33 | 无实例无真相：基于LoadRunner实现企业级服务器端性能测试的实践（下）

2018-09-12 茹炳晟

讲述：茹炳晟
时长 19:08 大小 8.77M

你好，我是茹炳晟。今天我和你分享的主题是：无实例无真相之基于 LoadRunner 实现企业级服务器端性能测试的实践（下）。

今天，我会继续和你分享如何基于 LoadRunner 完成企业级服务器端的性能测试。通过我上一次的分享，你已经清楚知道了，整个性能测试过程可以分为五个阶段，并且解决了整个测试过程中最难的一部分工作，即如何获取具体的性能测试需求。

现在，我们先来回顾一下，性能测试包含的五个阶段：性能需求收集以及负载计划制定、录制并增强虚拟用户脚本、创建并定义性能测试场景、执行性能测试场景，以及分析测试报告。所以，今天，我们就要解决剩下的 4 个阶段的问题了。

阶段 2：录制并增强虚拟用户脚本

我已经在上篇文章中和你提到，完成了性能测试需求分析后，你就已经明确了要开发哪些性能测试脚本。现在，我们就一起来看看开发性能测试脚本的步骤，以及相关的技术细节。

从整体角度来看，用 LoadRunner 开发虚拟用户脚本主要包括以下四个步骤：

1. 识别被测应用使用的协议；
2. 录制脚本；
3. 完善录制得到的脚本；
4. 验证脚本的正确性。

这里需要注意的是，完善录制得到的脚本这一步，会包含大量的技术细节，也有很多对你来说可能是新概念的名词，所以我会着重讲解这一步，帮你克服性能测试道路上的这些“拦路虎”。

步骤 1：识别被测应用使用的协议

如果你已经和系统设计、开发人员沟通过，明确知道了被测系统所采用的协议，那么你可以跳过这一步。如果还不知道具体使用的哪种协议的话，你可以使用 Virtual User Generator 模块自带的 Protocol Advisor 识别被测应用使用的协议，具体的操作方法也很简单：

1. 在 Virtual User Generator 中依次点击 File、Protocol、AdvisorAnalyze、Application，展开这些菜单。
2. 在打开的界面上按要求填写被测应用的信息。
3. Protocol Advisor 会自动运行被测系统。如果是网页应用，就会打开浏览器。
4. 在页面上执行一些典型的业务操作，完成这些业务操作后点击 "Stop Analyzing" 按钮停止录制。
5. Protocol Advisor 会根据刚才录制的内容自动分析被测应用使用的协议，并给出最终的建议。

接下来，你就可以使用 Protocol Advisor 建议的录制协议开始脚本录制工作了。如图 1 所示就是 Protocol Advisor 给出的建议录制协议界面。

图 1 Protocol Advisor 给出的建议录制协议界面

步骤 2：录制脚本

脚本录制的基本原理是，通过 GUI 界面对被测系统进行业务操作，Virtual User Generator 模块在后台捕获 GUI 操作所触发的客户端与服务器端的所有交互，并生产基于 C 语言的虚拟用户脚本文件。

也就是说，录制脚本的过程需要通过 GUI 实际执行业务操作，所以我建议你在开始录制脚本前，先多次演练需要这些 GUI 操作步骤，并明确知道哪些操作步骤会对服务器端发起请求。

我们要知道哪些操作步骤会对服务器发起请求的原因是，要将这些操作步骤在虚拟用户脚本中封装成“事务”（Transaction）。封装为“事务”的目的是统计响应时间，因为 LoadRunner 中的响应时间都是以“事务”为单位的。

具体的录制步骤，主要包括如下三步，

1. 首先，选择 Create/Edit Scripts 进入 Virtual User Generator 创建脚本的协议选择界面。
2. 选择正确的协议后进入 Start Recording 界面，选择需要录制的应用类型，并填写应用的详细信息。如果是 Web 应用，Application type 就应该选择 Internet Application，然后选择浏览器并填写这个 Web 应用的 URL，完成后自动打开浏览器。

