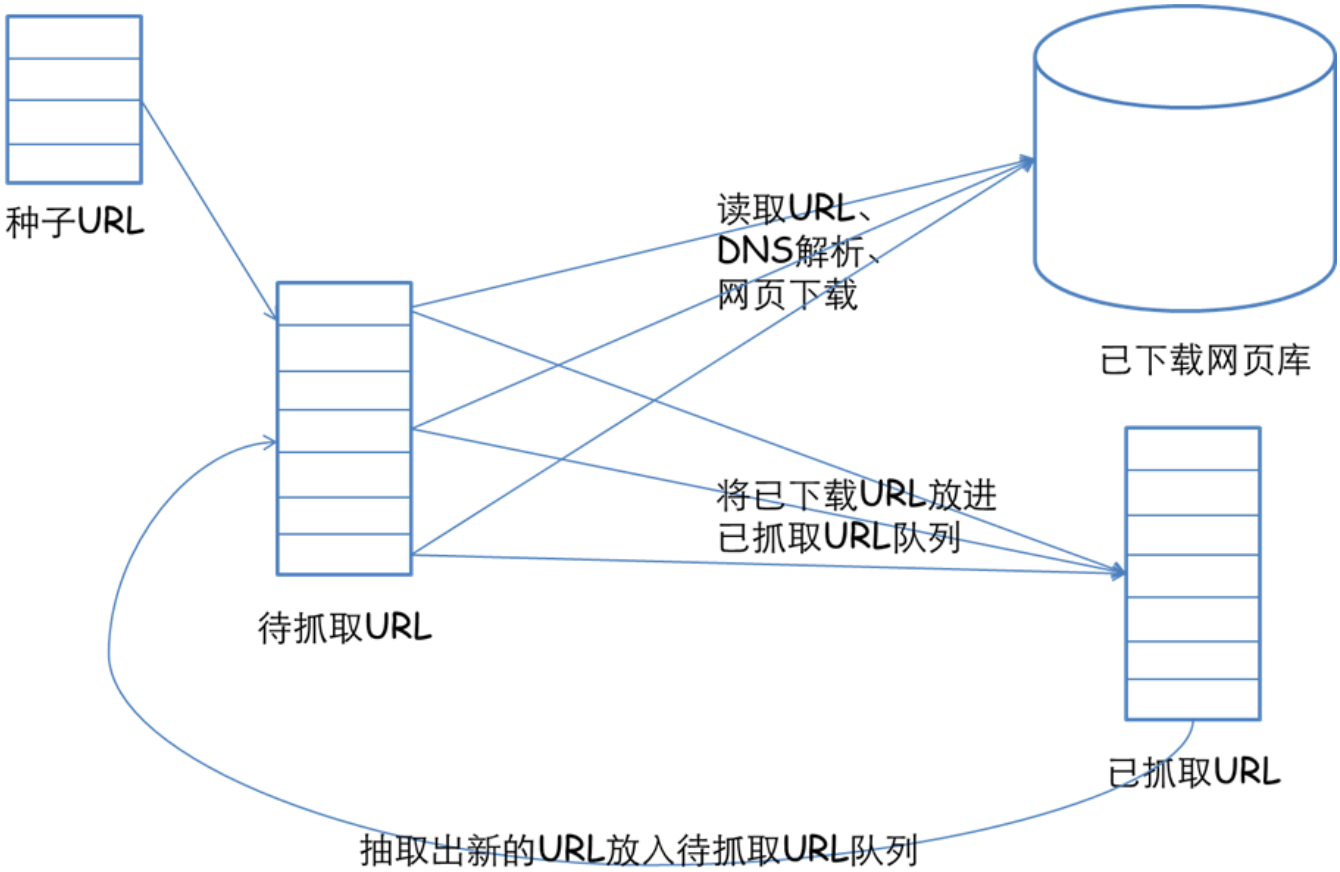
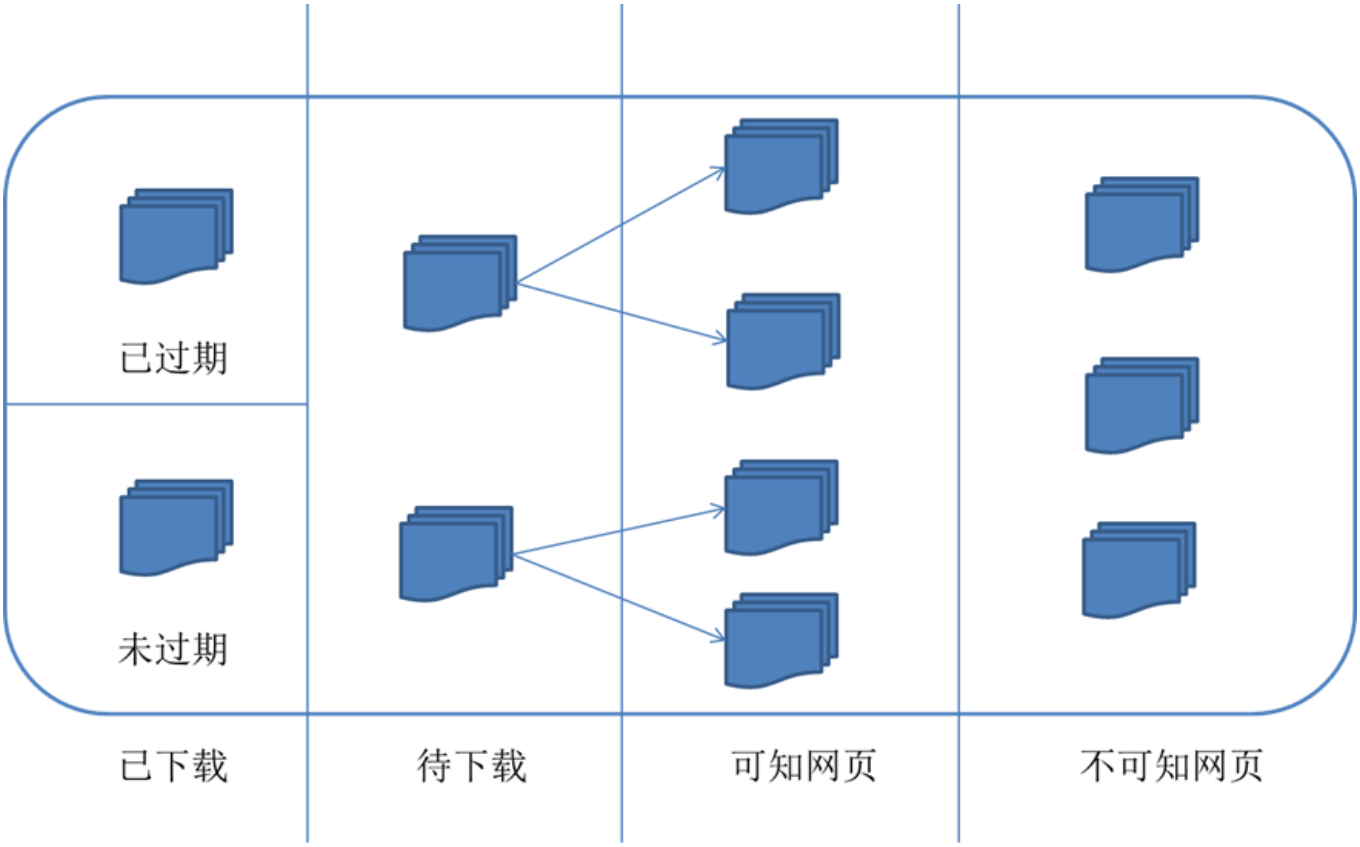
一、网络爬虫的基本工作流程



