

性能测试

The background features a dark blue gradient with flowing, wavy teal lines that create a sense of motion. Overlaid on these are several curved lines of small, glowing teal dots, resembling a digital grid or data stream.

课程目标

通过本课程的学习，您将可以

- 了解性能测试
- 熟悉Locust性能测试框架
- 了解caliper区块链性能测试

目录

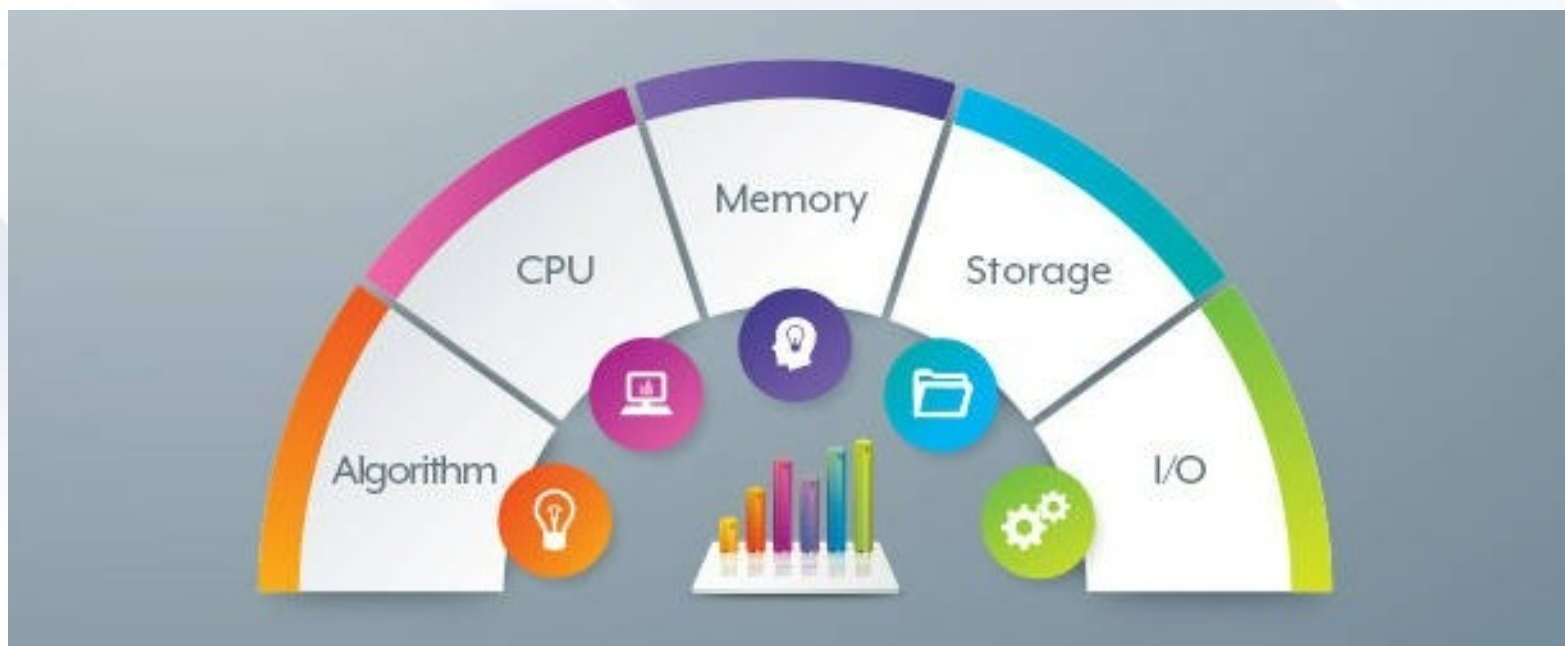
- 第一章 性能测试介绍
- 第二章 Locust性能测试框架
- 第三章 Caliper性能测试

第一章 性能测试介绍

- 1.1 什么是性能测试
- 1.2 为什么要进行性能测试
- 1.3 如何进行性能测试
- 1.3 小结与思考

1.1 什么是性能测试

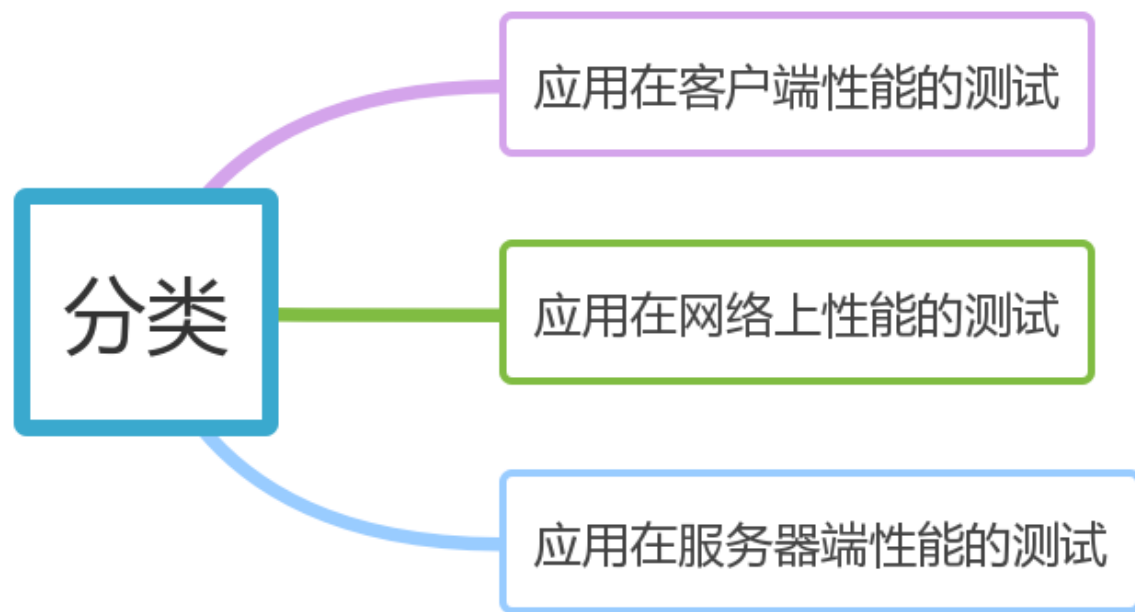
性能测试是通过自动化的测试工具模拟多种**正常**、**峰值**以及**异常负载**条件来对系统或应用的**各项性能指标**进行测试。下图中展示了常见的系统性能指标：Algorithm（指代系统或应用的算法）、CPU（中央处理器）、Memory（内存）、Storage（储存）以及I/O（Input/Output，输入/输出）。