3. 在该浏览器中执行业务操作，Virtual User Generator 模块会记录所有的业务操作，并生成脚本。

在录制脚本的过程中，我强烈建议直接对发起后端调用的操作添加事务定义，而不要等到脚本生成后再添加。因为 LoadRunner 脚本的可读性并不好，在录制完的脚本中添加事务定义的难度会很大。

在录制过程中，直接添加事务操作也很简单，主要包括如下三步：

1. 在开始执行 GUI 操作前，先点击图 2 中的“事务开始”按钮并填写事务名称；
2. 执行 GUI 操作；
3. 操作完成后，点击图 2 中的“事务结束”按钮。

这样你刚才执行 GUI 操作的脚本就会被 lr_start_transaction(“事务名称”) 和 lr_end_transaction(“事务名称”，LR_AUTO) 包围起来，也就完成了添加事务的定义。

图 2 Virtual User Generator 的脚本录制控制条

步骤 3：完善录制得到的脚本

脚本录制，只是虚拟用户脚本开发中最简单的一步。我在上一次分享《无实例无真相：基于 LoadRunner 实现企业级服务器端性能测试的实践（上）》时，提到由 Virtual User Generator 模块录制的脚本不能直接使用，我们还需要对录制的脚本做以下处理：

在两个事务之间加入思考时间（Think Time）；

对界面输入的数据做参数化（Parameterization）操作；

完成脚本的关联（Correlation）操作；

加入检查点（Check Point）。

这 4 步处理操作是虚拟用户脚本开发中最关键的地方，你不仅需要知道为什么要进行这些处理，更要能够完成这些处理，否则你录制的脚本无法成功回放。

第一，在两个事务之间加入思考时间

什么是思考时间呢?

用户在实际使用系统时，并不会连续不断地向后端服务器发起请求，在两次发起请求之间往往会有一个时间的间隔，这个时间间隔主要来自于两个方面：

一是，用户操作的人为等待时间，因为用户不可能像机器人那样快速地执行操作；

二是，用户可能需要先在页面上填写很多信息后之后，才能提交操作，那么填写这些信息就需要花费一定的时间。

所以，为了让虚拟用户脚本能够更真实地模拟实际用户的行为，我们就需要在两个事务之间加入一定的等待时间。这个等待时间，就是 LoadRunner 中的思考时间。

你只要直接调用 LoadRunner 提供的 lr_think_time() 函数，就可以在两个事务之间加入思考时间。但是，这个思考时间到底设置为多少，并没有那么容易知道。思考时间往往会涉及多方面的因素，严格计算的话会非常复杂。

所以，**在实际项目中，一般先粗略估计一个值（比如 15 s），然后在实际执行负载场景的过程中，再根据系统吞吐量调整。**

你在后续调整思考时间时，无需逐行修改虚拟用户脚本代码，可以在 Run-time Settings（运行时设置）中很方便地完成。如图 3 所示，Run-time Settings 中支持多种方式调整思考时间。

