬取
  可以启动多个spider工程，相互之间共享单个redis的requests队列。最适合广泛的多个域名网站的内容爬取。
- 分布式数据处理
  爬取到的scrapy的item数据可以推入到redis队列中，这意味着你可以根据需求启动尽可能多的处理程序来共享item的队列，进行item数据持久化处理
- Scrapy即插即用组件
  Scheduler调度器 + Duplication复制 过滤器， Item Pipeline，基本spider

scrapy-redis架构



- 首先Slaver端从Master端拿任务（Request、url）进行数据抓取，Slaver抓取数据的同时，产生新任务的Request便提交给 Master 处理；
- Master端只有一个Redis数据库，负责将未处理的Request去重和任务分配，将处理后的Request加入待爬队列，并且存储爬取的数据。

Scrapy-Redis默认使用的就是这种策略，实现简单，因为任务调度等工作Scrapy-Redis都已经帮我们做好了，我们只需要继承RedisSpider、指定redis_key就行了。

- 缺点: `Scrapy-Redis`调度的任务是`Request`对象，里面信息量比较大（不仅包含`url`，还有`callback`函数、`headers`等信息）
    - 降低爬虫速度
    - 占用`Redis`大量的存储空间
    - 需要一定硬件水平

`scrapy-redis`常用配置

```
# 使用了scrapy_redis的去重组件，在redis数据库里做去重
DUPEFILTER_CLASS = "scrapy_redis.dupefilter.RFPDupeFilter"

# 使用了scrapy_redis的调度器，在redis里分配请求
SCHEDULER = "scrapy_redis.scheduler.Scheduler"

#  在redis中保持scrapy-redis用到的各个队列，从而允许暂停和暂停后恢复，也就是不清理redis queues
SCHEDULER_PERSIST = True

# 通过配置RedisPipeline将item写入key为 spider.name : items 的redis的list中，供后面的分布式处理item 这个已经由 scrapy-redis 实现，不需要我们写代码，直接使用即可
ITEM_PIPELINES = {
    'scrapy_redis.pipelines.RedisPipeline': 100 ,
}


# 指定redis数据库的连接参数
REDIS_HOST = '127.0.0.1'
REDIS_PORT = 6379
```

`scrapy-redis`键名介绍

`scrapy-redis`中都是用key-value形式存储数据，其中有几个常见的key-value形式

名称	类型	含义
项目名:items	list	保存爬虫获取到的数据item 内容是 `json` 字符串
项目名:`dupefilter`	set	用于爬虫访问的URL去重 内容是 40个字符的 `url` 的hash字符串
项目名:`start_urls`	List	用于获取spider启动时爬取的第一个`url`
项目名:requests	`zset`	用于scheduler调度处理 requests 内容是 request 对象的序列化 字符串

`scrapy-redis`简单实例

在原来非分布式爬虫的基础上，使用`scrapy-redis`简单搭建一个分布式爬虫，过程只需要修改下面文件:

- `settings.py`
- `spider`文件
  - 继承类：由`scrapy.Spider`修改为`RedisSpider`
  - `start_url`已经不需要了，修改为：`redis_key = "xxxxx"`
  - 在`redis`数据库中，设置一个`redis_key`的值，作为初始的`url`，`scrapy`就会自动在`redis`中取出`redis_key`的值，作为初始`url`，实现自动爬取。

```
lpush xxxx:start_urls http:xxxxx.com/xxxx
```