系统与应用中常见的各项性能

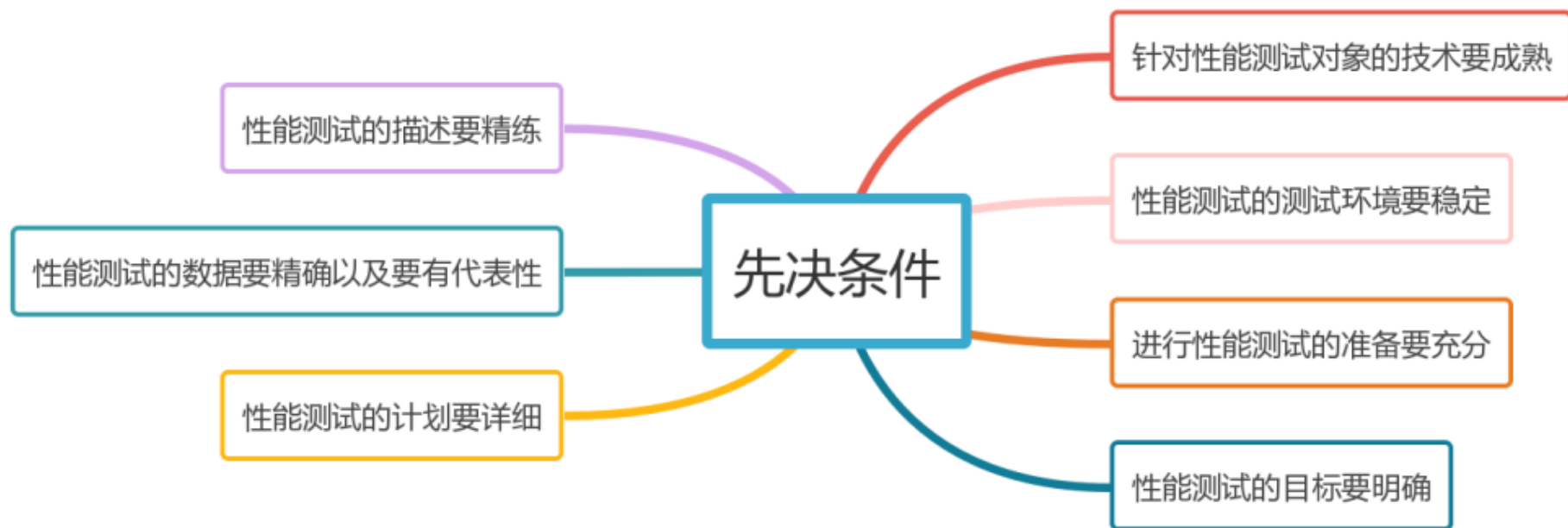
1.1 什么是性能测试

性能测试可以分为三个类别：



1.1 什么是性能测试

性能测试的先决条件有以下几点：



1.2 为什么要进行性能测试

进行性能测试的原因主要有以下三点。

进行性能测试的原因

评估系统的能力

验证系统的稳定性和可靠性

识别体系中的弱点

1.3 如何进行性能测试

进行性能测试会经历以下几个主要阶段：



性能测试的过程

1.3 如何进行性能测试

1、需求分析

在开启一个性能测试前，相关人员需要明确性能测试的需求：

产品/项目经理根据预期情况给出了相应需求点，例如某商家打折促销，需要公司提供的销售平台能够支持80个用户同时下单。

测试人员要充分了解需求，例如本次测试的业务有哪些，优先级顺序是怎样的，还需要弄清楚，预期通过的指标有哪些，例如活动期间，请求的平均响应时间不超过400ms；请求错误率不超过0.03%；预期用户大概20000，最大并发用户数的数量以及访问比较频繁的业务点等等。

开发人员需要根据自身相关经验给出相关测试建议，例如本次活动主要压力点、注意事项等等。

1.3 如何进行性能测试

2、测试方案

性能测试方案需包含测试需求分析、测试对象分析、测试重点分析、测试环境分析、测试场景构建几个方面：

- 1、**测试需求分析**：测试中涉及的性能指标的定义
- 2、**测试对象分析**：测试中涉及的业务及系统内部模块的定义
- 3、**测试重点分析**：测试执行策略、测试监控策略、测试风险的定义
- 4、**测试环境分析**：测试中涉及到的服务器资源、测试软件、测试数据、测试桩定义
- 5、**测试场景构建**：从性能负载、接口、疲劳角度定义测试场景

1.3 如何进行性能测试

3、测试准备

在制定了测试方案之后，需要着手准备相关的测试环境和工具，主要包含以下几个方面：

- 1、搭建一套比较接近生产环境的性能测试环境，这个环境需要独立部署，不能和功能测试环境和开发环境共用，这样性能测试结果才会比较准确。
- 2、选择性能测试工具，进行测试工具的安装和调试。
- 3、编写性能测试脚本，实现性能测试场景的逻辑。
- 4、准备性能测试数据，使业务能正常运行。

1.3 如何进行性能测试

4、测试执行

准备好相应的测试环境、工具以及数据等后，便可以开始执行测试。

5、测试结果获取

执行测试后将获得测试结果。

6、结果验证

验证测试结果，若验证通过则可以生成测试报告提交给产品/项目经理，若不通过则需对有缺陷的地方进行优化，在优化后再一次执行测试（这一步被称为回归测试），若测试通过则生成测试报告。

7、测试报告生成

测试完成的最终产出，提交给产品/项目经理以及相关负责人等查阅。测试人员在编写报告时需要注意如果报告是发给领导的，需尽量避免测试术语，用更容易理解的语句来描述。

1.4 小结与思考

本小节我们简单的了解了性能测试以及进行性能测试的三个原因。

问题思考

请同学们调研并思考性能测试可以细分为哪几类测试，并总结这些类别的特点。



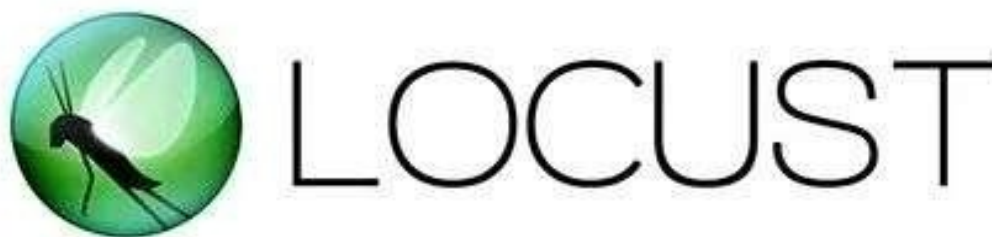
第二章 Locust性能测试框架

- 2.1 Locust简介
- 2.2 Locust的优点
- 2.3 Locust界面介绍
- 2.4 Python语法
- 2.5 Locust性能测试案例
- 2.6 小结与思考

