41 | 从小工到专家：聊聊测试执行环境的架构设计（下）

2018-10-01 茹炳晟

讲述：茹炳晟
时长 13:41　大小 6.27M

你好，我是茹炳晟，今天我和你分享的主题是“从小工到专家：聊聊测试执行环境的架构设计（下）”。

在上一篇文章中，我介绍了测试基础架构的概念，以及早期的和经典的两种测试基础架构。在文章的最后，我提到经典的测试基础架构中采用的 Selenium Grid 方案，在测试用例的数量持续增加的情况下，会带来集群扩容、Jenkins Job 臃肿不堪等诸多问题，因此我们考虑将 Selenium Grid 迁移到 Docker，并且提供便于 Jenkins Job 管理的统一测试执行平台。

所以，今天的这篇文章，我就会围绕这些瓶颈以及对应的解决方案来展开。

基于 Docker 实现的 Selenium Grid 测试基础架构

随着测试基础架构的广泛使用，以及大量的浏览器兼容性测试的需求，Selenium Grid 中 Node 的数量会变得越来越大，也就是说我们需要维护的 Selenium Node 会越来越多。

在 Node 数量只有几十台的时候，通过人工的方式去升级 WebDriver、更新杀毒软件、升级浏览器版本，可能还不是什么大问题。但是，当需要维护的 Node 数量达到几百台甚至几千台的时候，这些 Node 的维护工作量就会直线上升。虽然，你可以通过传统的运维脚本管理这些 Node，但维护的成本依然居高不下。

同时，随着测试用例数量的持续增长，Selenium Node 的数量也必然会不断增长，这时安装部署新 Node 的工作量也会难以想象。因为，每台 Node 无论是采用实体机还是虚拟机，都会牵涉到安装操作系统、浏览器、Java 环境，以及 Selenium。

而目前流行的 Docker 容器技术，由于具有更快速的交付和部署能力、更高效的资源利用，以及更简单的更新维护能力，也就使得 Docker 相比于传统虚拟机而言，更加得“轻量级”。

因此，**为了降低 Selenium Node 的维护成本，我们自然而然地想到了目前主流的容器技术，也就是使用 Docker 代替原本的虚拟机方案。**

基于 Docker 的 Selenium Grid，可以从三个方面降低我们维护成本：

1. 由于 Docker 的更新维护更简单，使得我们只要维护不同浏览器的不同镜像文件即可，而无需为每台机器安装或者升级各种软件；
2. Docker 轻量级的特点，使得 Node 的启动和挂载所需时间大幅减少，直接由原来的分钟级降到了秒级；
3. Docker 高效的资源利用，使得同样的硬件资源可以支持更多的 Node。也就是说，我们可以在不额外投入硬件资源的情况下，扩大 Selenium Grid 的并发执行能力。

因此，现在很多大型互联网企业的测试执行环境都在向 Docker 过渡。

而具体如何基于 Docker 搭建一套 Selenium Grid，你可以参考我在第 39 篇文章《从小作坊到工厂：什么是 Selenium Grid？如何搭建 Selenium Grid？》中介绍的方法。由此可见，将原本基于实体机或者虚拟机实现的 Selenium Grid 改进成为基于 Docker 实现的过程也很简单、灵活。

