Design and Implementation of Distributed Crawler System Based on Scrapy

Yuhao Fan

Research Institute Electronic Science and Technology University of Electronic Science and Technology of China, ChengDu 611731, China
13558798397@163.com

Abstract. At present, some large-scale search engines at home and abroad only provide users with non-custom search services, and a single-machine web crawler cannot solve the difficult task. In this paper, through the study and research of the original Scrapy framework, the original Scrapy framework is improved by combining Scrapy and Redis, a distributed crawler system based on Web information Scrapy framework is designed and implemented, and Bloom Filter algorithm is applied to dupefilter module to reduce memory consumption. The movie information captured from douban is stored in MongoDB, so that the data can be processed and analyzed. The results show that distributed crawler system based on Scrapy framework is more efficient and stable than the single-machine web crawler system.

1. Introduction

With the rapid development of the network, the World Wide Web has become a carrier of a large amount of information, how to effectively extract and use this information has become a huge challenge. The traditional general search engine to become the user access to the World Wide Web portal and guide, but some large-scale search engines at home and abroad only provide users with non-custom search services, and a single-machine web crawler cannot assume the difficult task. So the distributed web crawler with flexible customization, high information acquisition speed and large scale has come into being.

2. Overview of the Scrapy architecture

Scrapy is written with Twisted, a popular event-driven networking framework for Python. It consists of five parts, as shown in figure 1.

Scrapy Engine: The engine is responsible for controlling the data flow between all components of the system, and triggering events when certain actions occur.

Scheduler: The Scheduler receives requests from the engine and enqueues them for feeding them later (also to the engine) when engine requests them.

Downloader: The Downloader is responsible for fetching web pages and feeding them to the engine which, in turn, feeds them to the spiders.

Spiders: Spiders are custom classes written by Scrapy users to parse responses and extract items (scraped items) from them or additional requests to follow.
Item Pipeline: The Item Pipeline is responsible for processing the items once they have been extracted (or scraped) by the spiders. Typical tasks include cleansing, validation and persistence (like storing the item in a database).

![Fig. 1 Scrapy Architecture](image)

3. Design and Implementation of Distributed Web Crawler System

For distributed web crawler, it’s important to communicate with each other on a web crawler. At present, there are two solutions: one is Master-Slave mode, the other is Peer to Peer mode. Master-Slave architecture is widely used in distributed scenarios where a control node is responsible for the communication with all stand-alone web crawlers. Slaves only need to acquire the requests from the master. After downloading the requests, the extracted new request is returned to the master. As shown in figure 2.

![Fig. 2 Master-Slave mode](image)

In this paper, we use Master/Slave mode to achieve a distributed web crawler system by combining the scrapy and redis database. The new scheduler stores and schedules the fetched requests through redis and stores the items for crawling for subsequent processing.

3.1. Redis introduction

1. What is redis?

Redis is a key-value storage system written by Salvatore Sanfilippo. Redis provides a wealth of data structures, including lists, sets, ordered sets, and hashes, as well as rich operational methods for these data structures.

2. The advantages of redis

- High performance. Redis can read and write more than 105 seconds.
- Atomicity. Redis all operations are based on "atomic" as the basic unit, Redis also supports "atomic" merger operation.
3.2. The basic composition of Scrapy-Redis

(1) connection
Configure the settings, connect the redis database, dupefilter module and scheduler module call, the relevant Redis access operation to use this module.

(2) dupefilter
Using the redis set data type to store the request to the re-processing, scheduler does not use the module's dupefilter key to dispatch request, but by defining the queue. If it is not a duplicate request, it will be stored in the queue and be scheduled when the pop-up.

(3) queue
There are currently three types of queue scheduling: FIFO SpiderQueue, SpiderPriorityQueue, and LIFO SpiderStack, the default use SpiderPriorityQueue.

(4) pipelines
Redis is used to implement distributed data processing modules. Unlike Scrapy's Pipline, it receives project data from multiple crawlers and processes the data in accordance with the program settings.