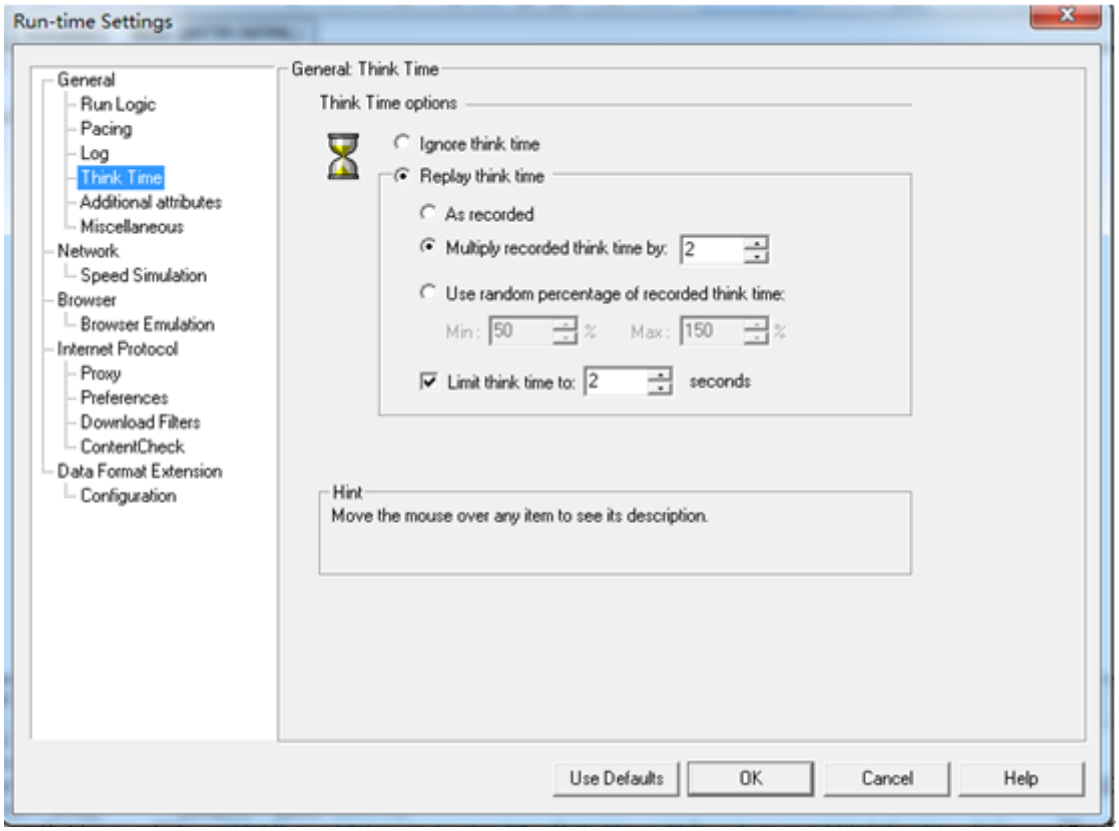

图 3 通过 Run-time Settings 统一调整思考时间

As recorded，代表的是直接使用 lr_think_time() 函数中指定的时间。

Mutiply recorded think time by，代表的是在 lr_think_time() 函数中指定的时间基础上乘以一个数字。比如这个数字是 2，那么所有的思考时间都会翻倍。

Use random percentage of recorded think time，指的是使用指定思考时间范围内的随机值。例如，如果 lr_think_time() 函数中指定的时间是 2 s，并且指定最小值为 50%，最大值为 200%，则实际的思考时间会取最小值 1 s（2 s50%）和最大值 4 s（2 s200%）之间的随机值。

Limit think time to，指的是为思考时间设置一个上限值，只要 lr_think_time() 函数中指定的时间没有超过这个上限值，就按照 lr_think_time() 函数指定的值，如果超过了就取这个上限值作为思考时间。