2.1 Locust简介

Locust是一个简单易用的分布式用户**负载测试**工具。它用于web站点(或其他系统)的负载测试，并计算一个系统可以处理多少**并发用户**。

Locust官方文档对该工具的运行方式做了一个很形象的比喻：在测试中，一群locusts(蝗虫)会“攻击”测试者指定的网站。每个蝗虫的行为由测试者定义，而整个“攻击”过程由网站实时监控。这将帮助测试者在允许实际用户使用之前测试并确定代码中的瓶颈。



2.2 Locust的优点

locust作为一款很受欢迎的性能测试工具，具有以下几个特点：



2.3 Locust界面介绍

启动界面

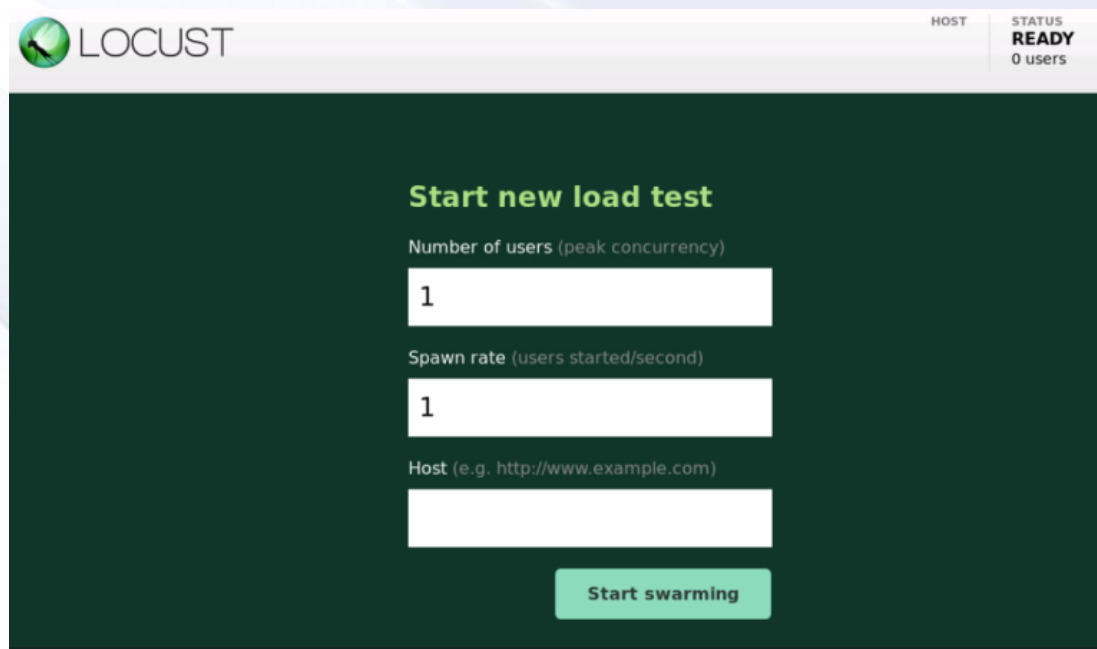
启动界面如下图所示。

Number of users (peak concurrency)：设置模拟的用户总数

Spawn rate (users started/second)：每秒启动的虚拟用户数

Host：服务器地址

Start swarming：执行locust脚本



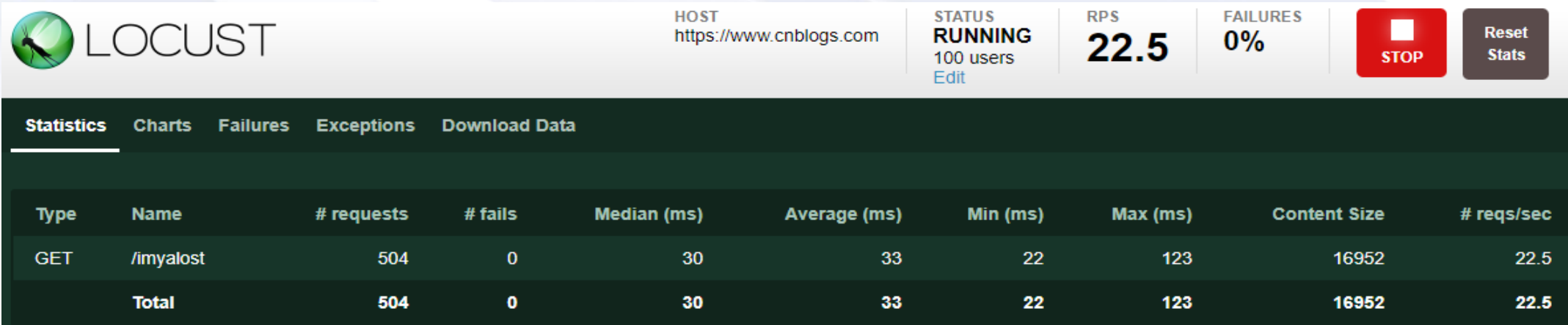
The screenshot shows the Locust web interface. At the top, there's a header with the Locust logo and the word 'LOCUST'. On the right, there's a status bar showing 'HOST' and 'STATUS READY 0 users'. The main area is a dark green box with the title 'Start new load test'. Below the title, there are three input fields: 'Number of users (peak concurrency)' with the value '1', 'Spawn rate (users started/second)' with the value '1', and 'Host (e.g. http://www.example.com)' which is empty. At the bottom right of the green box is a green button labeled 'Start swarming'.

2.3 Locust界面介绍

测试结果界面

测试结果界面默认在Statistics界面，如下图所示。

- Type**：请求类型，即接口的请求方法
- Name**：请求路径
- requests**：当前已完成的请求数量
- fails**：当前失败的数量
- Median**：响应时间的中间值，即50%的响应时间在这个数值范围内，单位为毫秒