1. 首先选取一部分精心挑选的种子URL；
2. 将这些URL放入待抓取URL队列；
3. 从待抓取URL队列中取出待抓取在URL，解析DNS，并且得到主机的ip，并将URL对应的网页下载下来，存储进已下载网页库中。此外，将这些URL放进已抓取URL队列。
4. 分析已抓取URL队列中的URL，分析其中的其他URL，并且将URL放入待抓取URL队列，从而进入下一个循环。

二、从爬虫的角度对互联网进行划分

对应的，可以将互联网的所有页面分为五个类型：



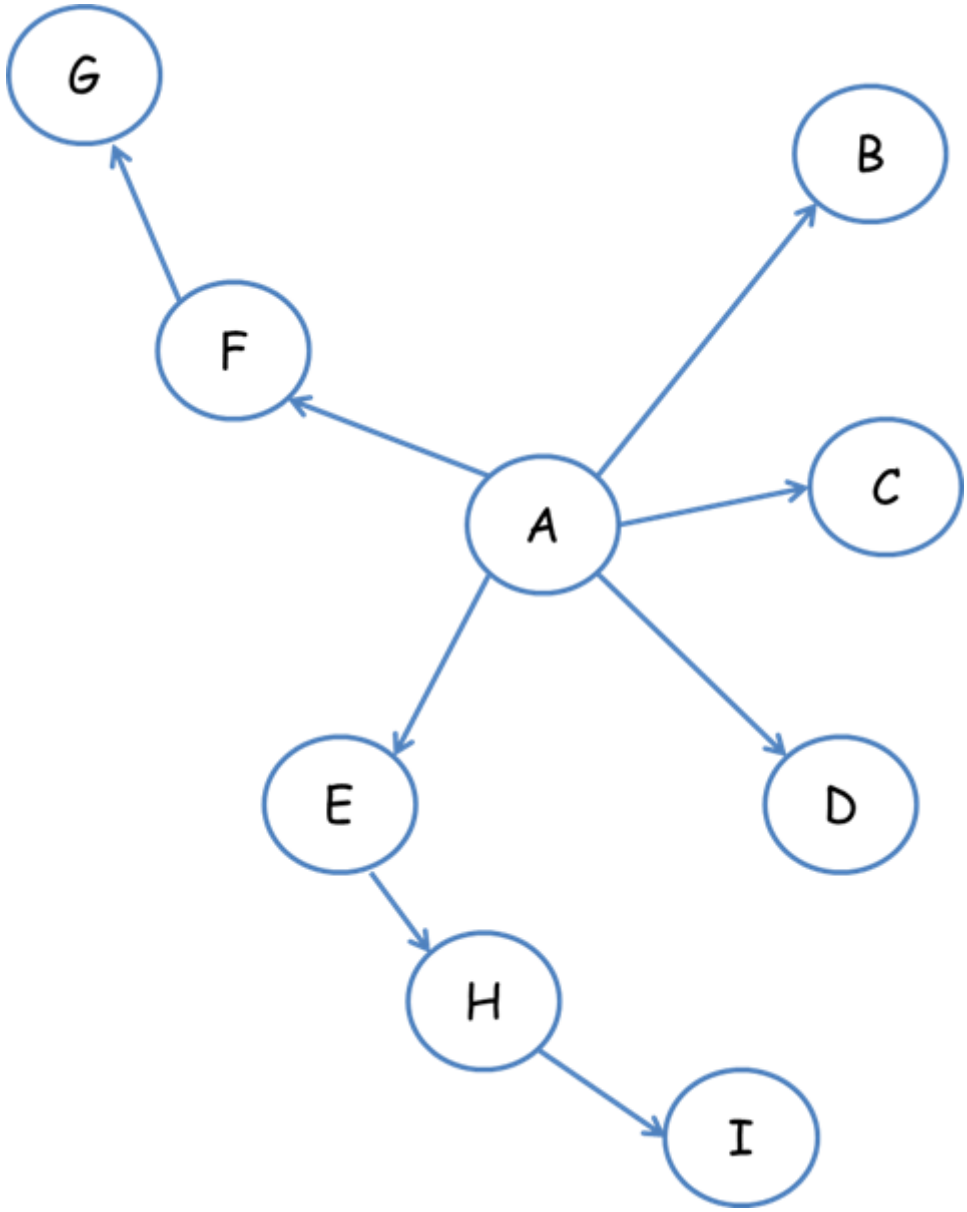
1. 已下载未过期网页
2. 已下载已过期网页：抓取到的网页实际上是互联网内容的一个镜像与备份，互联网是动态变化的，一部分互联网上的内容已经发生了变化，这时，这部分抓取到的网页就已经过期了。
3. 待下载网页：也就是待抓取URL队列中的那些页面
4. 可知网页：还没有抓取下来，也没有在待抓取URL队列中，但是可以通过对已抓取页面或者待抓取URL对应页面进行分析获取到的URL，认为是可知网页。
5. 还有一部分网页，爬虫是无法直接抓取下载的。称为不可知网页。

