目 录

[1 Locust原理 1](#_Toc3153)

[2主要属性 1](#_Toc25817)

[3 Task\_Set类 1](#_Toc23459)

[4基本测试 2](#_Toc32274)

[5复杂测试 3](#_Toc10896)

[5.1接口关联 3](#_Toc12580)

[5.2参数化 3](#_Toc25326)

[5.3检查点 4](#_Toc32755)

[6总结 4](#_Toc18664)

1 Locust原理

 Locust是一个用于可扩展的，分布式的，性能测试的，开源的，用Python编写框架/工具。主要思想就是模拟一群用户将访问你的网站。每个用户的行为由你编写的python代码定义，同时可以从Web界面中实时观察到用户的行为。

Locust完全是事件驱动的，因此在单台机器上能够支持几千并发用户访问。与其它许多基于事件的应用相比，Locust并不使用回调，而是使用gevent，而gevent是基于协程的，可以用同步的方式来编写异步执行的代码。每个用户实际上运行在自己的greenlet中。

特点：

①用简单python语言编写测试脚本，非常简单轻便。不需要笨重的UI和臃肿的xml代码，基于协同而非回调。

②分布式的，可扩展性的，可模拟上百万用户。Locust支持多机器的性能测试，每台机器可以模拟上千用户，当然这可以控制的，

③Locust有一个整洁的HTML+JS的用户界面，实时显示相关测试细节。由于用户界面是基于网络的，它是跨平台的和容易扩展。

④可以测试任何系统，尽管Locust是基于网站的，但它几乎可以测试任何系统，只要你写一个客户端。

在Locust类中，具有一个client属性，它对应着虚拟用户作为客户端所具备的请求能力，也就是我们常说的请求方法。通常情况下，我们不会直接使用Locust类，因为其client属性没有绑定任何方法。因此在使用Locust时，需要先继承Locust类，然后在继承子类中的client属性中绑定客户端的实现类。

对于常见的HTTP(S)协议，Locust已经实现了HttpLocust类，其client属性绑定了HttpSession类，而HttpSession又继承自requests.Session。因此在测试HTTP(S)的Locust脚本中，我们可以通过client属性来使用Python requests库的所有方法，包括GET/POST/HEAD/PUT/DELETE/PATCH等，调用方式也与requests完全一致。另外，由于requests.Session的使用，因此client的方法调用之间就自动具有了状态记忆的功能。常见的场景就是，在登录系统后可以维持登录状态的Session，从而后续HTTP请求操作都能带上登录态。

2主要属性

task\_set: 指向一个TaskSet类，TaskSet类定义了用户的任务信息，该属性为必填；

max\_wait/min\_wait: 每个用户执行两个任务间隔时间的上下限（毫秒），具体数值在上下限中随机取值，若不指定则默认间隔时间固定为1秒；

host：被测系统的host，当在终端中启动locust时没有指定--host参数时才会用到，例如locust -f locust.py --host=https://www.baidu.com（我们可以直接在性能脚本指定host）；

weight：同时运行多个Locust类时会用到，用于控制不同类型任务的执行权重。

3 Task\_Set类

以下面例子说明，新建一个py文件，文件名随意

from locust import HttpLocust, TaskSet

def login(l):

    l.client.post("/login", {"username":"ellen\_key", "password":"education"})

def test\_01(l):

    l.client.get("/")

def test\_02(l):

    l.client.get("/profile")

class UserBehavior(TaskSet):

    tasks = {test\_01:2, test\_02:1}

    def on\_start(self):

        login(self)

class WebsiteUser(HttpLocust):

    task\_set = UserBehavior

    min\_wait=5000

    max\_wait=9000

或者

from locust import HttpLocust, TaskSet, task

class UserBehavior(TaskSet):

    def on\_start(self):

        """ on\_start is called when a Locust start before any task is scheduled """

        self.login()

    def login(self):

        self.client.post("/login",{"username":"ellen\_key", "password":"education"})

    @task(2)

    def test\_01(self):

        self.client.get("/")

    @task(1)

    def test\_01(self):

        self.client.get("/profile")

class WebsiteUser(HttpLocust):

    task\_set = UserBehavior

    min\_wait=5000

    max\_wait=9000

性能测试工具要模拟用户的业务操作，就需要通过脚本模拟用户的行为。Task\_Set类控制脚本行为，向服务器发起压力。

TaskSet类实现了虚拟用户所执行任务的调度算法，包括规划任务执行顺序（schedule\_task）、挑选下一个任务（execute\_next\_task）、执行任务（execute\_task）、休眠等待（wait）、中断控制（interrupt）等等。在此基础上，我们就可以在TaskSet子类中采用非常简洁的方式来描述虚拟用户的业务测试场景，对虚拟用户的所有行为（任务）进行组织和描述，并可以对不同任务的权重进行配置。

在TaskSet子类中定义任务信息时，可以采取两种方式，@task装饰器和tasks属性。定义了两个类，一个User Behavior类，继承了TaskSet类，用于定义测试任务的，给属性tasks增加了两个任务，index函数和profile函数，这些人物被执行，然后返回执行时间，正常情况下，是在下面的最小时间和最大时间之间，从on\_start 开始，就像java的main函数一样，是任务开口，然后随机的挑选任务，通过client（相当于一个Httpsession）的方法执行http请求，但是会按照设置的比率来执行。Tasks属性把上面定义的函数变成任务，它是一个dict类型。相当于java的map类型。

一个WebsiteUser类，继承了HttpLocust类，这个类用于代表用户，生成一个实例，为每个每个模拟用户，发送http请求和设置测试参数，Task\_set属性，它是唯一必须要有的，它指向Task Set类，定义用户的行为，请求等待最小时间min\_wait和请求等待最大时间max\_wait属性，单位是毫秒。，weight属性指定用户的执行的比率，host属性是测试的应用的网址。

