



下载APP



19 | 性能测试工具：如何选择最合适的性能测试工具？

2021-06-29 尉刚强

《性能优化高手课》

课程介绍 >



讲述：尉刚强

时长 17:36 大小 16.12M



你好，我是尉刚强。这节课，我们来聊聊性能测试工具的选择和使用。

我们都知道，性能测试是在确保软件功能正确的前提下，通过一些测试与加压手段，来衡量软件性能表现、分析性能问题的一种技术方法。而性能测试工具，则是支撑性能测试工作效率的重要保障之一，所以在进行性能测试之前，选择一款合适的性能测试工具非常重要。

不过，虽然很多工具使用的性能测试原理是相似的，但由于**不同业务领域的差异性太大**选用的性能测试工具可能完全不同。



比如说，企业应用服务领域与专用设备领域所使用的性能测试工具，就是完全不同的体系，像 LoadRunner 这类就属于企业应用服务领域的性能测试工具。而在专用设备领域

中，智能汽车、无线通信设备所使用的性能测试工具，有很多都是专属硬件设备，基本都是定制化的，所以对其他领域的性能测试工具的选择借鉴意义很小。

因此今天这节课，我要给你介绍的性能测试工具选择，就主要聚焦在**企业应用与服务级**的维度上。

另外我们也知道，如今处于富技术工具的时代，单独针对某个业务领域的性能测试，可选的性能测试工具也依旧非常多。但如果我们选择的性能测试工具不合适，就将会长期影响到整个团队的性能测试效率和成本。

所以接下来，我会先给你介绍下该领域的性能测试工具现状与发展趋势，在此过程中，你就能明白每种性能测试工具的特点和优势，以及它背后主要解决了什么问题。然后，我还会带你理解选择性能测试工具的参考标准。这样，当你在掌握了选择这些工具的参考标准之后，也就找到了理解它们背后工作原理的捷径，以后在进行性能测试之前，你就可以快速地找到合适的性能测试工具，提升性能测试的效率。

性能测试工具的现状与发展趋势

好，下面我们就先来了解下，应用服务级性能测试工具的现状和发展趋势。不过在开始之前，我想先说明一下今天我要给你介绍的性能测试的相关工具，主要包括：JMeter、LoadRunner、Locust、PTS、k6、Postman、Chrome 调试器等。

虽然在课程中，我无法全面地罗列出目前业界所有的测试工具，但这些工具里有很多都是我在以往项目中真实使用过的，所以它们一定是最具代表性的。学完这部分内容后，接下来你就只需要搞明白自己当前业务产品的性能测试需求，就可以据此选择出合适的性能测试工具了。

OK，现在，我们就来具体了解下性能测试工具的变化趋势。早期的性能测试工具以 JMeter、LoadRunner 为代表，它们都使用桌面程序的运行方式，在业界也已经有很长的应用历史，所以内置的功能很丰富。而由于早期软件工程的生命周期很长，性能测试工具通常是从企业级应用的端到端进行性能测试，并会由专门的测试人员长期负责。所以，这类性能测试工具的特点就是**功能相对比较封闭，而且使用起来也比较笨重**。