三、抓取策略

在爬虫系统中，待抓取URL队列是很重要的一部分。待抓取URL队列中的URL以什么样的顺序排列也是一个很重要的问题，因为这涉及到先抓取那个页面，后抓取哪个页面。而决定这些URL排列顺序的方法，叫做抓取策略。下面重点介绍几种常见的抓取策略：

1. 深度优先遍历策略

深度优先遍历策略是指网络爬虫会从起始页开始，一个链接一个链接跟踪下去，处理完这条线路之后再转入下一个起始页，继续跟踪链接。我们以下面的图为例：



遍历的路径：A-F-G E-H-I B C D

1. 宽度优先遍历策略

宽度优先遍历策略的基本思路是，将新下载网页中发现的链接直接插入待抓取URL队列的末尾。也就是指网络爬虫会先抓取起始网页中链接的所有网页，然后再选择其中的一个链接网页，继续抓取在此网页中链接的所有网页。还是以上面的图为例：

遍历路径：A-B-C-D-E-F G H I

1. 反向链接数策略

反向链接数是指一个网页被其他网页链接指向的数量。反向链接数表示的是一个网页的内容受到其他人的推荐的程度。因此，很多时候搜索引擎的抓取系统会使用这个指标来评价网页的重要程度，从而决定不同网页的抓取先后顺序。

在真实的网络环境中，由于广告链接、作弊链接的存在，反向链接数不能完全等他我那个也的重要程度。因此，搜索引擎往往考虑一些可靠的反向链接数。

1. Partial PageRank 策略

Partial PageRank算法借鉴了PageRank算法的思想：对于已经下载的网页，连同待抓取URL队列中的URL，形成网页集合，计算每个页面的PageRank值，计算完之后，将待抓取URL队列中的URL按照PageRank值的大小排列，并按照该顺序抓取页面。

该算法基于两个直觉

- 当一个网页被更多网页所链接时，其排名会越靠前；
- 排名高的网页应具有更大的表决权，即当一个网页被排名高的网页所链接时，其重要性也应对应提高。

1. OPIC策略（在线页面重要性计算 Online Page Importance Computation）

该算法实际上也是对页面进行一个重要性打分。在算法开始前，给所有页面一个相同的初始现金（cash）。当下载了某个页面P之后，将P的现金分摊给所有从P中分析出的链接，并且将P的现金清空。对于待抓取URL队列中的所有页面按照现金数进行排序。

