基于Redis的分布式爬虫框架的设计

文/高 强
（山东青年政治学院）

摘 要： 随着互联网的高速发展，在互联网搜索服务中，搜索引擎扮演着越来越重要的角色。网络爬虫是搜索引擎系统中十分重要的组成部分，它负责从互联网中搜集网页，这些页面用于建立索引，从而为搜索引擎提供支持。面对当前极具膨胀的网络信息，集中式的单机爬虫早已无法适应目前的互联网信息规模，因此高性能的分布式网络爬虫系统成为目前信息采集领域研究的重点。本文对网络爬虫原理、分布式架构设计以及网络爬虫中的关键模块、瓶颈问题及解决办法进行了相关研究。

关键字： 分布式爬虫；多线程；Redis；Python

Design of Distributed Crawler Framework Based on Redis

GAO Qiang
（Shandong Youth Politics Institute）

Abstract: With the rapid development of Internet, search engine plays an increasingly important role in Internet search service. Web crawler is a very important component of search engine system. It is responsible for collecting web pages from the Internet, which is used to build indexes so as to provide support for search engines. Because of the great expansion of network information, centralized stand-alone web crawler has been unable to adapt to the Internet scale, so high-performance distributed web crawler system has become the focus of current research in the field of information collection. In this paper, the principles of web crawler, the design of distributed architecture, and the key modules, bottlenecks and solutions of crawler were studied.

Key words: distributed crawler; multithread; Redis; Python

随着互联网的高速发展，在互联网搜索服务中，搜索引擎扮演着越来越重要的角色。网络爬虫是搜索引擎系统中十分重要的组成部分，它负责从互联网中搜集网页，这些页面用于建立索引从而为搜索引擎提供支持。面对当前极速膨胀的网络信息，集中式的单机爬虫早已无法适应目前的互联网信息规模，因此高性能的分布式网络爬虫系统成为目前信息采集领域研究的重点。

本文阐述了在windows7 64位操作系统环境下通过Python语言编写的一个基于广度优先算法的分布式多进程网络爬虫，主要说明了分布式爬虫程序中实现的过程及一些主要问题，以及为何使用广度优先的爬行策略和使用多进程的原因及实现方法，及爬虫的数据存储问题等。

一、分布式网络爬虫概述

网络爬虫（又被称为网页蜘蛛、网络机器人

等），是按照一定的程序规则，自动的获取网页内容并自动按条件的筛选出信息的程序。

分布式爬虫，可以理解为集群爬虫程序，即将多个单机集中式爬虫程序组合连接到一起，分布式系统中的每一个节点都可以看作是一个集中式网络爬虫。分布式爬虫与集中式爬虫程序工作原理相同，但分布式爬虫程序由各个节点分工协作来完成网页信息的爬取，多台机器同时运行，从而使分布式爬虫效率远远高于集中式爬虫程序，达到更高效率、更快地抓取网页信息的目的[1]。

二、分布式网络爬虫工作原理

分布式爬虫问题其实也就是多台机器多个节点对多个url的同时处理问题，怎样schedule这些url，怎样汇总spider抓取的数据。最简单粗暴的方法就是将url进行分片，交给不同机器，最后对不同机器抓取的数据进行汇总。然而这样每个节点只能对自己处理的url去重，没办法全局的去重，另外性能也很难控制，可能有某台机器很早就跑完了，而别的机器还要跑很久。另一种思路就是把url存在某个地方，共享给所有的机器，总的调度器来分配请求，判断节点有没有闲置，闲置了就继续给它任务，直到所有的url都爬完，这种方法解决了去重问题，也能提高性能。本程序使用的是后者，即把url共享给所有节点，同时分配任务给闲置的节点。

（一）Redis

Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API。Redis提供了大量的数据结构，比如string、list、set、hashset、sorted set等，极大地方便了用户的使用；redis将数据存放在内存中，因此读写速度非常快，很适合当作分布式爬虫程序的url池的载体以及进行调度任务的调度器[2]。

（二）爬取策略

爬虫深度优先算法：深度优先搜索是一种在开发爬虫早期使用较多的方法。它的目的是要达到被搜索结构的叶结点(即那些不包含任何超链的HTML文件）。在一个HTML文件中，当一个超链被选择后，被链接的HTML文件将执行深度优先搜索，即在搜索其余的超链结果之前必须先完整地搜索单独的一条链。深度优先搜索沿着HTML文件上的超链走到不能再深入为止，然后返回到某一个HTML文件，再继续选择该HTML文件中的其他超链。当不再有其他超链可选择时，说明搜索已经结束。优点是能遍历一个Web 站点或深层嵌套的文档集合；缺点是因为Web结构相当深，有可能造成一旦进去，再也出不来的情况发生。

