我写了个非常简单的爬虫脚本，它爬取17k小说网的书名，并放入Redis中。在放入Redis之前，它会做一个检查，看是否数据已经被别的爬虫爬取过。如果没爬取就继续放入数据，否则不放入，继续爬取。（爬虫文件在spider/mycrawl.py）整体分布式爬虫框架设计为，Docker+Redis+MySQL。Docker跑爬虫，Redis负责数据处理，MySQL存储爬取后的数据。

首先下载docker的系统镜像：

[root@localhost ~]# docker pull docker.io/centos

Using default tag: latest

Trying to pull repository docker.io/library/centos ...

latest: Pulling from docker.io/library/centos

5e35d10a3eba: Pull complete

Digest: sha256:dcbc4e5e7052ea2306eed59563da1fec09196f2ecacbe042acbdcd2b44b05270

Status: Downloaded newer image for docker.io/centos:latest

用docker images看本地现有的镜像，确保下载成功：

[root@localhost ~]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

docker.io/ubuntu latest f975c5035748 11 days ago 112 MB

docker.io/centos latest 2d194b392dd1 12 days ago 195 MB

接着创建一波容器，并把它们相互进行连接：

[root@localhost ~]# docker run -tid --name h1 mytest:v1

eb0603dfe22e3d38b57c0311e0ec4e06c43d8351cb51dc6a9d48d4dc49af5449

[root@localhost ~]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

eb0603dfe22e mytest:v1 "/bin/bash" 9 seconds ago Up 8 seconds h1

14f1f3c200d4 f975 "/bin/bash" 49 minutes ago Up 48 minutes testabc

dfb1ede02a1b f975 "/bin/bash" About an hour ago Up About an hour blissful\_lamport

[root@localhost ~]# docker run -tid --name h2 --link h1 mytest:v1

e7057e5a0214790c9ffd72eef8430623604bb5aeab8c9c4b3f652571e33a8669

[root@localhost ~]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e7057e5a0214 mytest:v1 "/bin/bash" 8 seconds ago Up 6 seconds h2

eb0603dfe22e mytest:v1 "/bin/bash" 5 minutes ago Up 5 minutes h1

14f1f3c200d4 f975 "/bin/bash" 54 minutes ago Up 54 minutes testabc

dfb1ede02a1b f975 "/bin/bash" About an hour ago Up About an hour blissful\_lamport

[root@localhost ~]# docker attach e705

root@e7057e5a0214:/#

查看一下其对应的ip地址：

root@e7057e5a0214:/# cat /etc/hosts

127.0.0.1 localhost

::1 localhost ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

172.17.0.4 h1 eb0603dfe22e

172.17.0.5 e7057e5a0214

ping一下相应的地址，确保网络是通畅的：

[root@localhost ~]# ping 172.17.0.5

PING 172.17.0.5 (172.17.0.5) 56(84) bytes of data.

64 bytes from 172.17.0.5: icmp\_seq=1 ttl=64 time=0.296 ms

64 bytes from 172.17.0.5: icmp\_seq=2 ttl=64 time=0.155 ms

64 bytes from 172.17.0.5: icmp\_seq=3 ttl=64 time=0.087 ms

64 bytes from 172.17.0.5: icmp\_seq=4 ttl=64 time=0.091 ms

64 bytes from 172.17.0.5: icmp\_seq=5 ttl=64 time=0.119 ms

64 bytes from 172.17.0.5: icmp\_seq=6 ttl=64 time=0.090 ms

--- 172.17.0.5 ping statistics ---

6 packets transmitted, 6 received, 0% packet loss, time 5001ms

rtt min/avg/max/mdev = 0.087/0.139/0.296/0.075 ms

[root@localhost ~]# ping 172.17.0.4

PING 172.17.0.4 (172.17.0.4) 56(84) bytes of data.