(5) scheduler
Scheduler uses Redis to achieve distributed data distribution module. Instead of scrapy comes with the scheduler component function, all access to the url after the crawler processing, by the scheduler unified management. All crawlers will get the target url uniformly from the scheduler. Achieving a unified crawler management to avoid the repeated crawling. The basic composition of Scrapy-Redis as shown in figure 3.

![Fig. 3 The extended Scrapy framework](image)

4. Dupefilter module optimization
Using the redis own data type Set to store the captured URL queue can avoid repeated crawling. But with the increase in the number of URLs, the collection is getting bigger and bigger, while redis is the memory-based database, resulting in a very high memory. Bloom filter only use a series of bit to save the data, you can detect whether an element already exists in the collection, so this algorithm has a very good space utilization.

4.1. Bloom Filter algorithm introduction
Bloom Filter is a space-efficient random data structure that uses a bit array to represent a collection and judge whether an element belongs to that collection. Bloom Filter’s efficiency is a certain price: when judging whether an element belongs to a collection, it is possible that the elements do not belong to this collection mistaken for this set (false positive). Therefore, Bloom Filter is not suitable for those "zero error" applications. And in applications that tolerate low error rates, Bloom Filter makes a significant savings in storage space with minimal errors, as shown in Figure 4.
4.2. Based on Bloom Filter algorithm flow:
(1) Initialize an array of v, the length of M, the initial value is 0.
(2) The url that crawl from web need to be processed by k hash functions.
(3) For the generated k random numbers, the search algorithm is used to find the bit variable v.
(4) If the k random bits in the bit vector v corresponding bit is 1, then the url has been in the collection, and jump back to step 2; otherwise continue.
(5) Add the url to the collection and skip back to step 2.

5. Analysis of experimental results

5.1. experimental results
According to the original url of the Douban movie, the basic information of the movie is captured by the distributed crawler system and stored in MongoDB. The result of the crawling is shown in Figure 5.

5.2. Analysis of results
Deployment of the entire system: 1 server with redis and MongoDB databases installed as the master node of the crawler; 1 server with MongoDB database installed as slave; 1 single server with MongoDB database installed.

By analyzing the data in Table 1 we can see that each node crawls the number of pages per hour in 1400 or so. Due to network latency, I/O delay, etc., the number of pages per hour to crawl slightly lower than the theoretical maximum of 1800. In addition, the load of two reptile nodes is balanced. Simultaneous results also show that the extended Scrapy framework makes the crawl speed greatly improved.
Table 1. results of experiment

| Hours               | 1 hours | 2 hours | 3 hours | 4 hours |
|---------------------|---------|---------|---------|---------|
| (Single node) pages | 1535    | 3519    | 4589    | 6298    |
| movies              | 1417    | 3412    | 4441    | 6100    |
| (Double node) pages | 3578    | 6912    | 8971    | 12568   |
| movies              | 3416    | 6835    | 8701    | 11892   |

According to the data in Table 2, we can get the graph of the graph by Matlab. The slope of each line represents the crawling efficiency of the crawler. It can be seen from Figure 6 that the efficiency of a node for crawling is much lower than that of the two nodes at the same time. In summary, the distributed crawler crawl efficiency is much higher than the single crawler.

Fig. 6 Speed comparison chart

6. Concluding remarks
In this paper, we go deep into the exploration and expansion of the open source web crawler Scrapy framework to design and implement a distributed web crawler system, but there are still many deficiencies in the system, for example, all urls are stored in the redis database, once the main node fails, the entire system will not be able to continue working, The robustness of the system needs to be improved.

References
[1] Brin S, Page L. Peprint of: The anatomy of a large-scale hypertextual web search engine [J]. Computer Networks,2012,56 (18):3825-3833.
[2] Lawrence S, Giles C L. Accessibility of information on web [J]. Nature,1999.
[3] Iijima S. Helical Microtubes of Graphite Carbon[J]. Nature,1991354 (6348):56—58.
[4] Boldi P, Codenotti B, Santini M, et al. UbiCrawler: a scalable fully distributed Web crawler [J]. Software Practice & Experience,2004,34 (8):711-726.
[5] Pinkerton B, Lazowska E, Zahorjan J. Weberalwer: Finding What People Want [J]. 2001.