如图 1 所示，就是一个基于 Docker 实现的 Selenium Grid 的测试基础架构。

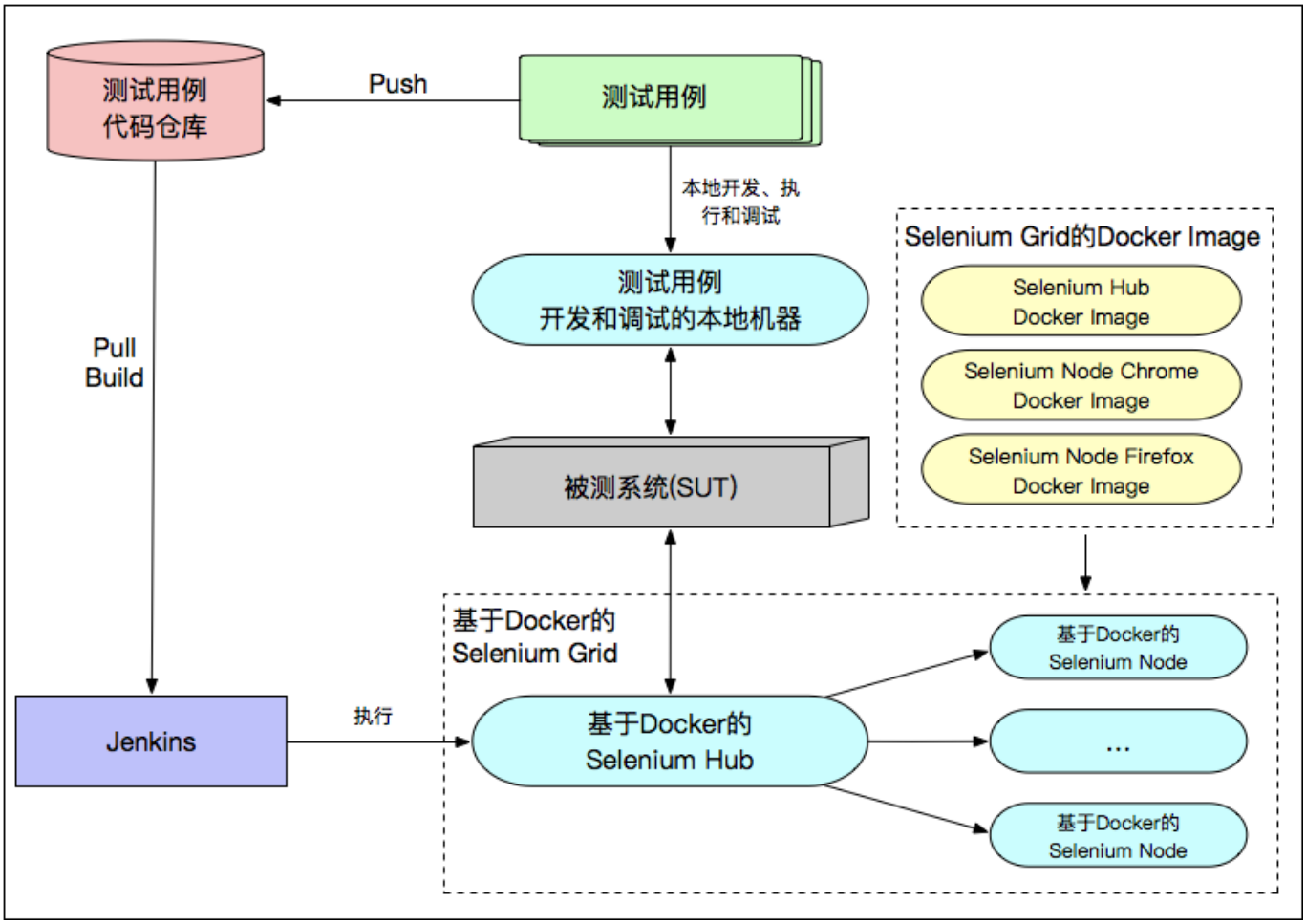

图 1 基于 Docker 实现的 Selenium Grid 测试基础架构

引入统一测试执行平台的测试基础架构

在实际的使用过程中，基于 Docker 的 Selenium Grid 使得测试基础架构的并发测试能力不断增强，也因此会有大量项目的大量测试用例会运行在这样的测试基础架构之上。

当项目数量不多，我们可以直接通过手工配置 Jenkins Job，并直接使用这些 Job 控制测试的发起和执行。但是，当项目数量非常多之后，测试用例的数量也会非常多，这时新的问题又来了：

1. 管理和配置这些 Jenkins Job 的工作量会被不断放大；
2. 这些 Jenkins Job 的命名规范、配置规范等也很难实现统一管理，从而导致 Jenkins 中出现了大量重复和不规范的 Job；
3. 当需要发起测试，或者新建某些测试用例时，都要直接操作 Jenkins Job。而这个过程，对于不了解这些 Jenkins Job 细节的人（比如，新员工、项目经理、产品经理）来说，

这种偏技术型的界面体验就相当不友好了。

为此，我们为了管理和执行这些发起测试的 Jenkins Job 实现了一个 GUI 界面系统。在这个系统中，我们可以基于通俗易懂的界面操作，完成 Jenkins Job 的创建、修改和调用，并且可以管理 Jenkins Job 的执行日志以及测试报告。