但如今，软件技术架构与软件工程都发生了很大的演进变化，比如软件技术架构从单体转向了分布式微服务化，而软件工程管理也从瀑布式向敏捷式这种更加快速的迭代方式演

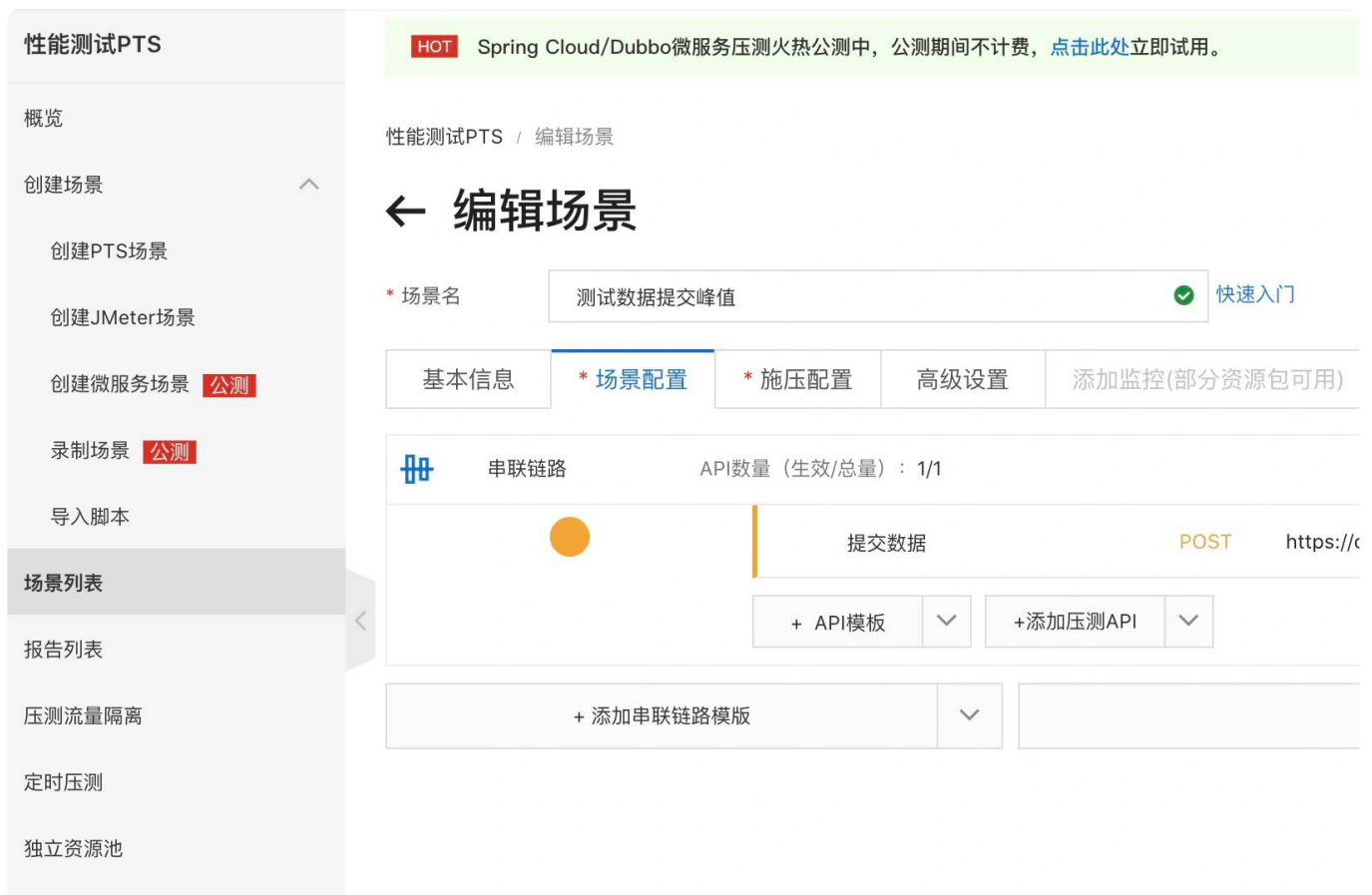
进。所以，为了更好地适应这个时代，性能测试工具也发生了很大改变。其中，**云服务化、代码化**就是两个最主要的变化趋势，下面我来重点说明下。

云服务化性能测试工具

现在可以说是云原生（Cloud Native）的时代，所有的技术与服务都逐步开始提供按需服务的能力，而性能测试对计算资源的需求比较大，所以**按需使用**的能力也很重要。

那么，关于云服务化的性能测试工具，这里我给你介绍一个典型的代表：阿里云的 PTS。

PTS（Performance Test Service）是阿里云上提供的一个性能测试服务，你在登录购买之后就可以立即使用。下面展示的就是 PTS 登录之后的控制器界面视图：



从图中你能看出，阿里云的 PTS 提供了场景录制、场景管理、施压配置、报告分析等性能测试相关的所有基本功能。当然，**它最大的特点是支持非常大的弹性计算能力**，所以你可以很轻松地配置出超级大规模和强度的压力性能测试。

实际上，云服务化的性能测试工具，与传统的性能测试工具（如 JMeter）在界面交互上是比较像的，所以，对原来使用传统性能测试工具的程序员来说切换成本并不大。同时，云

服务化的性能测试还提供了比较完整的帮助手册，所以学习成本也比较低，非常容易上手。

另外，从阿里云的 PTS 使用界面上你也会发现，云端性能测试工具还有一个好处，就是可以帮助我们省去性能测试时安装测试工具的步骤。因此，如果你的性能测试场景不是特别复杂，也不需要考虑性能测试与 CI 流水线集成的场景，那么使用云端性能测试工具，就会是一个不错的选择。


代码化性能测试工具

如果你是一个非常喜欢敲代码的程序员，那么你可能会对界面化的性能测试工具没有什么好感，毕竟这种工具是通过界面的配置能力来生成测试代码的，所以并没有代码实现灵活和丰富。