注意最大时间和最小时间属性可以在locust类中定义，也可以在task set类中定义，完全是一样的

* @task(1)和@task(2)表明@task(2)所装饰函数的执行频率是@task(1)所装饰函数的两倍，同样tasks = {test\_01:1, test\_02:2}效果是一样的。
* @task和@task表明@task所装饰函数的执行频率和@task所装饰函数的相同，同样tasks = {test\_01, test\_02}效果是一样的（不指定权重）。

TaskSet还可以嵌套：参考下面的代码

第一种：

class ForumPage(TaskSet):

    @task(20)

    def read\_thread(self):

        pass

    @task(1)

    def new\_thread(self):

        pass

    @task(5)

    def stop(self):

        self.interrupt()

class UserBehaviour(TaskSet):

    tasks = {ForumPage:10}

    @task

    def index(self):

        pass

第一种需要注意的是interrupt这个函数，如果没有这个函数locust就会一直执行formpage这个任务，只有通过这个函数，才能跳出来。执行formpage之外的函数。

第二种：

class MyTaskSet(TaskSet):

    @task

    class SubTaskSet(TaskSet):

        @task

        def my\_task(self):

            pass

4基本测试

定义WebsiteTasks类做为继承task\_set类的子类。在TaskSet子类中除了定义任务信息，还有一个是经常用到的，那就是on\_start函数。这个和Unittest中的TestCase的setUP功能类似，在正式执行测试前执行一次，主要用于完成一些初始化的工作。例如，当测试某个搜索功能，而该搜索功能又要求必须为登录态的时候，就可以先在on\_start中进行登录操作；前面也提到，HttpLocust使用到了requests.Session，因此后续所有任务执行过程中就都具有登录态了。

1. 先执行WebsiteTasks中的on\_start（只执行一次），作为初始化，例如执行登录等只需要执行一次的动作；
2. 从WebsiteTasks中随机挑选（如果定义了任务间的权重关系，那么就是按照权重关系随机挑选）一个任务执行；
3. 根据Locust类中min\_wait和max\_wait定义的间隔时间范围（如果TaskSet类中也定义了min\_wait或者max\_wait，以TaskSet中的优先），在时间范围中随机取一个值，休眠等待；
4. 重复2~3步骤，直至测试任务终止。

5复杂测试

掌握了HttpLocust和TaskSet，我们就基本具备了编写测试脚本的能力，但是面对较复杂的测试场景，可能有的朋友还是会感觉无从下手；例如，很多时候脚本需要做关联或参数化处理，这些在Jmeter中集成的功能，换到Locust中就不知道怎么实现了。可能也是这方面的原因，造成很多测试人员都感觉难以将Locust应用到实际的性能测试工作当中，性能测试脚本无非上就是接口关联、参数化、检查点、集合点等四个方面。

5.1接口关联

关于接口关联，在某些请求中，需要将之前接口的response中的一部分值做为下一个接口的入参，因此在构造请求时需要先从之前请求的response中提取出所需的参数，常见场景就是session\_id，token等。针对这种情况，Jmeter可以通过正则表达式匹配出前一个接口请求的response中所需要的参数，例如：\d+，\.\*分别是匹配任意数字和任意字符，根据左右边界或其它特征定位到参数值并将其保存到参数变量，然后在后续请求中使用该参数。采用同样的思想，我们在Locust脚本中也完全可以实现同样的功能，通过官方库函数re.search就能实现所有需求，如果不采用re模块用正则表达式匹配，因接口所返回值基本是json格式，故可以根据相应的“key”获取对应的参数赋值给变量。针对html页面，我们也可以采用lxml库，采用etree.HTML(html).xpath来实现元素定位。

5.2参数化

参数化这一项，主要用在测试数据方面，可以概括为三种类型：

* 循环取数据，数据可重复使用：模拟多个用户并发搜索商品，总共有100个商品名，每个虚拟用户都会依次循环加载这100个商品；
* 保证并发测试数据唯一性，不循环取数据：模拟多个用户并发注册账号，总共有100个账号，要求注册账号不重复，注册完毕后结束测试；
* 保证并发测试数据唯一性且循环取数据：模拟多用户并发登录账号，总共有100个账号，并发登录账号不相同，但数据可循环使用。

以上三种类型基本上可以覆盖我们日常性能测试工作中的所有参数化场景。Jmeter中集成的参数化模块，可以直接配置参数化策略。例如：可以读取csv和txt文本文件中的参数进行参数化设置，Locust中我们同样可以采用读取文本文件或csv/Excel文件的方式实现参数化，在WebsiteUser定义一个数据集，然后所有虚拟用户在WebsiteTasks中就可以共享该数据集了。如果不要求数据唯一性，数据集选择list数据结构，从头到尾循环遍历即可；如果要求数据唯一性，数据集选择queue数据结构，取数据时进行queue.get()操作即可，并且这也不会循环取数据；至于涉及到需要循环取数据的情况，那也简单，每次取完数据后再将数据插入到队尾即可，queue.put\_nowait(data)。

5.3检查点

在Locust脚本中，处理与unittest类似，只需要对响应的内容关键字进行assert xxx in response操作即可。

6总结

Locust是一款基于python的开源的性能测试工具，现在接口自动化基本采用python，如果接口自动化脚本是你自己编写的，那么locust是一款非常适合你的性能测试工具。但是locust要求一定的代码能力，故选用Jmeter还是locust进行性能测试请根据实际情况进行选择。