64 bytes from 172.17.0.4: icmp\_seq=1 ttl=64 time=0.396 ms

64 bytes from 172.17.0.4: icmp\_seq=2 ttl=64 time=0.085 ms

64 bytes from 172.17.0.4: icmp\_seq=3 ttl=64 time=0.096 ms

64 bytes from 172.17.0.4: icmp\_seq=4 ttl=64 time=0.095 ms

64 bytes from 172.17.0.4: icmp\_seq=5 ttl=64 time=0.086 ms

64 bytes from 172.17.0.4: icmp\_seq=6 ttl=64 time=0.084 ms

--- 172.17.0.4 ping statistics ---

6 packets transmitted, 6 received, 0% packet loss, time 5001ms

更新一下apt程序：

root@e7057e5a0214:/# apt-get update

再来安装radis：

apt-get -y install redis-server

接下来运行redis：

root@e7057e5a0214:/# /etc/init.d/redis-server restart

Stopping redis-server: redis-server.

Starting redis-server: redis-server.

启动客户端，确保运行成功：

root@e7057e5a0214:/# redis-cli

127.0.0.1:6379>

127.0.0.1:6379> exit

把镜像储存为文件：

[root@localhost ~]# docker save -o /mytest.tar 037b

把中心节点创建出来，根据之前的镜像：

[root@localhost ~]# docker run -tid --name center 037b

e9fe479f065ca5695c3d01f950ad015445419963a070c957db8dcec66a55d868

再看中心节点的容器是否正常：

[root@localhost ~]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e9fe479f065c 037b "/bin/bash" 2 minutes ago Up 2 minutes center

1ed379bbf7e7 f975 "/bin/bash" About an hour ago Up About an hour server1

5015e53d83a6 f975 "-name server1" About an hour ago Created reverent\_williams

e7057e5a0214 mytest:v1 "/bin/bash" 32 hours ago Up 32 hours h2

eb0603dfe22e mytest:v1 "/bin/bash" 32 hours ago Up 32 hours h1

14f1f3c200d4 f975 "/bin/bash" 32 hours ago Up 32 hours testabc

dfb1ede02a1b f975 "/bin/bash" 33 hours ago Up 33 hours blissful\_lamport

准备中心节点并配置好Redis与MySQL

先进入中心节点服务器：

[root@localhost ~]# docker attach e9fe

然后再看网络详细配置：

root@e9fe479f065c:/# cat /etc/hosts

127.0.0.1 localhost

::1 localhost ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

172.17.0.7 e9fe479f065c

接着用本地ip连接mysql：

root@e9fe479f065c:/# mysql -h 127.0.0.1 -u root -p

接着来看redis，先启动一下redis：

root@e9fe479f065c:/# /etc/init.d/redis-server restart

然后再本地连接redis：

root@e9fe479f065c:/# redis-cli

127.0.0.1:6379>

是可以的

再尝试用远程连接：

root@e9fe479f065c:/# redis-cli -h 172.17.0.7

Could not connect to Redis at 172.17.0.7:6379: Connection refused

由于是拒绝连接，修改redis的配置文件，配置允许的ip：

root@e9fe479f065c:/# vim /etc/redis/redis.conf

重启redis：

root@e9fe479f065c:/# /etc/init.d/redis-server restart

Stopping redis-server: redis-server.

Starting redis-server: redis-server.

再次连接，还是被拒绝：

root@e9fe479f065c:/# redis-cli -h 172.17.0.7

Could not connect to Redis at 172.17.0.7:6379: Connection refused

重启一下容器。

[root@localhost ~]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e9fe479f065c 037b "/bin/bash" About an hour ago Up About an hour center

停止一下容器：

[root@localhost ~]# docker stop e9fe

e9fe

[root@localhost ~]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e9fe479f065c 037b "/bin/bash" About an hour ago Exited (0) 3 seconds ago center

启动容器，进入容器，重启redis：

[root@localhost ~]# docker start e9fe

e9fe

[root@localhost ~]# docker attach e9fe

root@e9fe479f065c:/#

root@e9fe479f065c:/#

root@e9fe479f065c:/# /etc/init.d/redis-server restart

Stopping redis-server: redis-server.

Starting redis-server: redis-server.

