上节课我们的分布式爬虫部署完成并可以成功运行了，但是有个环节非常烦琐，那就是代码部署。

我们设想下面的几个场景：

- 如果采用上传文件的方式部署代码，我们首先需要将代码压缩，然后采用 SFTP 或 FTP 的方式将文件上传到服务器，之后再连接服务器将文件解压，每个服务器都需要这样配置。
- 如果采用 Git 同步的方式部署代码，我们可以先把代码 Push 到某个 Git 仓库里，然后再远程连接各台主机执行 Pull 操作，同步代码，每个服务器同样需要做一次操作。

如果代码突然有更新，那我们必须更新每个服务器，而且万一哪台主机的版本没控制好，还可能会影响整体的分布式爬取状况。

所以我们需要一个更方便的工具来部署 Scrapy 项目，如果可以省去一遍遍逐个登录服务器部署的操作，那将会方便很多。

本节我们就来看看提供分布式部署的工具 Scrapyd。

了解 Scrapyd

接下来，我们就来深入地了解 Scrapyd，Scrapyd 是一个运行 Scrapy 爬虫的服务程序，它提供一系列 HTTP 接口来帮助我们部署、启动、停止、删除爬虫程序。Scrapyd 支持版本管理，同时还可以管理多个爬虫任务，利用它我们可以非常方便地完成 Scrapy 爬虫项目的部署任务调度。

准备工作

首先我们需要安装 scrapyd，一般我们部署的服务器是 Linux，所以这里以 Linux 为例来进行说明。

这里推荐使用 pip 安装，命令如下：

```
pip3 install scrapyd
```

另外为了我们编写的项目能够运行成功，还需要安装项目本身的环境依赖，如上一节的项目需要依赖 Scrapy、Scrapy-Redis、Gerapy-Pyppeteer 等库，也需要在服务器上安装，否则会出现部署失败的问题。
安装完毕之后，需要新建一个配置文件 /etc/scrapyd/scrapyd.conf，Scrapyd 在运行的时候会读取此配置文件。

在 Scrapyd 1.2 版本之后，不会自动创建该文件，需要我们自行添加。首先，执行如下命令新建文件：

```
sudo mkdir /etc/scrapyd
sudo vi /etc/scrapyd/scrapyd.conf
```

接着写入如下内容：

```
[scrapyd]
eggs_dir    = eggs
logs_dir    = logs
items_dir   =
jobs_to_keep = 5
dbs_dir     = dbs
max_proc    = 0
max_proc_per_cpu = 10
finished_to_keep = 100
poll_interval = 5.0
bind_address = 0.0.0.0
http_port   = 6800
debug       = off
runner      = scrapyd.runner
application = scrapyd.app.application
launcher    = scrapyd.launcher.Launcher
webroot     = scrapyd.website.Root

[services]
schedule.json     = scrapyd.webservice.Schedule
cancel.json       = scrapyd.webservice.Cancel
addversion.json   = scrapyd.webservice.AddVersion
listprojects.json = scrapyd.webservice.ListProjects
listversions.json = scrapyd.webservice.ListVersions
listspiders.json  = scrapyd.webservice.ListSpiders
delproject.json   = scrapyd.webservice.DeleteProject
delversion.json   = scrapyd.webservice.DeleteVersion
listjobs.json     = scrapyd.webservice.ListJobs
daemonstatus.json = scrapyd.webservice.DaemonStatus
```

配置文件的内容可以参见官方文档 https://scrapyd.readthedocs.io/en/stable/config.html#example-configuration-file。这里的配置文件有所修改，其中之一是 max_proc_per_cpu 官方默认为 4，即一台主机每个 CPU 最多运行 4 个 Scrapy 任务，在此提高为 10。另外一个是 bind_address，默认为本地 127.0.0.1，在此修改为 0.0.0.0，以使外网可以访问。

Scrapyd 是一个纯 Python 项目，这里可以直接调用它来运行。为了使程序一直在后台运行，Linux 和 Mac 可以使用如下命令：

```
(scrapyd > /dev/null &)
```