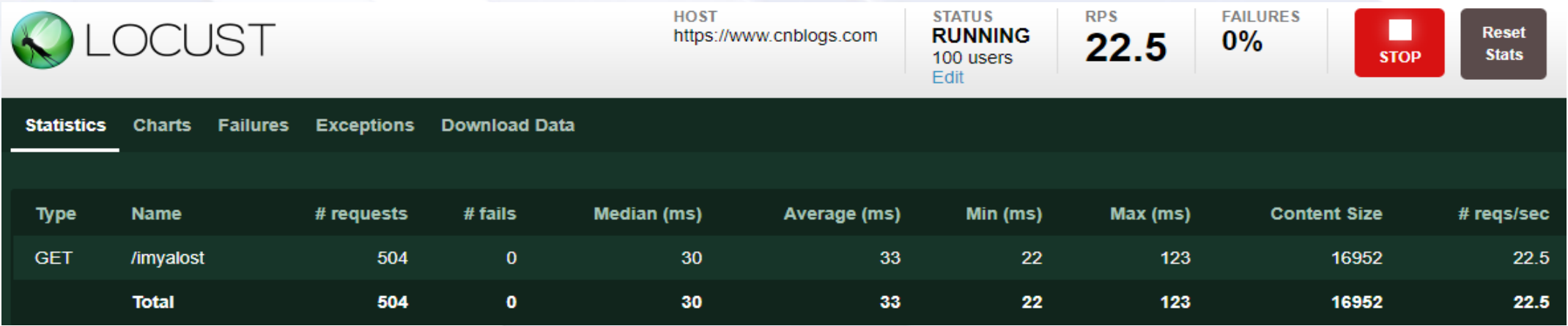


2.3 Locust界面介绍

测试结果界面

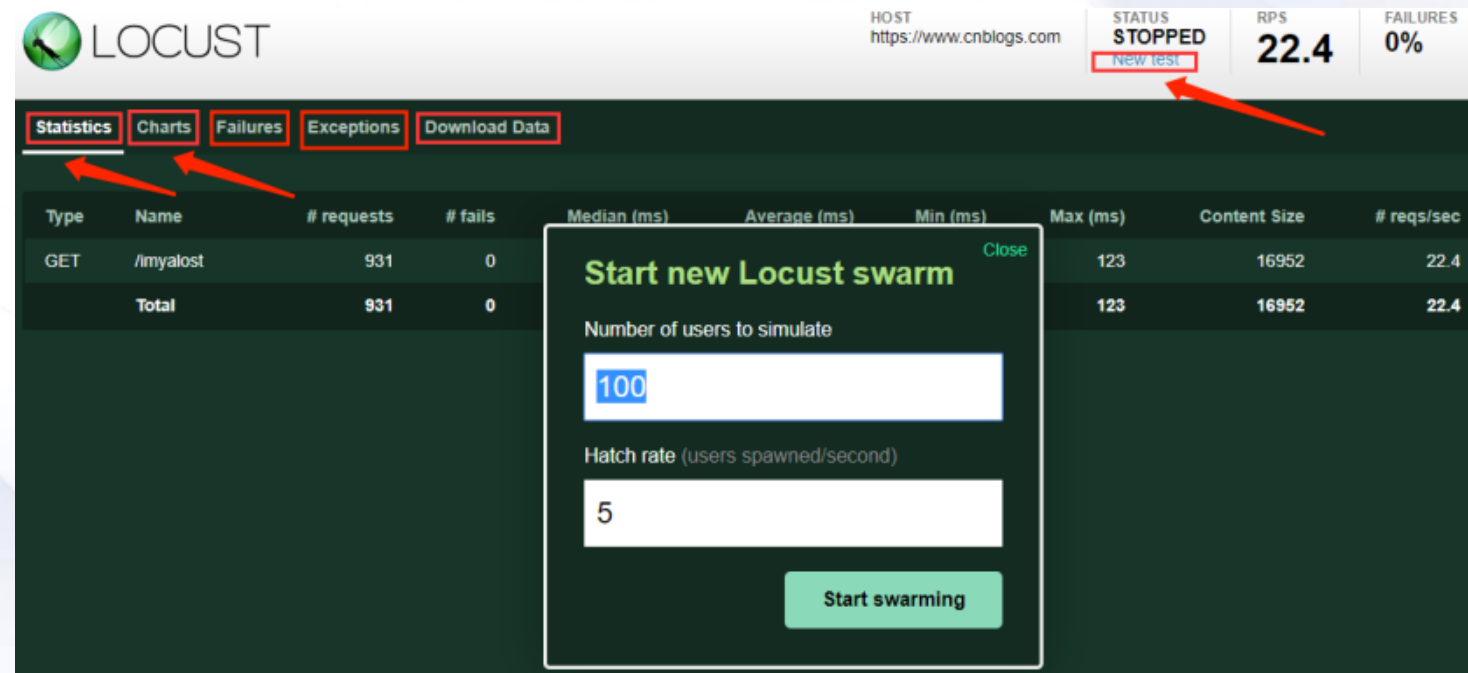
测试结果界面默认在Statistics模块，如下图所示。

- Average**：平均响应时间，单位为毫秒
- Min**：最小响应时间，单位为毫秒
- Max**：最大响应时间，单位为毫秒
- Content Size**：所有请求的数据量，单位为字节
- reqs/sec**：每秒钟处理请求的数量