接着尝试连接，就能连接上了

root@e9fe479f065c:/# redis-cli -h 172.17.0.7

172.17.0.7:6379>

然后把mysql也重启一下：

root@e9fe479f065c:/# /etc/init.d/mysql restart

\* Stopping MySQL database server mysqld [ OK ]

\* Starting MySQL database server mysqld [ OK ]

准备子节点并调试

根据之前的镜像来创建子节点，并且与主节点相连：

[root@localhost /]# docker run -tid --name c1 --link center 037b

ebe6c483c9f9b056674e3bed5a3d4c50f27f1d464e5df23c3e1e46f2c943a577

再看一下是否与主网络相通：

root@ebe6c483c9f9:/# cat /etc/hosts

127.0.0.1 localhost

::1 localhost ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

172.17.0.7 center e9fe479f065c

172.17.0.8 ebe6c483c9f9

然后将mysql连接一下试一试：

root@ebe6c483c9f9:/# mysql -h 172.17.0.7 -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 4

Server version: 5.7.21-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

接下来检查redis能不能连接远程：

root@ebe6c483c9f9:/# redis-cli -h 172.17.0.7 -p 6379

172.17.0.7:6379>

子节点设置成功

确保镜像创造出来的容器中安装了redis包，如果没有需要安装：

root@ebe6c483c9f9:/# pip install redis

Collecting redis

Downloading redis-2.10.6-py2.py3-none-any.whl (64kB)

100% |################################| 71kB 214kB/s

Installing collected packages: redis

Successfully installed redis-2.10.6

把爬虫代码放在子节点的根目录下(在spider目录下的mycrawl.py)，用python命令去执行它，

root@ebe6c483c9f9:/# python mycrawl.py

再给这个容器做成镜像

[root@localhost /]# docker commit ebe6 crawl:zjd

接着依据这个镜像，批量的创造子节点。这样每个子节点的配置完全一样：

[root@localhost /]# docker run -tid --name c2 --link center e65d

848dcb31a3b3830e1021e5a0df9c20068c6f95cbc4ec3d06cf9d8dc3c2bb755c

[root@localhost /]# docker run -tid --name c3 --link center e65d

3097f39af25da4702e0fb1a68fcd0e3a5f33edd1f8e6d331dec724404c9efb76

[root@localhost /]# docker run -tid --name c4 --link center e65d

f4d716e1d6a1dc3c01db975448d9571c07f4daf798366620ddd07f7f3080cb54

[root@localhost /]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f4d716e1d6a1 e65d "/bin/bash" 21 seconds ago Up 20 seconds c4

3097f39af25d e65d "/bin/bash" 26 seconds ago Up 25 seconds c3

848dcb31a3b3 e65d "/bin/bash" 34 seconds ago Up 31 seconds c2

ebe6c483c9f9 037b "/bin/bash" 2 hours ago Exited (1) 9 minutes ago c1

e9fe479f065c 037b "/bin/bash" 23 hours ago Up 21 hours center

接下来开启各个子节点，然后开启各个子节点中的python程序：

[root@localhost /]# docker start c1

c1

[root@localhost /]# docker attach c1

root@ebe6c483c9f9:/#

root@ebe6c483c9f9:/# python mycrawl.py

[root@localhost /]# docker attach c2

root@848dcb31a3b3:/#

root@848dcb31a3b3:/# python mycrawl.py

root@localhost /]# docker attach c3

root@3097f39af25d:/#

root@3097f39af25d:/# python mycrawl.py

[root@localhost /]# docker attach c4

root@f4d716e1d6a1:/#

root@f4d716e1d6a1:/# python mycrawl.py

可以通过主服务器查看到其结果：

root@e9fe479f065c:/# redis-cli

127.0.0.1:6379> hgetall rst

4243) "21927"

4244) "\xe5\x86\xb7\xe8\xa1\x80\xe6\xb2\xb3"

4245) "21888"

4246) "\xe7\x8b\xac\xe6\xa2\xa6\xe4\xbb\x99"

可以看到已经爬到了4246条。

此时可以将主程序所有爬取的数据写入mysql数据库，分布式爬虫功能实现完成。