爬虫广度优先算法：整个的广度优先爬虫过程就是从一系列的种子节点开始，把这些网页中的"子节点"(也就是超链接)提取出来，放入队列中依次进行抓取。被处理过的链接需要放 入一张表(通常称为Visited表)中。每次新处理一个链接之前，需要查看这个链接是否已经存在于Visited表中。如果存在，证明链接已经处理过，跳过，不做处理，否则进行下一步处理。

广度优先遍历是爬虫中使用最广泛的一种爬虫策略，本程序使用的即是广度优先算法。使用广度优先搜索策略的主要原因有三点：重要的网页往往离种子比较近；万维网的实际深度最多能达到17 层，但到达某个网页总存在一条很短的路径，而广度优先遍历会以最快的速度到达这个网页；广度优先有利于多爬虫的合作抓取，多爬虫合作通常先抓取站内链接，抓取的封闭性很强。

（三）多进程

python中的多线程其实并不是真正的多线程，如果想要充分地使用多核CPU的资源，在python中大部分情况需要使用多进程。Python提供了非常好用的多进程multiprocessing包，只需要定义一个函数，Python会完成其他所有事情[3]。借助这个包可以轻松完成从单进程到并发执行的转换。multiprocessing支持子进程、通信和共享数据、执行不同形式的同步，提供了Process、Queue、Pipe、Lock等组件。

实现方案：

Mongodb数据库中用于存放url对应的记录，每条记录格式为：

```
{
  “_id”:url,
  “state”:OUTSTANDING|PROCESSING|COMPLETE
```

"deepth"：

}

其中，用url作为id，使得数据库中的url唯一；state记录url的访问状态——未访问、正在访问、访问过；deepth代表当前url的深度，在广度优先情况下对深度进行控制。同时，MongoDB的虽然不支持事务，但原子操作保证了多线程之间的并发同步。Client端运行相同的爬虫程序，从mongodb数据库队列中取出url进行爬取页面；由于数据库存储速率的瓶颈，本次选择将页面存储在本地，通过正则表达式提取页面的url，并将其加入mongodb数据库队列。Client端采用"多线程+多进程"的方式，使用线程池和进程池，进程数目等于cpu核心数目，每个进程中线程数目为5。

（四）第三方库

需要用的第三方库有：redis、requests、Lxml。

redis库提供了Python连接操纵Redis的API，安装过程：打开cmd窗口，运行pip install redis等待安装完成，安装完成后，再从cmd窗口中运行pip install cssselect完成cssselect模块的安装。

Requests库是一常用的http请求库，它使用Python语言编写，可以方便地发送http请求，以及方便地处理响应结果。安装过程：打开cmd窗口，运行pip install requests等待安装完成即可。

Lxml库是python的一个html/xml解析并建立dom的库，Lxml的特点是功能强大，性能也不错[4]。用Lxml库抓取网页信息原理：先将HTML网页进行解析，然后按照标签进行筛选抓取。

（五）相关技术点

Redis通常被认为是一种持久化的存储器关键字-值型存储，可以用于几台机子之间的数据共享平台[5]。

连接数据库：

注意：假设现有几台在同一局域网内的机器分别为Master和几个Slaver，Master连接时host为localhost即本机的ip。

```
_db = redis.Reds(host='localhost', port=6379, db=0)
```

Slaver连接时的host也为Master的ip，端口port和数据库db不写时为默认值6379、0。

例如，Master可利用redis将url上传到数据库：

```
for i in range(20): # 将需爬取的前20页的url并存入urls集合
    url = 'http://www.qiushibaike.com/hot/page/%d/' % (i + 1)
    _db.sadd('urls', url) # 将url插入关键字urls集合中，若url已存在则不再插入，进而Master和Slaver可从数据库里获取url。
url = _db.spop('urls') # 随机从urls集合中取出一个url返回并将其删去。
```

三、结束语

本文给出了基于Redis的分布式爬虫的原型，未来网络爬虫的研究将围绕如何提高链接的准确性，降低进行计算的时空复杂度，以及增加网络爬虫的自适应性，进一步提高爬虫的性能。完

作者简介：高强（1979-），男，硕士，讲师，研究方向：数据库、数据挖掘、大数据的分布式存储等。

收稿日期：2017-07-09

参考文献

[1]（澳）理查德劳森（Richard Lawson）.用Python写网络爬虫[M].北京：人民邮电出版社，2016.

[2]黄健宏.Redis设计与实现[M].北京：机械工业出版社，2014.

[3]刘凌霞，郝宁波，吴海涛.21天学通PYTHON[M].北京：电子工业出版社，2016.

[4]李晓明，闫宏飞，王继民. 搜索引擎——原理，技术与系统[M].北京：科学出版社，2005.

[5]董晓英.网络环境下信息资源的管理与信息服务[M]. 北京：中国对外翻译出版公司，2000.