2.3 Locust界面介绍

点击New test按钮可对模拟的总虚拟用户数和每秒启动的虚拟用户数进行编辑，如下图所示。



各模块说明

Statistics：类似于jmeter中Listen的聚合报告；

Charts：测试结果变化趋势的曲线展示图，分别为每秒完成的请求数（RPS）、响应时间、不同时间的虚拟用户数；

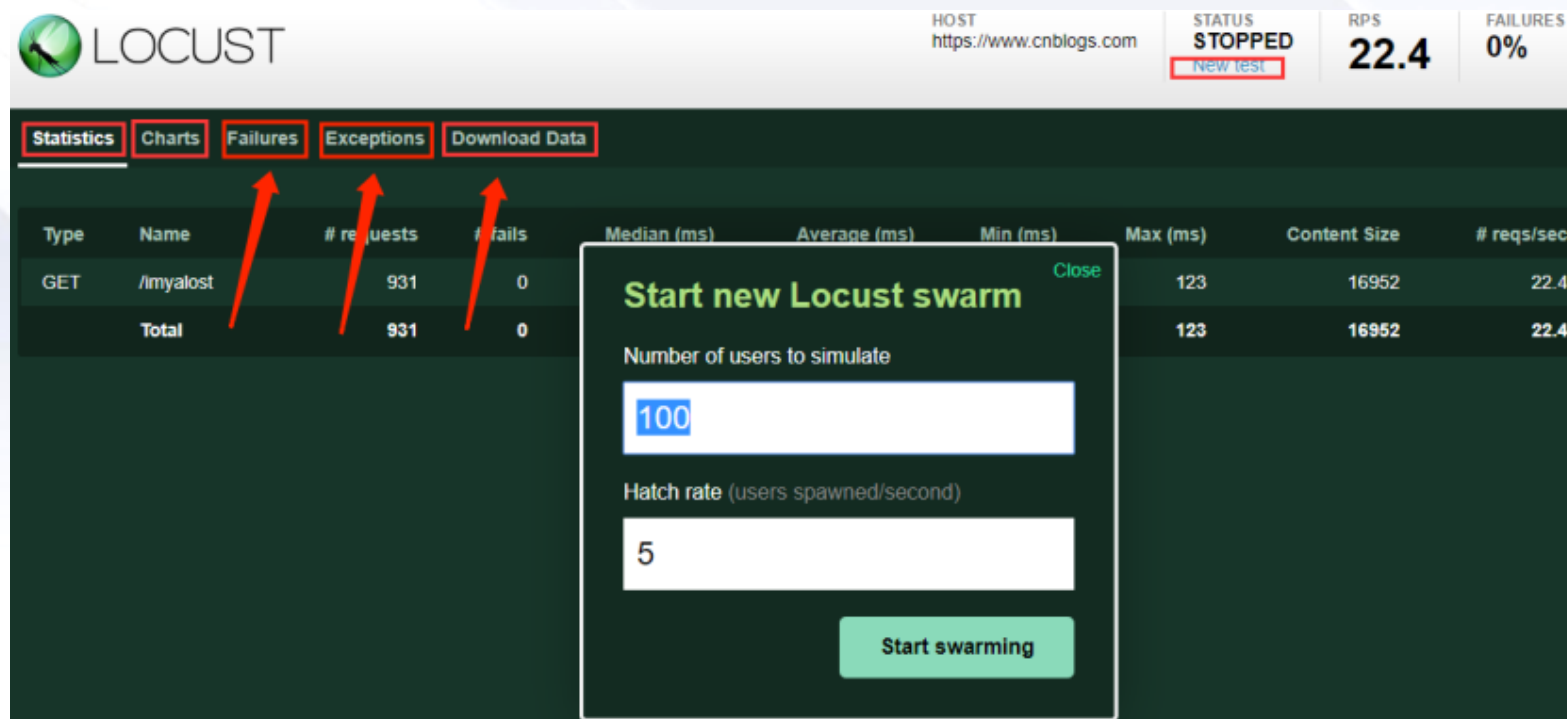
2.3 Locust界面介绍

各模块说明

Failures：失败请求的展示界面；

Exceptions：异常请求的展示界面；

Download Data：测试数据下载模块，提供三种类型的CSV格式的下载，分别是：Statistics、responsetime、exceptions；



LOCUST

HOST: https://www.cnblogs.com

STATUS: STOPPED

RPS: 22.4

FAILURES: 0%

Statistics Charts Failures Exceptions Download Data

Type	Name	# requests	# fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Content Size	# reqs/sec
GET	/lmyalost	931	0				123	16952	22.4
Total		931	0				123	16952	22.4

Start new Locust swarm

Number of users to simulate

100

Hatch rate (users spawned/second)

5

Start swarming

2.4 Python语法

Locust性能测试是基于Python来完成的。

import

第6节课程中我们讲过Python中库的用法（import requests），而因完成Locust性能测试使用的代码对库的需求与之前单纯的接口测试不同，库的引用方式略微有些差异。

本次课程的性能测试案例需要locust库，命令如下：

```
from locust import HttpUser, task, between
```

2.4 Python语法

class

用来描述具有相同的属性和方法的对象的集合。它定义了该集合中每个对象所共有的属性和方法。对象是类的实例。如下所示，Python 类是由类头（class 类名）和类体（统一缩进的变量和函数）构成。

```
class User(HttpUser):  
    wait_time = between(1, 3)
```

2.4 Python语法

HttpUser (类)

HttpUser类表示一个(群)用户，实际执行测试时Locust会为每个用户生成一个用户类的实例。在用户类中可以定义一些常见的属性。

```
class User(HttpUser):  
    wait_time = between(1, 3)
```

2.4 Python语法

wait_time (属性)

用户类的wait_time属性用于定义多个用户在执行任务之间的等待时间，案例中设置的时间是1~3秒。

```
class User(HttpUser):  
    wait_time = between(1, 3)
```

2.4 Python语法

@task()

@task() 是一个装饰器，装饰下面的swagger_ui方法为一个任务。1表示一个Locust实例被挑选执行的频率为1。此处数值越大，执行频率越高。

```
@task(1)
def swagger_ui(self):
    self.client.get("usercenter/data/authorchannel?cmd=frontpage")
```

2.4 Python语法

def

在python中，定义一个函数需要用到def关键字，其后紧跟着的是函数名，函数名（ ）内是参数。

```
@task(1)
def swagger_ui(self):
    self.client.get("?sfrom=baidu-top")
```

2.5 Locust性能测试案例

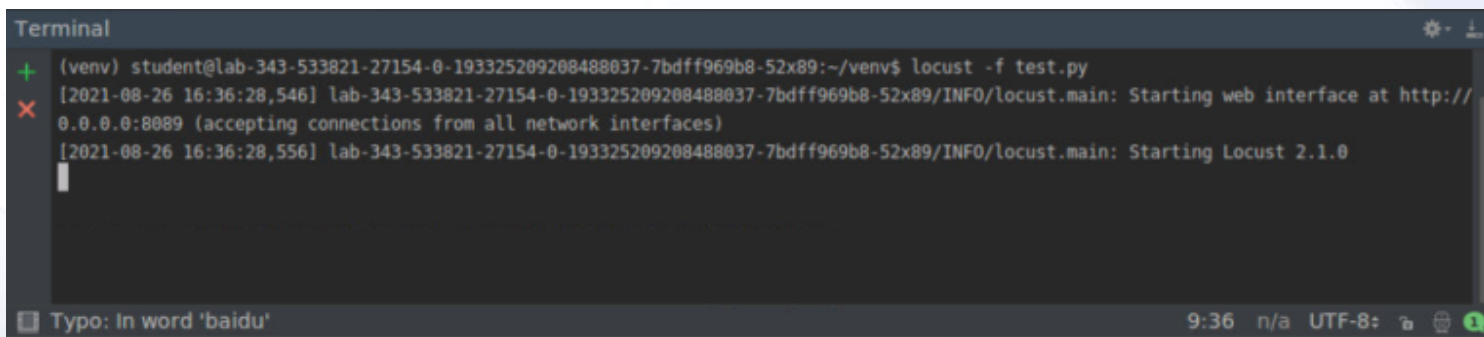
locust性能测试案例可以在pycharm中运行。这个案例是检测百度视频（好看视频）网页的性能状况。网页的完整地址为：
<https://haokan.baidu.com/?sfrom=baidu-top>

以我们刚才介绍的python语法为参照，我们来尝试编写并运行一个locust性能测试的案例，以下为案例的代码：

```
test1.py x
1  from locust import HttpUser, task, between
2
3  class User(HttpUser):
4      wait_time = between(1, 3)
5
6      @task(2)
7      def swagger_ui(self):
8          self.client.get("?sfrom=baidu-top")
9
10
```