这样 Scrapyd 就可以在后台持续运行了，控制台输出直接忽略。当然，如果想记录输出日志，可以修改输出目标，如下所示：

```
(scrapyd> ~/scrapyd.log &)
```

此时会将 Scrapyd 的运行结果输出到~/scrapyd.log 文件中。当然也可以使用 screen、tmux、supervisor 等工具来实现进程守护。

安装并运行了 Scrapyd 之后，我们就可以访问服务器的 6800 端口看到一个 WebUI 页面了，例如我的服务器地址为 120.27.34.25，在上面安装好了 Scrapyd 并成功运行，那么我就可以在本地的浏览器中打开：http://120.27.34.25:6800，就可以看到 Scrapyd 的首页，这里请自行替换成你的服务器地址查看即可，如图所示：

Scrapyd

Available projects: **zhihusite, zhihuuser, default, sinaapi**

- Jobs
- Logs
- Documentation

How to schedule a spider?

To schedule a spider you need to use the API (this web UI is only for monitoring)

Example using curl:

curl http://localhost:6800/schedule.json -d project=default -d spider=somespider

For more information about the API, see the Scrapyd documentation

如果可以成功访问到此页面，那么证明 Scrapyd 配置就没有问题了。

Scrapyd 的功能

Scrapyd 提供了一系列 HTTP 接口来实现各种操作，在这里我们可以将接口的功能梳理一下，以 Scrapyd 所在的 IP 为 120.27.34.25 为例进行讲解。

daemonstatus.json

这个接口负责查看 Scrapyd 当前服务和任务的状态，我们可以用 curl 命令来请求这个接口，命令如下：

```
curl http://139.217.26.30:6800/daemonstatus.json
```

这样我们就会得到如下结果：

```
{"status": "ok", "finished": 90, "running": 9, "node_name": "datacrawl-vm", "pending": 0}
```

返回结果是 Json 字符串，status 是当前运行状态， finished 代表当前已经完成的 Scrapy 任务，running 代表正在运行的 Scrapy 任务，pending 代表等待被调度的 Scrapyd 任务，node_name 就是主机的名称。

addversion.json

这个接口主要是用来部署 Scrapy 项目，在部署的时候我们需要首先将项目打包成 Egg 文件，然后传入项目名称和部署版本。

我们可以用如下的方式实现项目部署：

```
curl http://120.27.34.25:6800/addversion.json -F project=wenbo -F version=first -F egg=@weibo.egg
```

在这里 -F 即代表添加一个参数，同时我们还需要将项目打包成 Egg 文件放到本地。
这样发出请求之后我们可以得到如下结果：

```
{"status": "ok", "spiders": 3}
```

这个结果表明部署成功，并且其中包含的 Spider 的数量为 3。此方法部署可能比较烦琐，在后面我会介绍更方便的工具来实现项目的部署。

schedule.json

这个接口负责调度已部署好的 Scrapy 项目运行。我们可以通过如下接口实现任务调度：

```
curl http://120.27.34.25:6800/schedule.json -d project=weibo -d spider=weibocn
```

在这里需要传入两个参数，project 即 Scrapy 项目名称，spider 即 Spider 名称。返回结果如下：

```
{"status": "ok", "jobid": "6487ec79947edab326d6db28a2d86511e8247444"}
```

status 代表 Scrapy 项目启动情况，jobid 代表当前正在运行的爬取任务代号。

cancel.json

这个接口可以用来取消某个爬取任务，如果这个任务是 pending 状态，那么它将会被移除，如果这个任务是 running 状态，那么它将会被终止。

我们可以用下面的命令来取消任务的运行：

```
curl http://120.27.34.25:6800/cancel.json -d project=weibo -d job=6487ec79947edab326d6db28a2d86511e8247444
```

在这里需要传入两个参数，project 即项目名称，job 即爬取任务代号。返回结果如下：

```
{"status": "ok", "prevstate": "running"}
```

status 代表请求执行情况，prevstate 代表之前的运行状态。

listprojects.json

这个接口用来列出部署到 Scrapyd 服务上的所有项目描述。我们可以用下面的命令来获取 Scrapyd 服务器上的所有项目描述：

```
curl http://120.27.34.25:6800/listprojects.json
```