这，其实就是统一测试执行平台的雏形了。

有了这个测试执行平台的雏形后，我们逐渐发现可以在这个平台上做更多的功能扩展，于是这个平台就逐渐演变成了测试执行的统一入口。

在这里，我列举了这个平台两个最主要的功能和创新设计，希望可以给你以及你所在公司的测试基础架构建设带来一些启发性的思考。

第一，测试用例的版本化管理。我们都知道，应用的开发有版本控制机制，即：每次提测、发布都有对应的版本号。所以，为了使测试用例同样可追溯，也就是希望不同版本的开发代码都能有与之对应的测试用例，很多大型企业或者大型项目都会引入测试用例的版本化管理。最简单直接的做法就是，采用和开发一致的版本号。

比如，被测应用的版本是 1.0.1，那么测试用例的版本也命名为 1.0.1。在这种情况下，当被测应用版本升级到 1.0.2 的时候，我们会直接生成一个 1.0.2 版本的测试用例，而不应该直接修改 1.0.1 版本的测试用例。

这样，当被测环境部署的应用版本是 1.0.1 的时候，我们就选择 1.0.1 版本的测试用例；而当被测环境部署的应用版本是 1.0.2 的时候，我们就相应地选择 1.0.2 版本的测试用例。

所以，我们就在这个统一的测试执行平台中，引入了这种形式的测试用例版本控制机制，直接根据被测应用的版本自动选择对应的测试用例版本。

第二，提供基于 Restful API 的测试执行接口供 CI/CD 使用。这样做的原因是，测试执行平台的用户不仅仅是测试工程师以及相关的产品经理、项目经理，很多时候 CI/CD 流水线才是主力用户。因为，在 CI/CD 流水线中，每个阶段都会有不同的发起测试执行的需求。

我们将测试基础架构与 CI/CD 流水线集成的早期实现方案是，直接在 CI/CD 流水线的脚本中硬编码发起测试的命令行。这种方式最大的缺点在于灵活性差：

当硬编码的命令行发生变化，或者引入了新的命令行参数的时候，CI/CD 流水线的脚本也要一起跟着修改；

当引入了新的测试框架时，发起测试的命令行也是全新的，那么 CI/CD 流水线的脚本也必须被一起改动。

因此，为了解决耦合性的问题，我们在这个统一的测试执行平台上，提供了基于 Restful API 的测试执行接口。任何时候你都可以通过一个标准的 Restful API 发起测试，CI/CD 流水线的脚本也无须再知道发起测试的命令行的具体细节了，只要调用统一的 Restful API 即可。