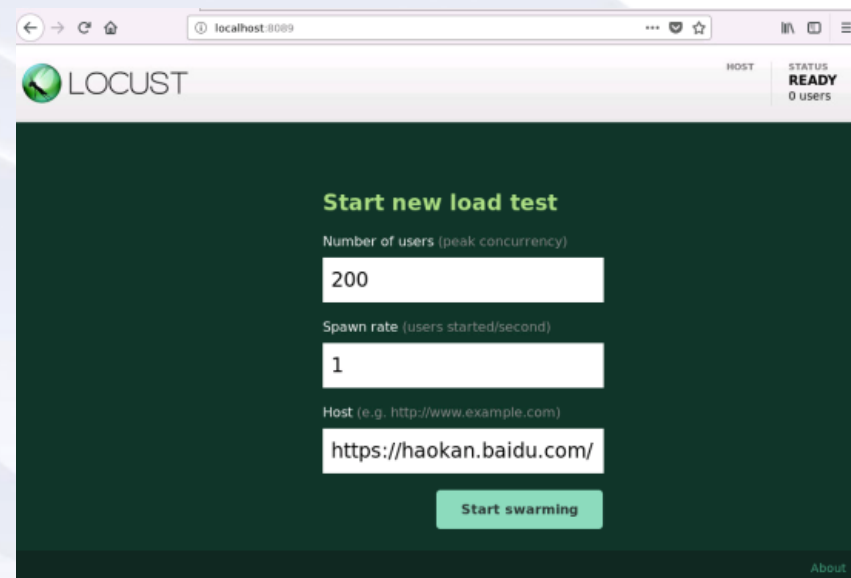
2.5 Locust性能测试案例

我们可以通过在终端输入 “locust -f test.py” 来启动程序，启动完成如下图所示。



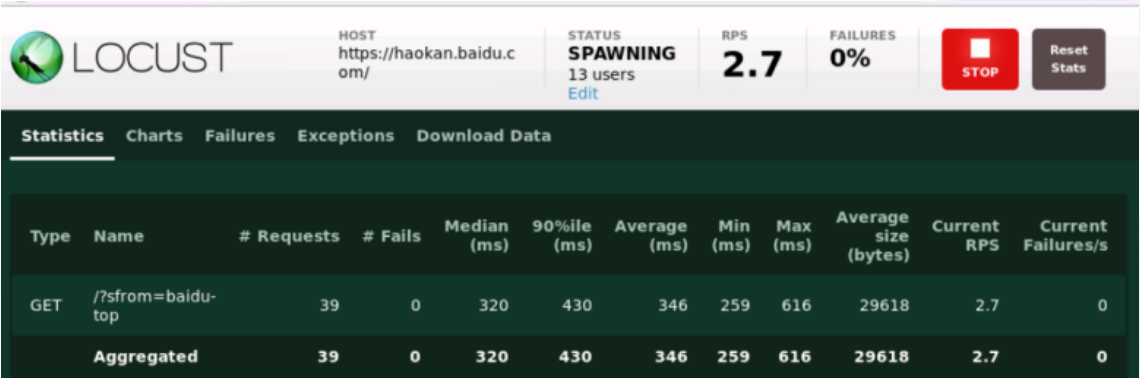
```
Terminal
+ (venv) student@lab-343-533821-27154-0-193325209208488037-7bdf969b8-52x89:~/venv$ locust -f test.py
[2021-08-26 16:36:28,546] lab-343-533821-27154-0-193325209208488037-7bdf969b8-52x89/INFO/locust.main: Starting web interface at http://0.0.0.0:8089 (accepting connections from all network interfaces)
[2021-08-26 16:36:28,556] lab-343-533821-27154-0-193325209208488037-7bdf969b8-52x89/INFO/locust.main: Starting Locust 2.1.0
```