与PageRank的区别在于：PageRank每次需要迭代计算，而OPIC策略不需要迭代过程。所以计算速度远远快与PageRank，适合实时计算使用。

1. 大站优先策略

对于待抓取URL队列中的所有网页，根据所属的网站进行分类。对于待下载页面数多的网站，优先下载。这个策略也因此叫做大站优先策略。

四、更新策略

互联网是实时变化的，具有很强的动态性。网页更新策略主要是决定何时更新之前已经下载过的页面。常见的更新策略又以下三种：

1. 历史参考策略

顾名思义，根据页面以往的历史更新数据，预测该页面未来何时会发生变化。一般来说，是通过泊松过程进行建模进行预测。

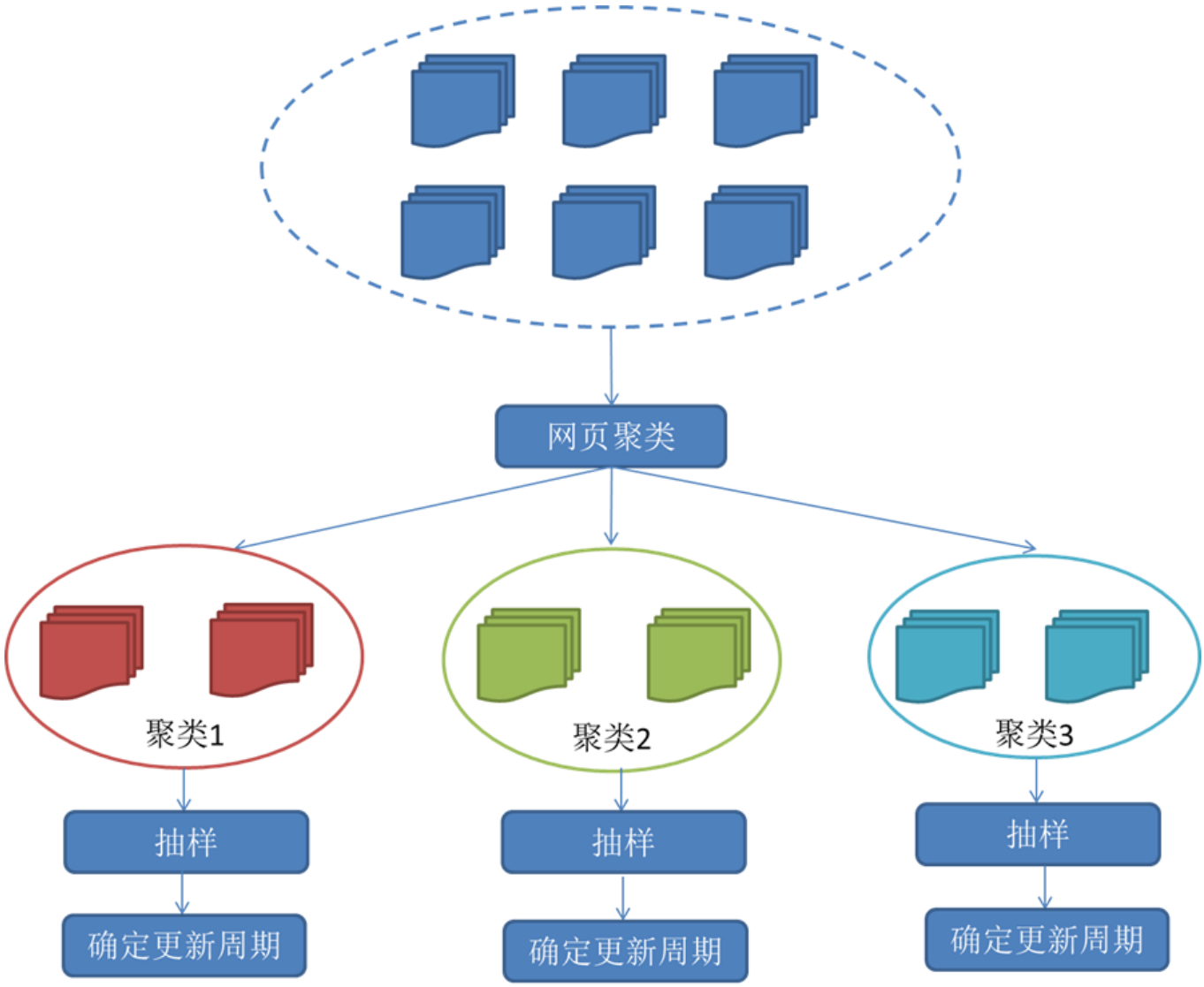
1. 用户体验策略

尽管搜索引擎针对于某个查询条件能够返回数量巨大的结果，但是用户往往只关注前几页结果。因此，抓取系统可以优先更新那些现实在查询结果前几页中的网页，而后再更新那些后面的网页。这种更新策略也是需要用到历史信息的。用户体验策略保留网页的多个历史版本，并且根据过去每次内容变化对搜索质量的影响，得出一个平均值，用这个值作为决定何时重新抓取的依据。

1. 聚类抽样策略

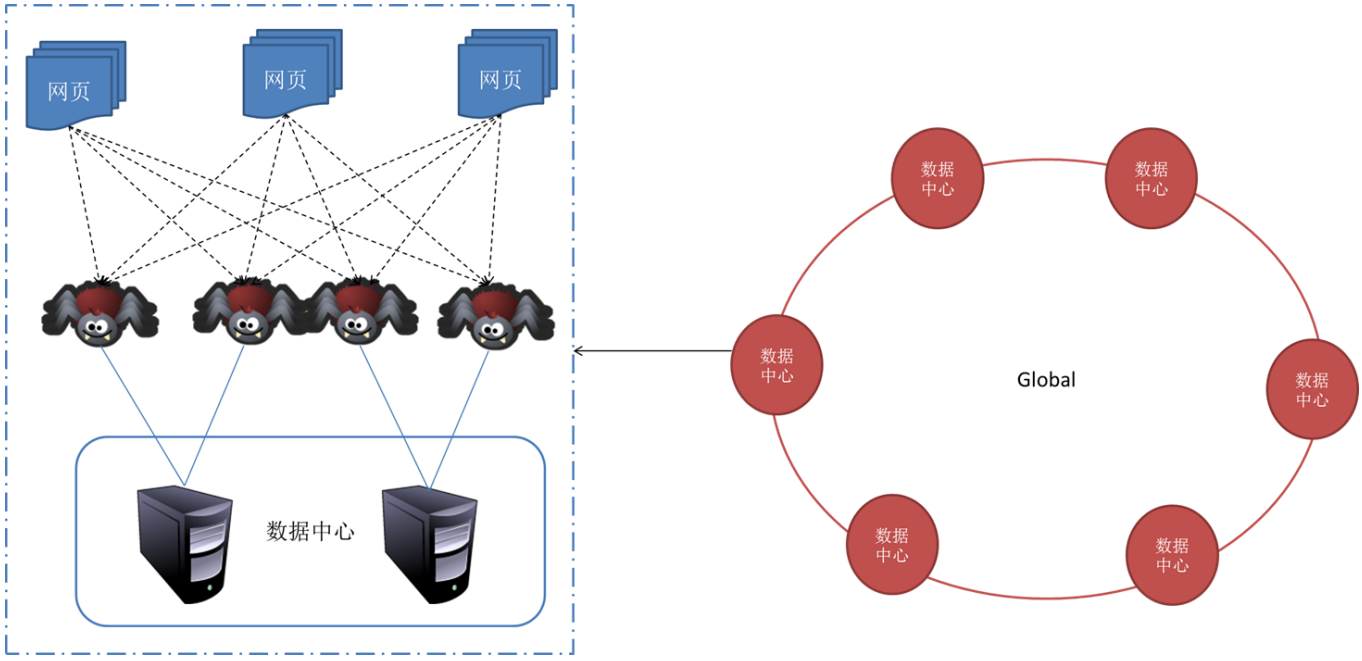
前面提到的两种更新策略都有一个前提：需要网页的历史信息。这样就存在两个问题：第一，系统要是为每个系统保存多个版本的历史信息，无疑增加了很多的系统负担；第二，要是新的网页完全没有历史信息，就无法确定更新策略。

这种策略认为，网页具有很多属性，类似属性的网页，可以认为其更新频率也是类似的。要计算某一个类别网页的更新频率，只需要对这一类网页抽样，以他们的更新周期作为整个类别的更新周期。基本思路如图：



五、分布式抓取系统结构

一般来说，抓取系统需要面对的是整个互联网上数以亿计的网页。单个抓取程序不可能完成这样的任务。往往需要多个抓取程序一起来处理。一般来说抓取系统往往是一个分布式的三层结构。如图所示：