如图 2 所示，就是引入了统一测试执行平台的测试基础架构。

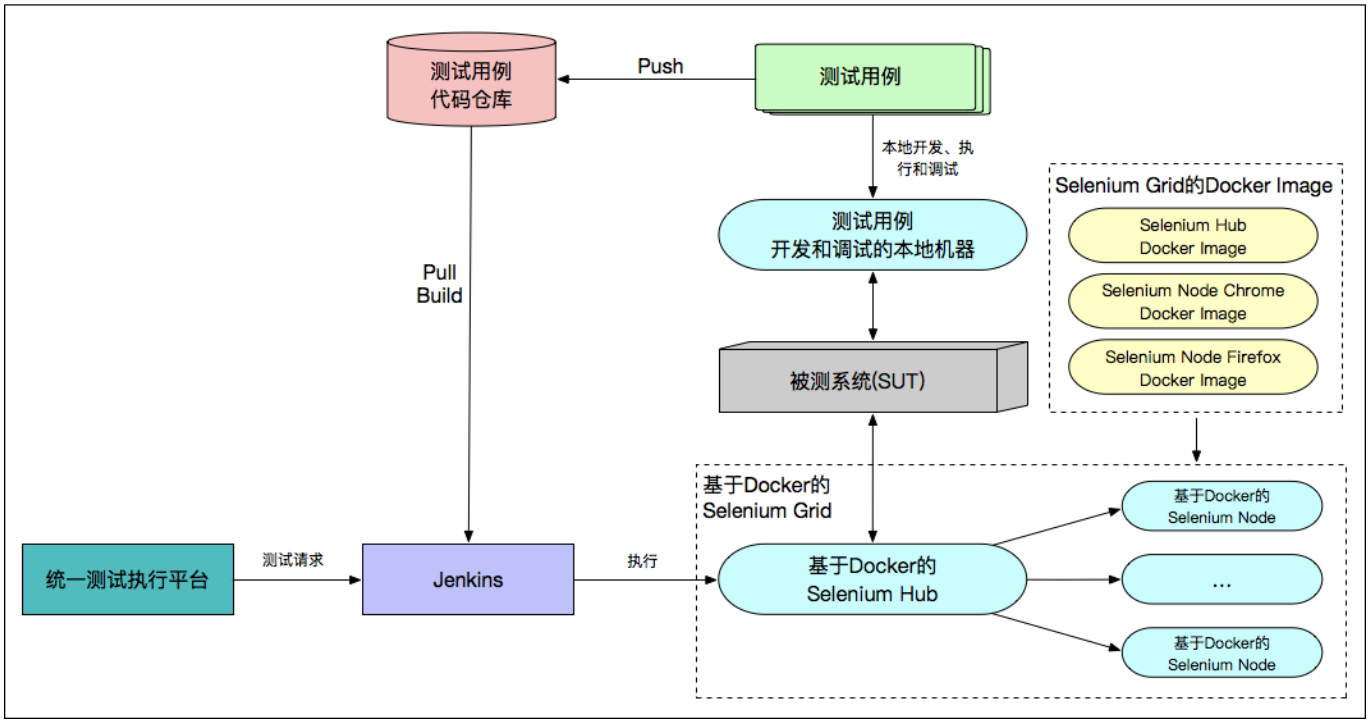

图 2 引入统一测试执行平台的测试基础架构

基于 Jenkins 集群的测试基础架构

这个引入了统一测试执行平台的测试基础架构，看似已经很完美了。但是，随着测试需求的继续增长，又涌现出了新的问题：单个 Jenkins 成了整个测试基础架构的瓶颈节点。因为，来自于统一测试执行平台的大量测试请求，会在 Jenkins 上排队等待执行，而后端真正执行测试用例的 Selenium Grid 中很多 Node 处于空闲状态。