不过我告诉你，现在有些性能测试工具，已经可以完全基于代码来开发性能测试场景用例了，所以你就可以像开发业务代码一样去编写性能测试用例，比如 Locust、k6 等。我把这种工具归纳为**代码化性能测试工具**。


这里我先以 Locust 为例，给你展示下它基于代码开发性能测试用例的过程。

实际上，对于这种性能测试工具来说，你可以完全基于 Python 来编写构造性能测试用例，下面的示例是我曾经编写的一个性能测试用例中的小片段，代码中要体现的功能是测试接口发送一个 POST 请求，并且携带 cookies 和 body 消息体：

 复制代码


```
1 // test_submit_case.py
2 from locust import HttpLocust, TaskSet, task
3 class WebsiteTasks(TaskSet):
4     @task
5     def add_entry(self):
6         body = {"field_1": "asdf"}
7         cookies = dict(_session='V1hj aWV4Ujd3WndDWWZBa0ZiTjdvd2tLV2p3NmtkUzA0R
8         self.client.post("/test/api", json=body, cookies=cookies)
9
10 class WebsiteUser(HttpLocust):
11     task_set = WebsiteTasks
12     min_wait = 5000
13     max_wait = 15000
```

这样，在运行如下代码之后，你就可以在界面启动性能测试和查看结果了。

 复制代码

```
1 locust -H https://xxx.net -f ./test_submit_case.py
```

而 k6 也是基于代码化的性能测试工具，但同时，它也提供了基于界面自动生成性能测试代码的机制。这是一个使用 Node.js 编写的性能测试用例，用来测试页面的 get 操作：

 复制代码

```
1 //test_get.js
2 import http from 'k6/http';
3 import { sleep } from 'k6';
4 export default function () {
5   http.get('http://test.k6.io');
6   sleep(1);
7 }
```

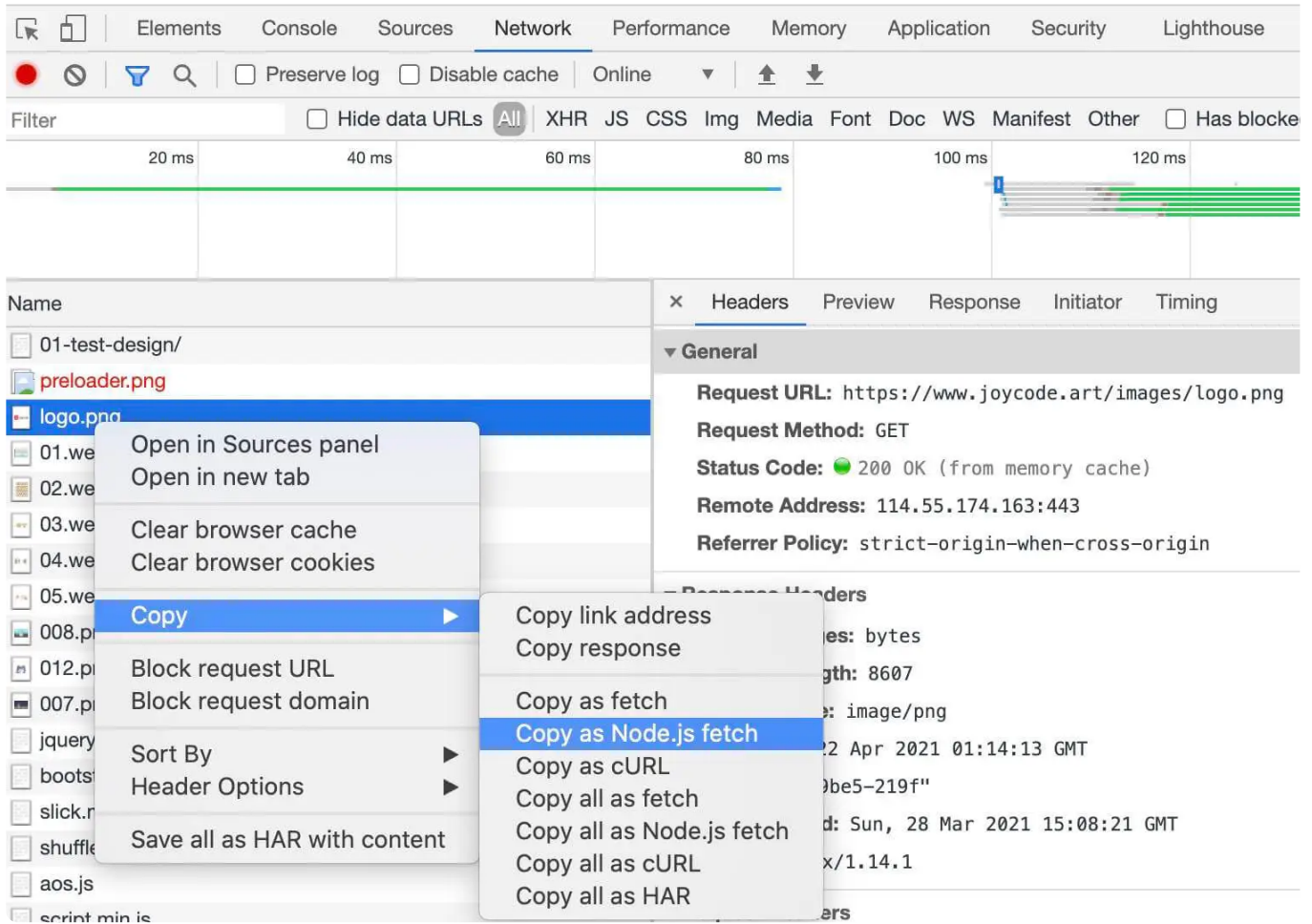
在安装完 k6 之后，你就可以使用命令 `k6 run testget.js` 来运行上面的性能测试用例，并且生成测试结果以供进行性能分析。