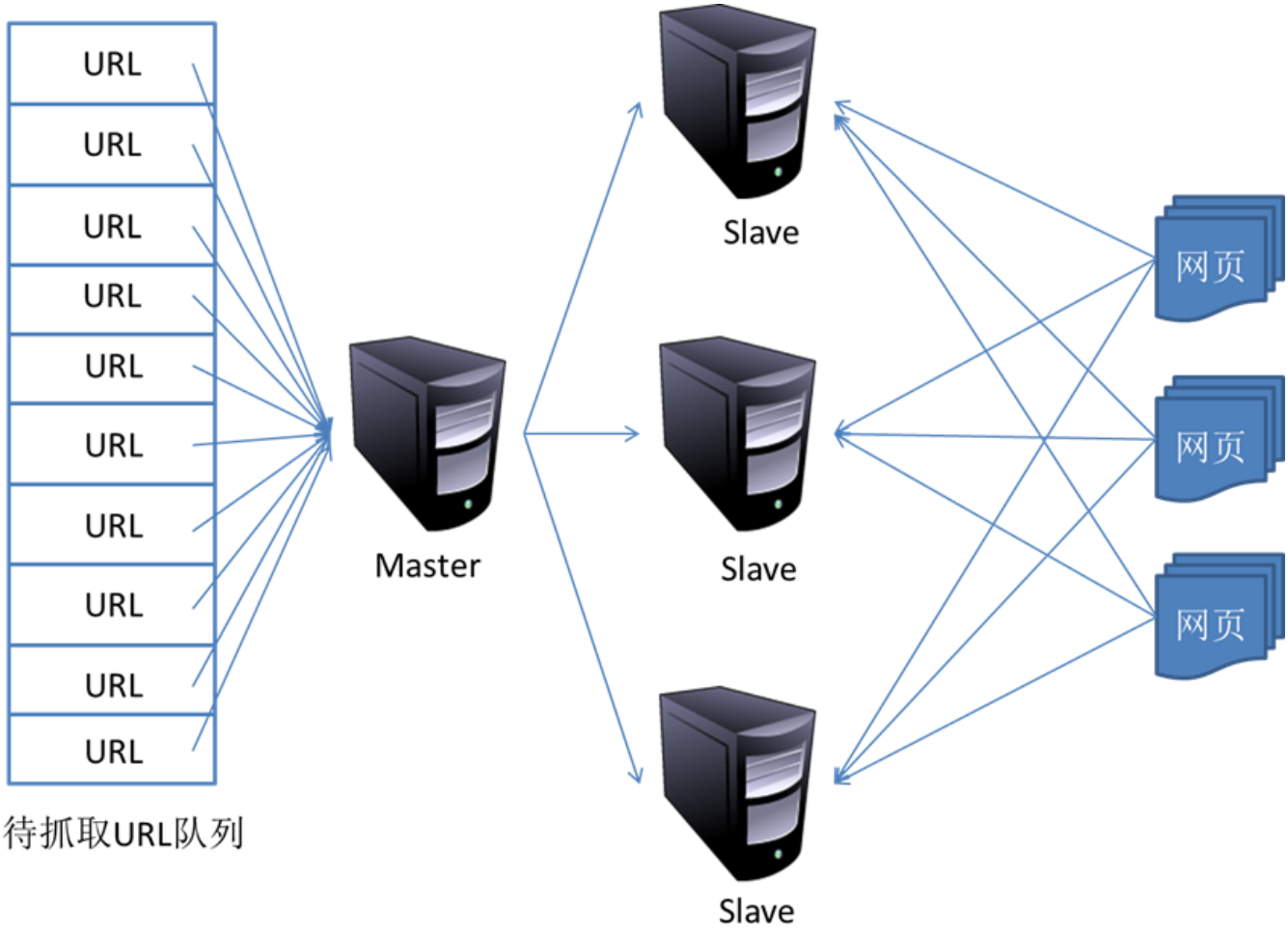


最下一层是分布在不同地理位置的数据中心，在每个数据中心里有若干台抓取服务器，而每台抓取服务器上可能部署了若干套爬虫程序。这就构成了一个基本的分布式抓取系统。

对于一个数据中心内的不同抓去服务器，协同工作的方式有几种：

1. 主从式（Master-Slave）

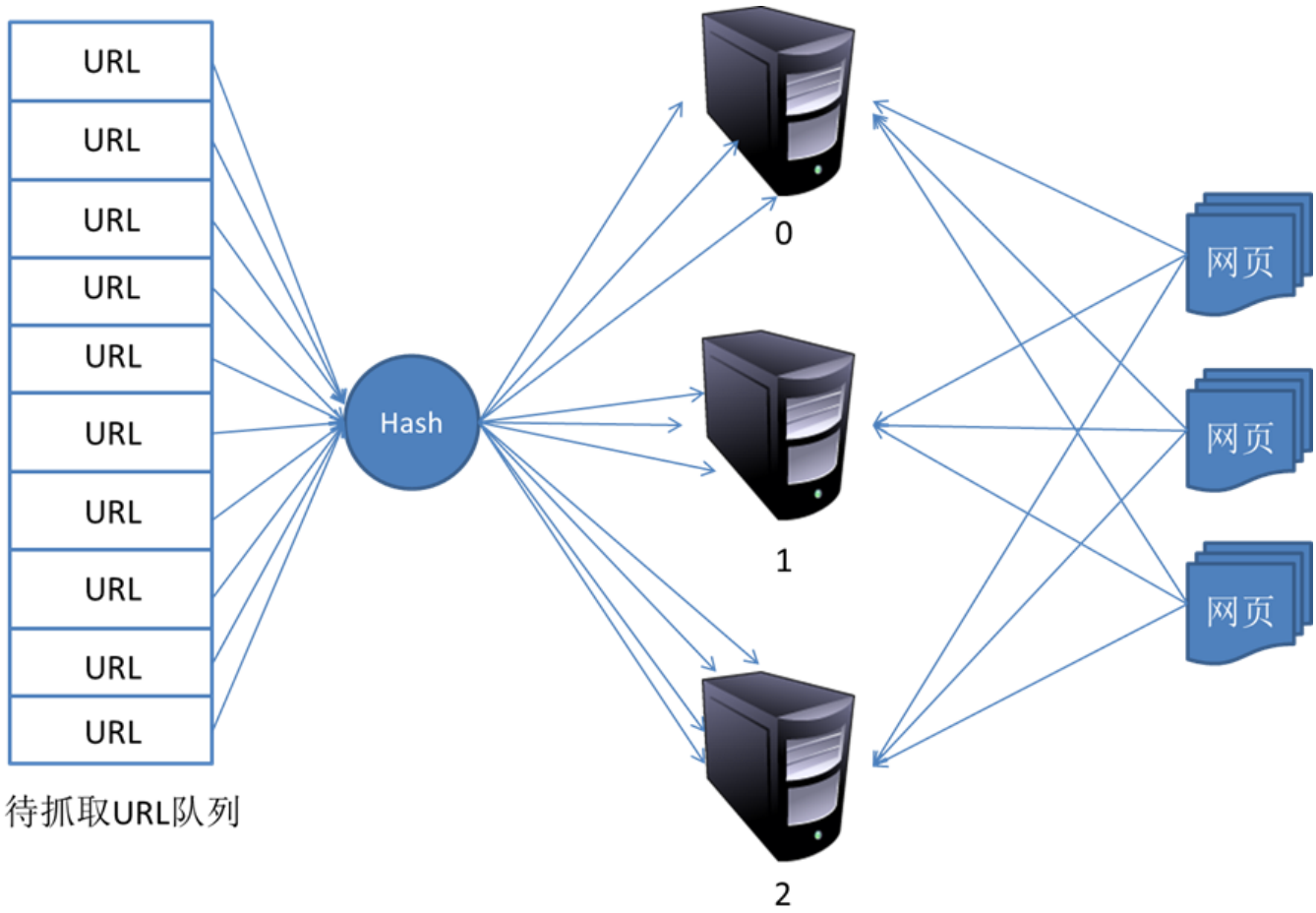
主从式基本结构如图所示：



对于主从式而言，有一台专门的Master服务器来维护待抓取URL队列，它负责每次将URL分发到不同的Slave服务器，而Slave服务器则负责实际的网页下载工作。Master服务器除了维护待抓取URL队列以及分发URL之外，还要负责调解各个Slave服务器的负载情况。以免某些Slave服务器过于清闲或者劳累。 这种模式下，Master往往容易成为系统瓶颈。

1. 对等式（Peer to Peer）