为此，将测试基础架构中的单个 Jenkins 扩展为 Jenkins 集群的方案就势在必行了。如图 3 所示，就是基于 Jenkins 集群的测试基础架构。

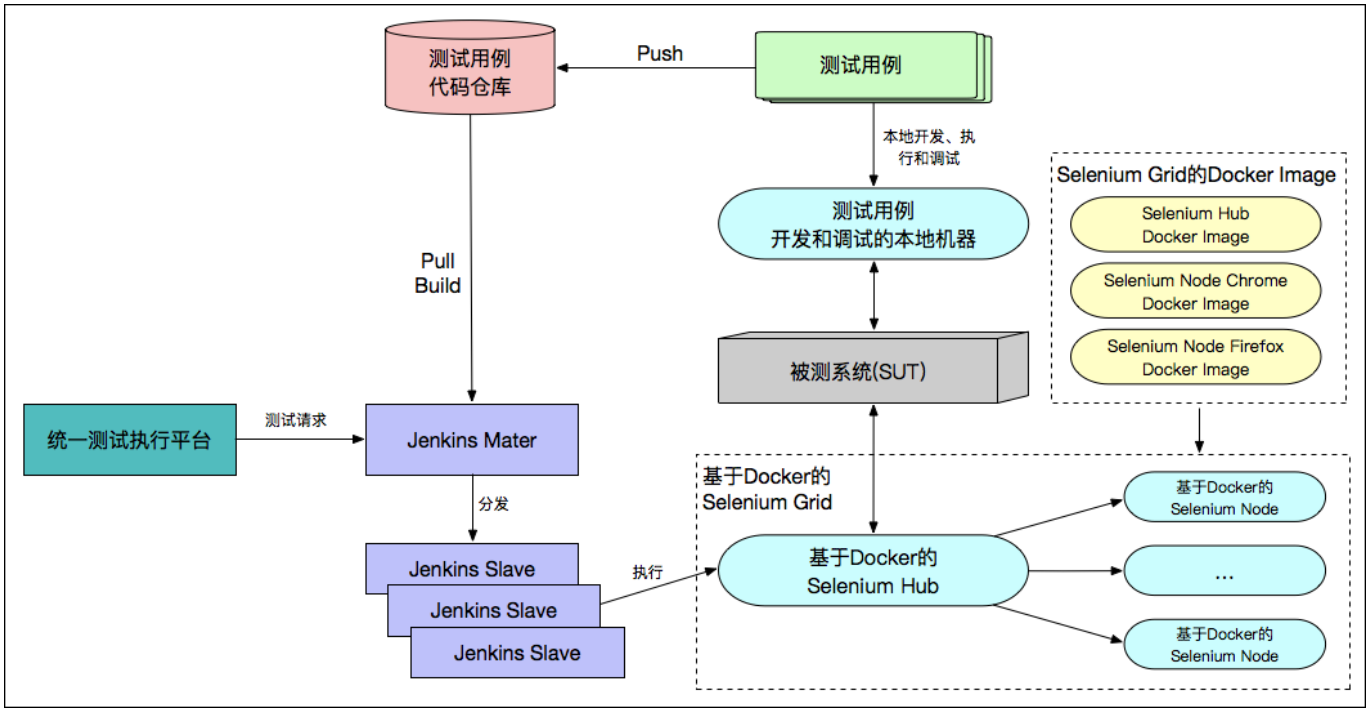

图 3 基于 Jenkins 集群的的测试基础架构

因为 Jenkins 集群中包含了多个可以一起工作的 Jenkins Slave，所以大量测试请求排队的现象就再也不会出现了。

而这个升级到 Jenkins 集群的过程中，对于 Jenkins 集群中 Slave 的数量到底多少才合适并没有定论。一般的做法是，根据测试高峰时段 Jenkins 中的排队数量来预估一个值。通常最开始的时候，我们会使用 4 个 Slave 节点，然后观察高峰时段的排队情况，如果还是有大量排队，就继续增加 Slave 节点。

测试负载自适应的测试基础架构

引入了 Jenkins 集群后，整个测试基础架构已经很成熟了，基本上可以满足绝大多数的测试场景了。但是，还有一个问题一直没有得到解决，那就是：Selenium Grid 中 Node 的数量到底多少才合适?

> 如果 Node 数量少了，那么当集中发起测试的时候，就会由于 Node 不够用而造成测试用例的排队等待，这种场景在互联网企业中很常见；

而如果 Node 数量多了，虽然可以解决测试高峰时段的性能瓶颈问题，但是又会造成空闲时段的计算资源浪费问题。当测试基础架构搭建在按使用付费的云端时，计算资源的浪费就是资金浪费了。

为了解决这种测试负载不均衡的问题，Selenium Grid 的自动扩容和收缩技术就应运而生了。

Selenium Grid 的自动扩容和收缩技术的核心思想是，通过单位时间内的测试用例数量，以及期望执行完所有测试的时间，来动态计算得到所需的 Node 类型和数量，然后再基于 Docker 容器快速添加新的 Node 到 Selenium Grid 中；而空闲时段则去监控哪些 Node 在指定时间长短内没有被使用，并动态地回收这些 Node 以释放系统资源。