实际上，k6 也是一款云服务化的性能测试工具，当你注册登录了 k6 Cloud 之后，就可以直接在 k6 Cloud 上使用脚本或者界面创建性能测试用例。

作为辅助使用的性能测试工具

另外，还有一些性能测试工具，它们提供的功能并不算非常完备，比如 **Postman 工具**，它可以临时地针对某个 REST 接口进行性能测试，所以在做性能测试用例开发和调试时，你可以拿来使用。而还有一些工具只是作为性能测试工具的协助手段，比如 **Chrome 调试器**，你可以用它来快速获取一个 REST 请求的 Node.js 接口调用代码。

这里我们也来看看使用 Chrome 调试器的具体操作流程：首先，使用快捷键进入 Chrome Debug 模式，在操作 Web 页面之后，在菜单 Network 下寻找到被测试的 REST 接口；然后点击右键，我们就可以选择对应的菜单操作，如以下截图所示：



操作之后，你就可以获取对应 REST API 调用的 Node.js 代码，如下所示：

复制代码

```
1 fetch("https://www.joycode.art/images/logo.png", {
2   "headers": {
3     "sec-ch-ua": "\"Chromium\";v=\"88\"", "\"Google Chrome\";v=\"88\"", "\";Not A
4     "sec-ch-ua-mobile": "?0"
5   },
6   "referrer": "https://www.joycode.art/blog/01-test-design/",
7   "referrerPolicy": "strict-origin-when-cross-origin",
8   "body": null,
9   "method": "GET",
10  "mode": "cors"
11 });
```

接着，你就可以将其添加到 k6 对应的性能测试脚本中，来运行性能测试。当然，你也可以完全使用 Node.js，来自己快速开发构造一个简易的性能测试场景。

好，到这里，我们就对性能测试工具的使用现状和发展趋势有一定的了解了。不过我们还是需要再进一步思考一下：现在的性能测试工具这么多，并不是每一款都适合自己的性能

测试场景呀。所以，我们还需要基于一些评选参考标准，来帮助我们选择合适的性能测试工具。

性能测试工具评选参考因素

我们知道，传统的性能测试是在项目交付的后期开展的，而这样做会出现的问题就是，通过性能测试发现的性能问题，修复成本太大，甚至可能会打乱产品的交付节奏与计划。同时，由于性能测试与开发之间的脱节情况也比较严重，所以就会导致性能测试问题的发现与分析过程效率很低。

因此，为了更好地开展性能测试，我们其实可以**把性能测试的工作尽量提前，打破性能测试团队与特性开发团队之间的壁垒，让性能测试成为研发团队内部的一项活动。**

那么为了更好地实现这个目标，依据我在性能测试中积累的实践经验，我给你总结了一些很实用的评选参考标准，你可以基于这些标准来选择性能测试工具，这样就可以更好地把测试工作和开发工作结合起来，进一步也就可以帮助团队之间进行良好的协作了：



这些评选参考标准是**按照优先级程度由高到低**排列的，下面我就逐个给你详细介绍下。

是否支持测试接口类型

首先你要注意，这其实是一个**必要的条件**。也就是说，这个评判标准与软件是否满足业务功能一样重要。毕竟如果测试工具不支持测试接口，那就不能完成最基本的性能测试需求了。

我举个例子，如果你的被测接口是 FTP 接口，但选择的性能测试工具只支持对 REST 接口的测试，那么这款性能测试工具就不是一个好的选择。像 JMeter 这类历史沉淀比较久的性能测试工具，它所支持的测试接口类型就很丰富，比如 MOM (Message-Oriented Middleware , 消息中间件)、JDBC SQL、FTP 等。