这里不需要传入任何参数。返回结果如下：

```
{"status": "ok", "projects": ["weibo", "zhihu"]}
```

status 代表请求执行情况，projects 是项目名称列表。

listversions.json

这个接口用来获取某个项目的所有版本号，版本号是按序排列的，最后一个条目是最新的版本号。

我们可以用如下命令来获取项目的版本号：

```
curl http://120.27.34.25:6800/listversions.json?project=weibo
```

在这里需要一个参数 project，就是项目的名称。返回结果如下：

```
{"status": "ok", "versions": ["v1", "v2"]}
```

status 代表请求执行情况，versions 是版本号列表。

listspiders.json

这个接口用来获取某个项目最新的一个版本的所有 Spider 名称。我们可以用如下命令来获取项目的 Spider 名称：

```
curl http://120.27.34.25:6800/listspiders.json?project=weibo
```

在这里需要一个参数 project，就是项目的名称。返回结果如下：

```
{"status": "ok", "spiders": ["weibocn"]}
```

status 代表请求执行情况，spiders 是 Spider 名称列表。

listjobs.json

这个接口用来获取某个项目当前运行的所有任务详情。我们可以用如下命令来获取所有任务详情：

```
curl http://120.27.34.25:6800/listjobs.json?project=weibo
```

在这里需要一个参数 project，就是项目的名称。返回结果如下：

```
{"status": "ok",
 "pending": [{"id": "78391cc0fcaf11e1b0090800272a6d06", "spider": "weibocn"}],
 "running": [{"id": "422e608f9f28cef127b3d5ef93fe9399", "spider": "weibocn", "start_time": "2017-07-12 10:14:03.594664"}],
 "finished": [{"id": "2f16646cfcaf11e1b0090800272a6d06", "spider": "weibocn", "start_time": "2017-07-12 10:14:03.594664", "end_time": "2017-07-12 10:24:03.594664"}]}
```

status 代表请求执行情况，pendings 代表当前正在等待的任务，running 代表当前正在运行的任务，finished 代表已经完成的任务。

delversion.json

这个接口用来删除项目的某个版本。我们可以用如下命令来删除项目版本：

```
curl http://120.27.34.25:6800/delversion.json -d project=weibo -d version=v1
```

在这里需要一个参数 project，就是项目的名称，还需要一个参数 version，就是项目的版本。返回结果如下：

```
{"status": "ok"}
```

status 代表请求执行情况，这样就代表删除成功了。

delproject.json

这个接口用来删除某个项目。我们可以用如下命令来删除某个项目：

```
curl http://120.27.34.25:6800/delproject.json -d project=weibo
```

在这里需要一个参数 project，就是项目的名称。返回结果如下：

```
{"status": "ok"}
```

status 代表请求执行情况，这样就代表删除成功了。
以上就是 Scrapyd 所有的接口，我们可以直接请求 HTTP 接口即可控制项目的部署、启动、运行等操作。

ScrapydAPI 的使用

以上的这些接口可能使用起来还不是很方便，没关系，还有一个 ScrapydAPI 库对这些接口又做了一层封装，其安装方式如下：

```
pip3 install python-scrapyd-api
```

下面我们来看下 ScrapydAPI 的使用方法，其实核心原理和 HTTP 接口请求方式并无二致，只不过用 Python 封装后使用更加便捷。
我们可以用如下方式建立一个 ScrapydAPI 对象：

```
from scrapyd_api import ScrapydAPI
scrapyd = ScrapydAPI('http://120.27.34.25:6800')
```

然后就可以通过调用它的方法来实现对应接口的操作了，例如部署的操作可以使用如下方式：

```
egg = open('weibo.egg', 'rb')
scrapyd.add_version('weibo', 'v1', egg)
```

这样我们就可以将项目打包为 Egg 文件，然后把本地打包的 Egg 项目部署到远程 Scrapyd 了。
另外 ScrapydAPI 还实现了所有 Scrapyd 提供的 API 接口，名称都是相同的，参数也是相同的。