通常情况下，几百乃至上千台 Node 的扩容都可以在几分钟内完成，Node 的销毁与回收的速度同样非常快。

至此，测试基础架构已经演变得很先进了，基本可以满足大型电商的测试执行需求了。测试负载自适应的测试基础架构，具体如图 4 所示。

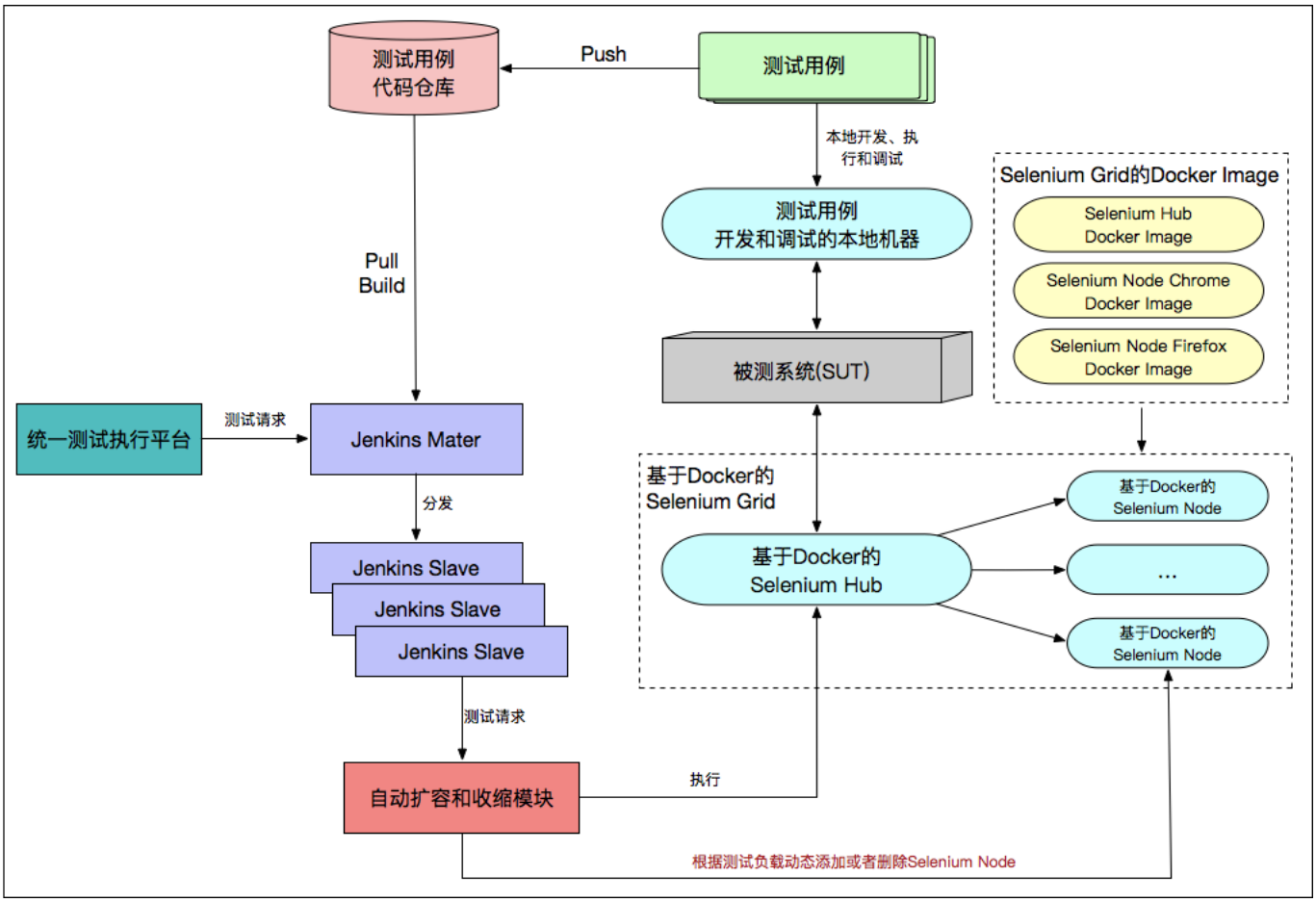

图 4 测试负载自适应的测试基础架构

如何选择适合自己的测试基础架构?

现在，我已经介绍完了测试基础架构的演进，以及其中各阶段主要的架构设计思路，那么对于企业来说，应当如何选择最适合自己的测试基础架构呢?

其实，对于测试基础架构的建设，我们切忌不要为了追求新技术而使用新技术，而是应该根据企业目前在测试执行环境上的痛点，来有针对性地选择与定制测试基础架构。

比如，你所在的企业如果规模不是很大，测试用例执行的总数量相对较少，而且短期内也不会有大变化的情况，那么你的测试基础架构完全就可以采用经典的测试基础架构，而没必要引入 Docker 和动态扩容等技术。

再比如，如果是大型企业，测试用例数量庞大，同时还会存在发布时段大量测试请求集中到来的情况，那么此时就不得不采用 Selenium Gird 动态扩容的架构了。而一旦要使用动态扩容，那么势必你的 Node 就必须做到 Docker 容器化，否则无法完全发挥自动扩容的优势。

所以说，采用什么样的测试基础架构不是由技术本身决定的，而是由测试需求推动的。

总结

在今天这篇文章中，我从测试基础架构演进的视角，和你分享了测试基础架构发展的前世今生。

首先，为了降低测试用例过多时 Selenium Grid 的维护成本，我们用 Docker 容器代替了经典测试基础架构中的实体机 / 虚拟机，形成了基于 Docker 实现的 Selenium Grid 测试基础架构。