但在新的软件技术架构实现中，接口越来越标准化，所以现在新产生的性能测试工具，支持的接口类型可能就比较少（比如 PTS 就没有 LoadRunner 支持的接口多）。因此，在选择性能测试工具前，你首先需要确保该工具支持你需要进行性能测试的所有接口类型。

压测资源的弹性能力

这同样也是一个很重要的考虑因素，但也是在调研和选择性能测试工具时，我们比较容易忽略的因素。以 Locust 性能测试工具为例，受制于本地 CPU 的单核性能，当压测并发用户数达到一定的数量时，测试工具侧的性能就很容易先到达瓶颈，从而造成测试出的系统性能不准确。

那么这个时候，你其实也可以通过分布式部署和运行模式，同时启动很多个节点来进行性能测试，但这样造成的性能测试复杂度就增加了。也就是说，如果你使用 JMeter、LoadRunner 这类需要安装部署的性能测试工具，就需要考虑到这些因素。

但对于像云服务化的性能测试工具，比如 PTS，你就不用担心性能测试工具本身的性能限制了。

代码化能力

这是我个人很看重的一项评估依据。为了提升运维效率，DevOps 已经被更多人所关注和认可，而基础设施代码化是支持 DevOps 的重要技术手段之一。那么对性能测试而言，测试用例代码化带来的好处，就是你不仅可以使版本管理工具来管理测试用例，还能更容易地实现性能测试自动化，并且能很方便地集成到 CI 流水线上。

测试结果的可视化能力

很多性能测试工具都支持了测试结果的图形化显示，但是你要注意，这中间依旧存在一定的差异。

首先，不同的图表显示方式对结果分析的帮助是不一样的；其次，性能测试结果还有一个重要价值，就是可以帮助我们分析软件迭代演进期间，性能基线的变化趋势。

比如，使用 k6 这种性能测试工具，可以便于我们把最终的测试结果对接到数据库中，然后我们就可以使用 Grafana 去显示和分析性能基线的变化趋势。那么，这种可视化能力的价值，就要大于只能在工具内来图形化显示性能数据的能力。

安装部署便利性

这也是一个考量因素。我们知道，有些性能测试工具，可能只需几个命令就可以自动化安装了，而基于云的性能测试工具，可能压根就不需要去安装。

那么就我的观察和实践，我发现越复杂的安装过程，就越容易在安装过程中出错，而且对每一位使用该工具的研发人员来说，这都是不能逃避的工作。所以，从整个研发团队的角度来思考的话，需要安装部署的性能测试工具所花费的成本会更大。

是否支持录制脚本

使用录制脚本可以比较方便地构造测试场景与数据，但是以我个人的经验而言，这个优先级并不是非常高。

主要是由于两方面的因素：一方面，我们在录制脚本的使用过程中很容易出错，而且还需要手动修改生成的脚本，使用起来也不是非常高效；另一方面，录制脚本的部分能力，我们其实可以依赖其他工具协助完成，比如 Chrome 的 Debug 模式。

是否收费，是否有售后

这个考量因素的优先级因公司而异，有些公司并没有支付性能测试工具的预算，所以收费产品只能排除在外。而有些公司会有这样的预算，那么你就可以从研发团队成本的视角，

来考虑选择合适的性能工具。

小结

首先你要知道，对软件系统的性能测试并不是一次性的，为了保持软件系统在演进过程中的性能，可以长期处于有效状态，你需要不断对其进行性能测试，而性能测试工具就是保证性能测试效率的关键因素。

那么在今天的课程中，我就从企业应用与服务级的性能测试视角，给你介绍了性能测试工具的发展现状。同时，基于为了更好地开展性能测试的目标，我还带你了解了选择性能测试工具的一些参考因素。当你参与产品的性能测试场景与需求存在差异时，你仍然也可以参考这些评选参考因素中的内容，来作为你选择性能测试工具的依据。

思考题

在你的产品性能测试中，性能测试用例是代码化的吗？具体用例是怎么管理的呢？

欢迎在留言区分享你的答案和思考。如果觉得有收获，也欢迎你把今天的内容分享给更多的朋友。

分享给需要的人，Ta订阅后你可得 **20** 元现金奖励

 赞 0  提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 18 | Benchmark测试（下）：如何做好宏基准测试？

下一篇 20 | 性能看护：如何更好地守护产品性能？

更多学习推荐

Java 面试必考 300 题

最新汇总

限时免费领取



精选留言

写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。