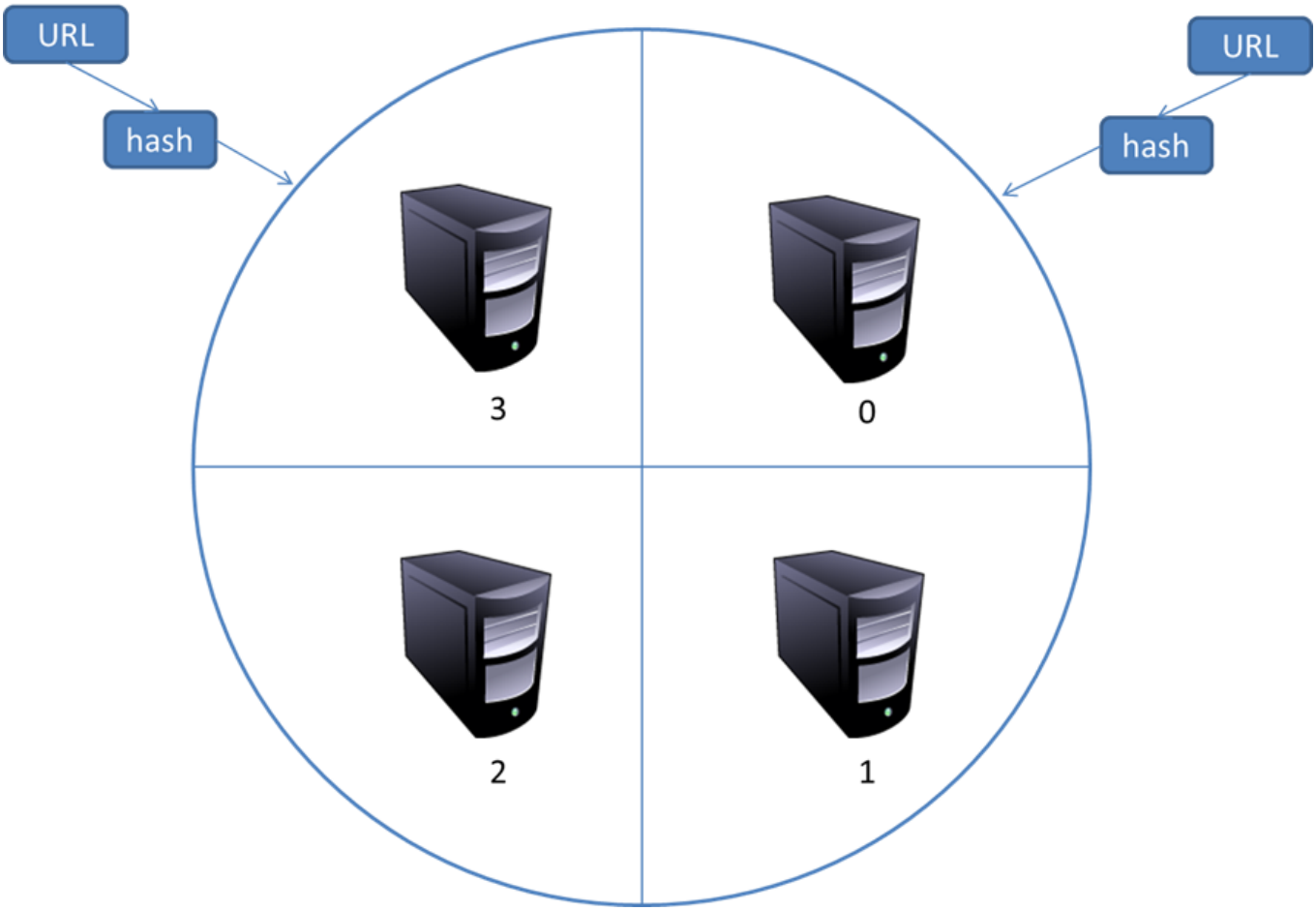
对等式的基本结构如图所示：



在这种模式下，所有的抓取服务器在分工上没有不同。每一台抓取服务器都可以从待抓取在URL队列中获取URL，然后对该URL的主域名的hash值H，然后计算H mod m（其中m是服务器的数量，以上图为例，m为3），计算得到的数就是处理该URL的主机编号。

举例：假设对于URL www.baidu.com，计算器hash值H=8，m=3，则H mod m=2，因此由编号为2的服务器进行该链接的抓取。假设这时候是0号服务器拿到这个URL，那么它将该URL转给服务器2，由服务器2进行抓取。

这种模式有一个问题，当有一台服务器死机或者添加新的服务器，那么所有URL的哈希求余的结果就都要变化。也就是说，这种方式的扩展性不佳。针对这种情况，又有一种改进方案被提出来。这种改进的方案是一致性哈希法来确定服务器分工。其基本结构如图所示：