而后，我们发现测试用例的数量达到一定规模后，管理和执行发起测试的 Jenkins Job 成了问题。于是，我们引入了一个基于 GUI 界面的测试执行平台，并在其上扩展了诸如测试用例版本化、提供基于 Restful API 的测试执行接口等功能。从而，形成了由这个统一测试执行平台发起测试的测试基础架构形态。

而为了进一步解决由单个 Jenkins 带来的系统瓶颈问题，我们过渡到了基于 Jenkins 集群的测试基础架构，通过多个同时工作的 Jenkins Slave，解决了大量测试请求排队的问题。

随后，为了解决 Selenium Grid 中 Node 的数量到底多少才合适的问题，我和你谈论了创新设计的 Selenium Grid 的自动扩容和收缩技术。至此，测试负载自适应的测试基础架构也终于“千呼万唤始出来”了。

最后，我谈论了不同企业该如何选择最适合自己的测试基础架构的问题。这里，我强调了一定要根据测试需求选择测试基础架构，而不能一味地追求最新的技术。

我希望通过这种授人以渔的方式，可以帮你拓宽思路，并将测试基础架构的设计思路、思想，运用到你的实际工作中去。

思考题

其实，在我讲述的这个测试基础架构的设计和搭建过程中，还有很多可以优化和创新的点。你觉得哪些部分可以再优化呢？你还有哪些想法呢？

欢迎你给我留言，我们一起讨论。

上一篇　40 | 从小工到专家：聊聊测试执行环境的架构设计（上）

下一篇　42 | 实战：大型全球化电商的测试基础架构设计

精选留言 (13)

写留言

无 心
2018-10-08

1

CI 持续集成，在工作环境中，确实很方便，跟踪项目质量测试结果，敏捷测试 快速的迭代

Robert小七
2019-05-22

现在的公司用的基础架构和这个很像，但是多了基础用例，配置管理，mock，性能，等。可以算是一个真正的devops平台了！

口水窝
2019-05-16

看完了演进的过程，有种酣畅淋漓，大快人心的感觉，在技术飞速发展分时代，作者能够把这些事件的东西通过原理的形式展现出来，点赞！再有疑问，对于这些演进的原材料，是从官方文档了解来的吗？

展开

楚耳
2019-04-21

老师这套环境只能支持基于seleium的ui脚本执行，换其它脚本就行不通了，如果是自己基于python写的接口脚本呢，有什么好的分布式执行框架吗？

Lily
2019-03-04

sqlmap支持其他类型的数据库hive、Libra、hadoop这些吗？

展开

年轻人的瞎...
2019-01-14

想知道ci，cd在这里面扮演什么角色，会在测试框架那个流程使用，可以在哪里看相关内容，最近工作最新有用到

小老鼠
2018-11-29

1、你们这个平台是开源的吗？若是我想要份
2、selenium 我不太喜欢，里面问题好多。比如IE浏览器对于复杂页面，比如淘宝、京东首页无法获得元素，同样Chrome、FireFox就没问题

展开

作者回复: 没有开源，目前看selenium还是比较靠谱的方案

胖虫子
2018-11-15

一般能用jenkins就狠很可以了，要到最后两层，得很大的企业才行

展开

夏洛克的救...
2018-11-05

如何理解挡板测试呢?

展开

才子
2018-10-21

老师说的这个，对于接口测试也是这个思路吗？在接口测试中有类似selenium grid的分布式框架吗？

Struggling
2018-10-07

CI-CD这块老师能不能再深入讲解一下，比如有没有用到可视化的工具来集成

展开

作者回复: cicd并不是这个专栏的重点内容，而且cicd不是靠一个工具就可以实现的，都是完整的一套工具链体系。

_CountingS...

2018-10-01

我觉得可以整合现在大热的k8s实现自动扩缩容

展开

作者回复: 是的，k8s的确是个很棒的方案，但是我们没有采用的主要原因在于测试执行集群的我规模并没有大到非k8s不可，实施k8s的投入还是比较大的，如果后续集群规模继续扩大，我们就会考虑引入k8s，总之一句话就是工程上我们不求最先进，只求够用就好的原则

Robert小七

2018-10-01

这一套架构估计能用到的企业也不多了

展开

作者回复: 大的互联网企业基本都是必须要有的，否则很难支持快速的迭代与发布