按照右图中的参数来设置模拟用户总数（200）、每秒启动的虚拟用户数（1）以及服务器地址（<https://haokan.baidu.com/>）。



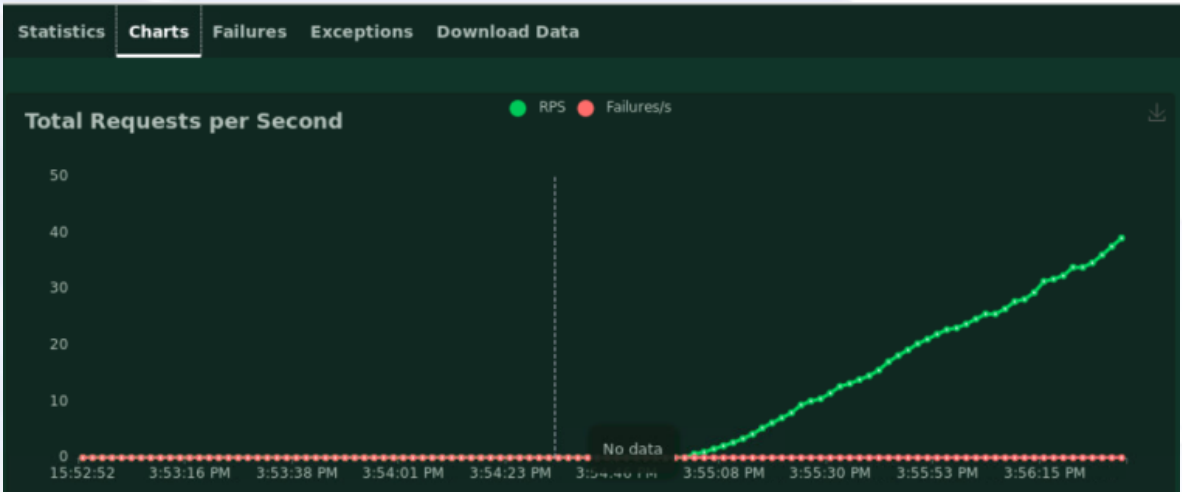
2.5 Locust性能测试案例

设置完成后启动页面数据页面会不断更新性能状况，如下图所示：



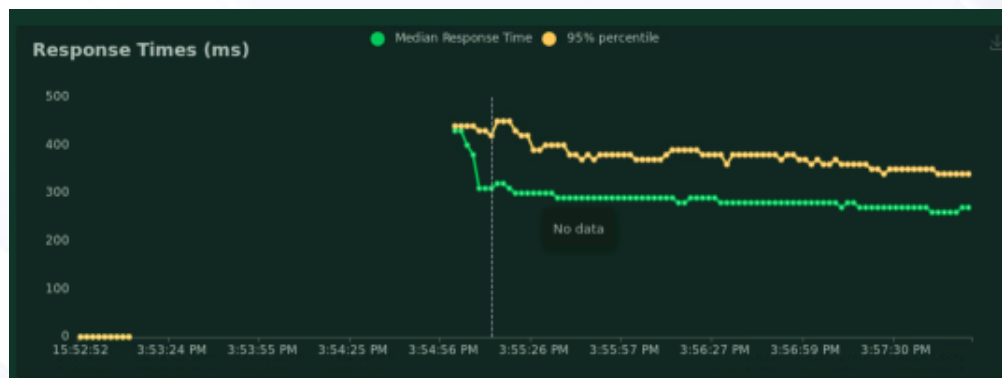
可以点击Charts进入图表界面来观察情况。

右侧这张图表示每秒请求次数，图中的线段上下波动较小，说明性能稳定：

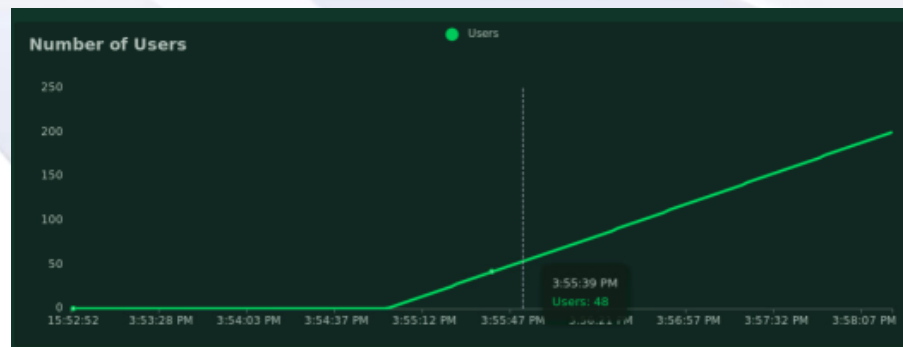


2.5 Locust性能测试案例

以下这张图表示响应时间，黄色的线表示最大响应时间，绿色的线表示最小响应时间，一般响应时间介于2~5秒为正常，超过8秒较差，图中所示的响应时间属于正常范畴，最大值如果持续高位就需考虑进行性能优化：

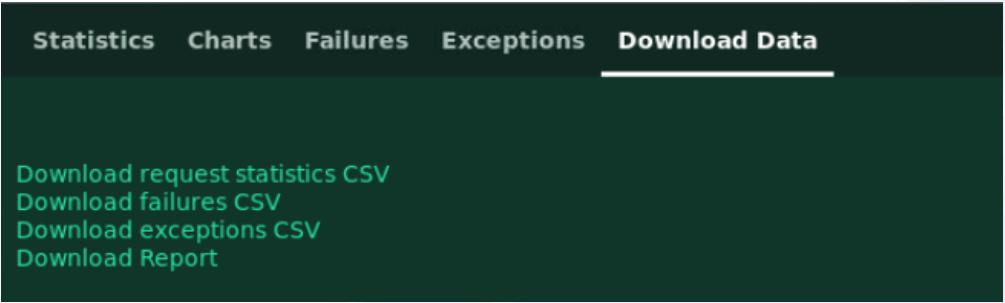


以下这张图表示虚拟用户数，从图中可以看到用户数在随时间的增加而增加：



2.5 Locust性能测试案例

如果后续需要提取数据做进一步的分析，我们可以在Download Data处下载我们想要的数 据，如下图所示。



如果选择了Download Report，系统将生成一份Locust测试报告，如右图所示，报告中的内容包含了本次性能测试的详细信息，例如：测试时长、目标主机、请求数据、响应时长数据、以及之前提到的三张表格。

Locust Test Report

During: 2021-08-26 16:38:18 - 2021-08-26 16:40:00

Target Host: https://haokan.baidu.com/

Download the Report

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/?sfrom=baidu-top	2349	0	289	218	555	29667	23.0	0.0
Aggregated		2349	0	289	218	555	29667	23.0	0.0

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/?sfrom=baidu-top	280	290	300	310	340	360	410	560
Aggregated		280	290	300	310	340	360	410	560

Charts

2.6 小结与思考

本小节我们了解了Locust，它是一个简单易用的分布式用户负载测试工具，用于web站点(或其他系统)的性能测试，可以计算一个系统可以处理多少并发用户。

问题思考

请同学们调研还有哪些性能测试工具，并将它们与Locust进行对比，并思考Locust与这些性能测试工具相比未来还有哪些功能可以增加与完善？



第三章 Caliper性能测试

- 3.1 Caliper简介
- 3.2 Caliper工作架构
- 3.3 Caliper测试
- 3.4 小结与思考

3.1 Caliper简介

Hyperledger Caliper是一款区块链基准测试工具，用户可以在定义好测试集的情况下针对自己的**区块链网络**进行**性能测试**，获取一系列的测试结果并生成**测试报告**。Caliper性能测试不作为本课程的掌握对象，这里仅对其进行简单地介绍，有兴趣的同学可以查阅相关资料进行了解。

目前支持的区块链解决方案：

Hyperledger Besu

Hyperledger Fabric v1.X, v2.X

Ethereum

FISCO BCOS

3.1 Caliper简介

Caliper当前支持的性能指标：

1、交易成功率：交易成功率是指在系统运行期间，系统处理请求交易（**成功数/(成功数+失败数)*100%**）的比率,期间系统是可用的，失败可能是由于程序处理逻辑错误、数据错误、超时等原因造成的。

2、交易TPS：TPS的英文是Transaction Per Second，指每秒钟发生的交易数量，是软件测试结果的测量单位。一个事务是指一个客户端向服务器发送请求，然后服务器做出反应的过程。客户端在发送请求时开始计时，收到服务器响应后结束计时，以此来计算使用的时间和完成的事务个数。

3、交易响应时间：响应时间指执行一个请求从开始到最后收到响应数据所花费的总体时间，即从客户端发起请求到收到服务器响应结果的时间。它是一个系统最重要的指标之一，它的数值大小直接反映了系统的快慢。

4、资源消耗：消耗的资源包括CPU、内存、以及网络IO等。

3.1 Caliper简介

Caliper性能测试框架可以分为三个层次：

适配层 Adaptation Layer

适配层用于将现有的区块链系统与Caliper框架集成。适配器使用了相应的链SDK和API实现了Caliper Blockchain NBIs。

接口&核心层

区块链操作接口：包含部署合约、调用合约、查询账本状态等操作；
资源监控：对docker容器、本地进程进行监控，包括CPU、内存、网络IO等的状态；
性能分析：读取预定义的性能数据（如TPS、延迟、成功率等）并打印报告。这些数据在调用NBI的时候会记录（如创建时间，交易提交时间，交易结果等）；
生成HTML报告。