一致性哈希将URL的主域名进行哈希运算，映射为一个范围在0-232之间的某个数。而将这个范围平均的分配给m台服务器，根据URL主域名哈希运算的值所处的范围判断是哪台服务器来进行抓取。

如果某一台服务器出现问题，那么本该由该服务器负责的网页则按照顺时针顺延，由下一台服务器进行抓取。这样的话，及时某台服务器出现问题，也不会影响其他的工作。

In [ ]:

```
# HTTP请求的Python实现
## 使用urlib3 、 urllib
# urllib作为Python的标准库，基本上涵盖了基础的网络请求功能。
# Urllib3是一个功能强大，条理清晰，用于HTTP客户端的Python库。需要安装 $ pip install urllib3
from urllib import request
r = request.urlopen('http://www.zhihu.com') # 这是一个get请求
r.status # HTTP状态码 200表示请求成功 303表示重定向 400表示请求错误 401表示未授权 403表示禁止访问 404表示文件未找到 500表示服务器错误
#print(r.read().decode()) # 编码格式不正确 print(resp.read().decode())
```

下面详细介绍urllib

In [ ]:

```
# post
from urllib import request

resp = request.urlopen('http://httpbin.org/post', data=b'word=hello') #   括号内加 ， timeout=10
print(resp.read().decode())
```

u/U:表示unicode字符串 不是仅仅是针对中文, 可以针对任何的字符串，代表是对字符串进行unicode编码。 一般英文字符在使用各种编码下, 基本都可以正常解析, 所以一般不带u；但是中文, 必须表明所需编码, 否则一旦编码转换就会出现乱码。 建议所有编码方式采用utf8

r/R:非转义的原始字符串 与普通字符相比，其他相对特殊的字符，其中可能包含转义字符，即那些，反斜杠加上对应字母，表示对应的特殊含义的，比如最常见的"\n"表示换行，"\t"表示Tab等。而如果是以r开头，那么说明后面的字符，都是普通的字符了，即如果是"\n"那么表示一个反斜杠字符，一个字母n，而不是表示换行了。以r开头的字符，常用于正则表达式，对应着re模块。

b:bytes python3.x里默认的str是(py2.x里的)unicode, bytes是(py2.x)的str, b"“前缀代表的就是bytes

In [ ]:

```
import urllib3
url = "http://httpbin.org"
fields = {
  'name':'xfy'
}
http = urllib3.PoolManager()
r = http.request('post',url+"/post",fields=fields)
print(r.data.decode())
```

可以看到很简单，只是第一个参数get换成了post。

并且参数不需要再像urllib一样转换成byte型了。

Accept-Encoding，HTTP Header中Accept-Encoding 是浏览器发给服务器,声明浏览器支持的编码类型

"Content-Type": 即是Internet Media Type，互联网媒体类型，也叫做MIME类型。在互联网中有成百上千中不同的数据类型，HTTP在传输数据对象时会为他们打上称为MIME的数据格式标签，用于区分数据类型。常见的Content-Type有数百个，下面例举了一些

- HTML文档标记：text/html;
- JPEG图片标记：image/jpeg;
- js文档标记：application/javascript;
- xml文件标记：application/xml;

但是有时候post请求没有问题，但是服务器拒绝你的请求，问题出在请求中的头信息，服务器会检验请求头，来判断是否来自于浏览器的访问。这也是反爬虫的常用手段

User Agent中文名为用户代理，简称 UA，它是一个特殊字符串头，使得服务器能够识别客户使用的操作系统及版本、CPU 类型、浏览器及版本、浏览器渲染引擎、浏览器语言、浏览器插件等。

设置headers

In [ ]:

```
import urllib3
headers = {
   'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36'
}
http = urllib3.PoolManager();
r = http.request('get',url+"/get",headers = headers)
print(r.data.decode())
```

设置代理

https://www.kuaidaili.com/free/intr/ (https://www.kuaidaili.com/free/intr/)

In [ ]:

```
import urllib3
url = "http://httpbin.org"
headers = {
   'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36'
}
proxy = urllib3.ProxyManager('http://124.192.106.247:3128',headers = headers)
r = proxy.request('get',url+"/ip")
print(r.data.decode())
```

上述结果可能会很慢

当请求的参数为json

在发起请求时,可以通过定义body 参数并定义headers的Content-Type参数来发送一个已经过编译的JSON数据

In [ ]:

```
import urllib3
url = "http://httpbin.org"
import json
data = {'name':'田冉'}

json_data = json.dumps(data)

http = urllib3.PoolManager()
r = http.request('post',url+"/post",body = json_data,headers = {'Content-Type':'application/json'})
print(r.data.decode('unicode_escape'))
```

上传文件

In [ ]:

```
#元组形式
with open('a.html','rb') as f:
    data = f.read()
http = urllib3.PoolManager()
r = http.request('post','http://httpbin.org/post',fields = {'filefield':('a.html',data,'text/plain')})
print(r.data.decode())

#二进制形式

r = http.request('post','http://httpbin.org/post',body = data,headers={'Content-Type':'image/jpeg'})
print(r.data.decode())
```

超时设置

In [ ]:

```
# 1全局设置超时
# http = urllib3.PoolManager(timeout = 3)
# 2在request里设置
# http.request('post','http://httpbin.org/post',timeout = 3)
from urllib import request
resp = request.urlopen('http://httpbin.org/post', data=b'word=hello') #   括号内加 ， timeout=10
print(resp.read().decode())
```

重试和重定向

In [ ]:

```
import urllib3
http = urllib3.PoolManager()
#重试
r = http.request('post','http://httpbin.org/post',retries = 5) #请求重试测次数为5次 ， 默认为3次
print(r.retries) #Retry(total=5, connect=None, read=None, redirect=0, status=None)
#关闭重试
http.request('post','http://httpbin.org/post',retries = False) #请求重试测次数为5次 ， 默认为3次

r = http.request('get','http://httpbin.org/redirect/1',redirect = False)
print(r.retries)# Retry(total=3, connect=None, read=None, redirect=None, status=None)
print(r.status)
print(r.data.decode())
print("----------------------")
print(r.get_redirect_location())
#302不是异常
```