例如我们可以调用 list_projects 方法即可列出 Scrapyd 中所有已部署的项目：

```
scrapyd.list_projects()
['weibo', 'zhihu']
```

另外还有其他的方法在此不再一一列举了，名称和参数都是相同的，更加详细的操作可以参考其官方文档：http://python-scrapyd-api.readthedocs.io/。
我们可以通过它来部署项目，并通过 HTTP 接口来控制任务的运行，不过这里有一个不方便的地方就是部署过程，首先它需要打包 Egg 文件然后再上传，还是比较烦琐的，这里再介绍另外一个工具 Scrapyd-Client。

Scrapyd-Client 部署

Scrapyd-Client 为了方便 Scrapy 项目的部署，提供两个功能：

- 将项目打包成 Egg 文件。
- 将打包生成的 Egg 文件通过 addversion.json 接口部署到 Scrapyd 上。

也就是说，Scrapyd-Client 帮我们把部署全部实现了，我们不需要再去关心 Egg 文件是怎样生成的，也不需要再去读 Egg 文件并请求接口上传了，这一切的操作只需要执行一个命令即可一键部署。

要部署 Scrapy 项目，我们首先需要修改一下项目的配置文件，例如我们之前写的 Scrapy 项目，在项目的第一层会有一个 scrapy.cfg 文件，它的内容如下：

```
[settings]
default = scrapypyppeteer.settings

[deploy]
#url = http://localhost:6800/
project = scrapypyppeteer
```

在这里我们需要配置 deploy，例如我们要将项目部署到 120.27.34.25 的 Scrapyd 上，就需要修改为如下内容：

```
[deploy]
url = http://120.27.34.25:6800/
project = scrapypyppeteer
```

这样我们再在 scrapy.cfg 文件所在路径执行如下命令：

```
scrapyd-deploy
```

运行结果如下：

```
Packing version 1501682277
Deploying to project "weibo" in http://120.27.34.25:6800/addversion.json
Server response (200):
{"status": "ok", "spiders": 1, "node_name": "datacrawl-vm", "project": "scrapypyppeteer", "version": "1501682277"}
```

返回这样的结果就代表部署成功了。

我们也可以指定项目版本，如果不指定的话默认为当前时间戳，指定的话通过 version 参数传递即可，例如：

```
scrapyd-deploy --version 201707131455
```

值得注意的是在 Python3 的 Scrapyd 1.2.0 版本中我们不要指定版本号为带字母的字符串，需要为纯数字，否则可能会出现报错。

另外如果我们有多台主机，我们可以配置各台主机的别名，例如可以修改配置文件为：

```
[deploy:vm1]
url = http://120.27.34.24:6800/
project = scrapypyppeteer

[deploy:vm2]
url = http://139.217.26.30:6800/
project = scrapypyppeteer
```

有多台主机的话就在此统一配置，一台主机对应一组配置，在 deploy 后面加上主机的别名即可，这样如果我们想将项目部署到 IP 为 139.217.26.30 的 vm2 主机，我们只需要执行如下命令：

```
scrapyd-deploy vm2
```

这样我们就可以将项目部署到名称为 vm2 的主机上了。
如此一来，如果我们有多台主机，我们只需要在 scrapy.cfg 文件中配置好各台主机的 Scrapyd 地址，然后调用 scrapyd-deploy 命令加主机名称即可实现部署，非常方便。

如果 Scrapyd 设置了访问限制的话，我们可以在配置文件中加入用户名和密码的配置，同时修改端口，修改成 Nginx 代理端口，如在模块一我们使用的是 6801，那么这里就需要改成 6801，修改如下：

```
[deploy:vm1]
url = http://120.27.34.24:6801/
project = scrapypyppeteer
username = admin
password = admin

[deploy:vm2]
url = http://139.217.26.30:6801/
project = scrapypyppeteer
username = germey
password = germey
```

这样通过加入 username 和 password 字段，我们就可以在部署时自动进行 Auth 验证，然后成功实现部署。

总结

以上我们介绍了 Scrapyd、Scrapyd-API、Scrapyd-Client 的部署方式，希望你可以多多尝试。