应用层

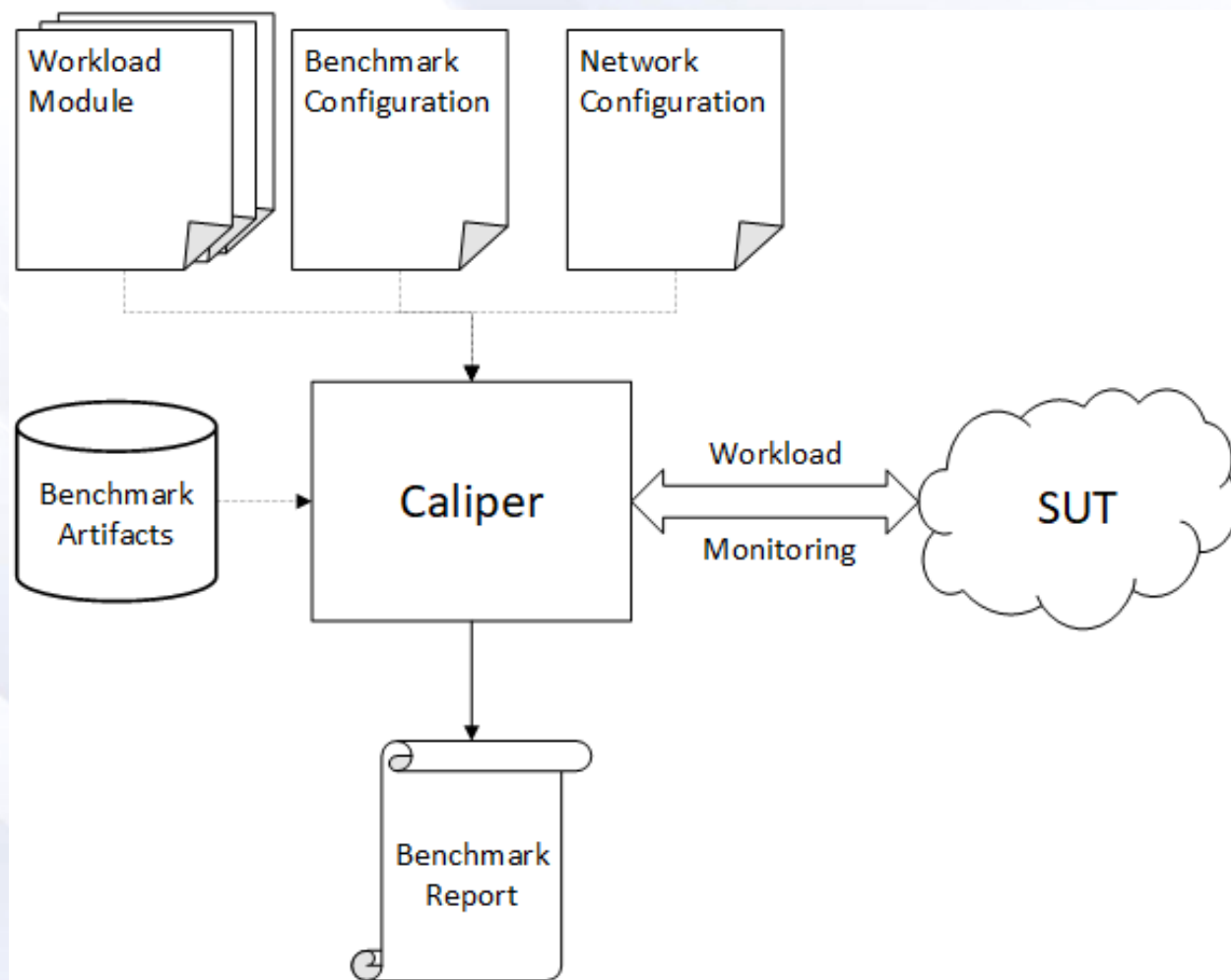
应用层来运行区块链测试场景。每一个测试场景由一个配置文件来定义，包含底层区块链网络的配置和测试参数。
项目内置一个默认的区块链引擎，当然开发人员也可以基于NBI来定义自己的区块链引擎。

3.2 Caliper工作架构

在最简单的形式下，Caliper是一种针对特定被测系统(SUT)生成工作负载并持续监控其响应的服务。Caliper根据观察到的SUT响应生成测试报告。

感兴趣的同学，可以访问以下网址来详细了解Caliper。

<https://hyperledger.github.io/caliper/v0.4.2/architecture/>



3.3 Caliper测试报告

在测试完成后，Caliper也可以生成一份测试报告，位于caliper-workspace目录，报告名称为report.html的，部分内容如图：

Caliper Report

Basic information

DLT: fabric
Benchmark: simple
Description: This is an example benchmark for caliper, to test the backend DLT's performance with simple account opening & querying transactions
Test Rounds: 12
[Details](#)

Benchmark results

[Summary](#)
[round 0](#)
[round 1](#)
[round 2](#)
[round 3](#)

Summary

Test	Name	Succ	Fail	Send Rate	Max Latency	Min Latency	Avg Latency	Throughput
1	open	1000	0	50 tps	0.25 s	0.03 s	0.13 s	50 tps
2	open	1000	0	100 tps	0.19 s	0.04 s	0.10 s	100 tps
3	open	1000	0	150 tps	0.65 s	0.06 s	0.24 s	149 tps
4	open	1000	0	195 tps	6.47 s	0.35 s	2.55 s	136 tps
5	open	1000	0	245 tps	6.38 s	0.77 s	3.47 s	148 tps
6	open	1000	0	286 tps	6.01 s	2.40 s	3.83 s	151 tps
7	query	5000	0	100 tps	0.10 s	0.01 s	0.01 s	100 tps
8	query	5000	0	200 tps	0.17 s	0.01 s	0.04 s	200 tps
9	query	5000	0	300 tps	0.74 s	0.01 s	0.21 s	300 tps
10	query	5000	0	340 tps	20.28 s	2.26 s	12.18 s	246 tps
11	query	5000	0	354 tps	19.97 s	3.51 s	12.02 s	249 tps
12	query	5000	0	357 tps	19.60 s	1.99 s	11.79 s	255 tps

3.4 小结与思考

本小节我们了解了Caliper性能测试及其相关测试报告。有兴趣的同学可以查阅相关资料对Caliper性能测试进行更多的了解。

问题思考

请同学们调研并思考Caliper如何在FISCO BCOS上进行性能测试？