302重定向又称之为暂时性转移(Temporarily Moved)，英文名称：302 redirect。 也被认为是暂时重定向（temporary redirect），一条对网站浏览器的指令来显示浏览器被要求显示的不同的URL，当一个网页经历过短期的URL的变化时使用。一个暂时重定向是一种服务器端的重定向，能够被搜索引擎蜘蛛正确地处理。

Http 302对应生活中的真实例子，可以类比手机所对应的呼叫转移功能，这样打进A手机的电话，均转移到B手机接听。

urllib3 本身设置了https的处理，但是有警告

In [ ]:

```
import urllib3
http = urllib3.PoolManager()
r = http.request('GET', 'https://www.zhihu.com') #http://httpbin.org/robots.txt
r.status
# r.data #r.data.decode()
```

In [ ]:

```
import urllib3
urllib3.disable_warnings() #禁用各种警告 配置文件，一次配置，（进程生效？）
url = "https://www.12306.cn/mormhweb/"
http = urllib3.PoolManager()
r = http.request('get',url)
print(r.data.decode())
```

In [ ]:

```
import os
filename = 'data.txt'
pathDir = './'
if os.path.exists(pathDir + '/' + filename):
    print(filename + '文件在' + pathDir + '中存在 ！')
else:
    # 打开文件,不存在则创建 新建文件data.txt
    file = open('./data.txt', 'w')
    print(filename + '文件不存在目录下,新建文件data.txt ')
    # 将文件名写入到指定文件中
    file.write(filename)
    # 关闭
    file.close()
```

In [ ]:

```
import time
import os
from selenium import webdriver
filename = 'data.txt'
pathDir = './'
def sleeptime(hour,min,sec):
    return hour*3600 + min*60 + sec;
second = sleeptime(0,0,300);
#这是隔10秒执行一次，控制抓取时间，实际应用为5分钟，300秒

while True:
    input_str = ""
    input_str1 = ""
    input_str2 = ""
# browser = webdriver.Firefox()
    browser = webdriver.Firefox(executable_path='geckodriver')
    browser.get('http://jiaotong.baidu.com/top/report/?citycode=36&qq-pf-to=pcqq.c2c')
    data_time = browser.find_element_by_class_name("clock-info").text
    data_CityCongestionIndex = browser.find_element_by_class_name("container3").text
    data_CongestionRoad = browser.find_element_by_class_name("cc-datalist-list").text
    print("开始抓取数据")
    get_time = time.strftime("%Y/%m/%d ")+data_time[0:8]
    print("抓取时间：",get_time)
    # print("城市拥挤程度：",data_CityCongestionIndex)
    mainRoad = data_CityCongestionIndex[data_CityCongestionIndex.index("平均速度")-5:data_CityCo
ngestionIndex.index("平均速度")-1]
    print("城市主干路拥挤指数：", mainRoad)
    speedRoad = data_CityCongestionIndex[data_CityCongestionIndex.index("平均速度")-10:data_City
CongestionIndex.index("平均速度")-5]
    print("城市高速路拥挤指数：", speedRoad)
    str = data_CongestionRoad
    input_str1 = get_time +","+mainRoad+","+speedRoad
    for i in range(10):
```

```
        print("拥堵路:", str[0:str.index("\n")])
        input_str2 = input_str2 + "," + str[0:str.index("\n")]
        str = str[str.index("\n") + 1:]
        print("拥堵路名称:", str[0:str.index("\n")])
        input_str2 = input_str2 + "," + str[0:str.index("\n")]
        str = str[str.index("\n") + 1:]

        print("拥堵指数:", str[0:str.index("\n")])
        input_str2 = input_str2 + "," + str[0:str.index("\n")]
        str = str[str.index("\n") + 1:]

        if i==9 :
            print("平均速度:", str[0:])
            input_str2 = input_str2 + "," + str[0:]
        else:
            print("平均速度:", str[0:str.index("\n")])
            input_str2 = input_str2 + "," + str[0:str.index("\n")]
            str = str[str.index("\n") + 1:]

    input_str = input_str1+input_str2 +"\n"
    #写入数据
    file = open(pathDir+'/'+filename, 'a')
    file.write(input_str)
    file.close()
    print("结束抓取数据")
    browser.quit()
    time.sleep(second);
```

Selenium是一个用于Web应用程序测试的工具。Selenium 测试直接运行在浏览器中,就像真正的用户在操作一样。支持的浏览器包括IE(7, 8, 9, 10, 11),等

In [ ]:

```
# https://github.com/mozilla/geckodriver/releases
from selenium import webdriver
print(help(webdriver.Firefox) )
```

In [ ]:

```
import sys
print(sys.path)
```

In [3]:

```
import pandas as pd
data = pd.read_csv('data.txt', encoding='ANSI') # 开始不加encoding='ANSI'会报错。也可以指定标题
 names= ['a', 'b', 'c', 'd', 'e']
print(data)
```

```
Empty DataFrame
Columns: [2019/03/18 08:51:21, 2.25, 0.98 , 1, 兰工坪路, 3.98, 9.01, 2, 福利西路,
3.65, 10.49, 3, 和定路, 3.54, 13.35, 4, 南滨河东路, 2.95, 13.29, 5, 万新北路(安宁
西路附近，南向北附近), 2.92, 13.07, 6, 北滨河中路, 2.85, 17.89, 7, 白银路, 2.69, 1
4.09, 8, 盐场路, 2.39, 16.70, 9, 万新北路(万新北路附近，北向南附近), 2.23, 17.09,
10, 东岗西路, 2.23.1, 18.53]
Index: []

[0 rows x 43 columns]
```

In [24]:

```
# Python将数据保存到CSV中
import numpy as np
arr1 = np.arange(100).reshape(10,10)
arr1
```

Out[24]:

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

In [25]:

```
import pandas as pd
data1 = pd.DataFrame(arr1)
data1.to_csv('data1.csv')
```

In [26]:

```
# 这个例子不讲
# 大部分情况下，我们不需要行、列信息。则代码改为：
import pandas as pd
data2 = pd.DataFrame(arr1, index = None, columns = None) # columns:原第一行的索引，index:原第一列的索引
data2.to_csv('data2.csv')
```

In [16]:

```
?pd.DataFrame # 查看函数说明
```

In [ ]:

```

```