第二，对界面输入的数据做参数化操作

数据的参数化，其实很好理解，我再给你举个例子，你马上就能明白。

假设，你录制的虚拟用户脚本完成的是用户登录操作，那么由于脚本回放时需要支持多用户的并发，所以必须要把脚本中的用户名和密码独立出来，放入专门的数据文件中，然后在这

个文件中提供所有可能被用到的用户名和密码。

有没有感觉这个概念很熟悉，它其实和我以前介绍到的数据驱动的自动化测试完全相同。

图 4 给出了参数化配置的界面截图，LoadRunner 支持的参数化的数据源很丰富，既可以是 excel 文件，也可以是数据库中的表等。

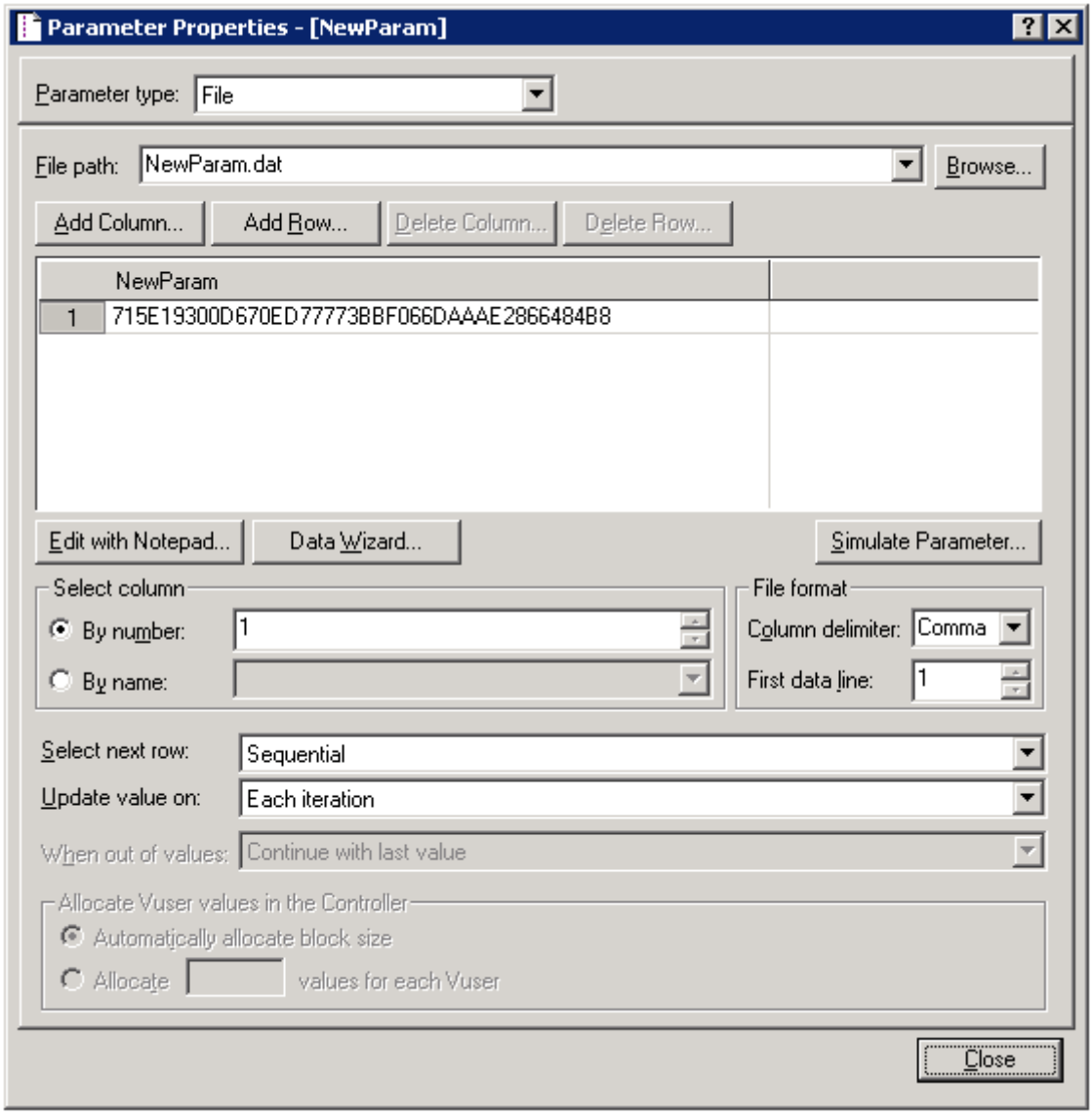

图 4 虚拟用户脚本参数化配置的界面截图

这里需要特别说明的是，凡是参数文件中使用的测试数据都需要在执行性能测试前，在被测系统中事先准备好。比如，还是以用户登录的脚本为例，假定你的参数文件中提供了 5000 个用于并发执行的用户信息，那么这 5000 个用户必须是已经实际存在于系统中的，这就要求你要在开始测试前事先准备好这 5000 个用户。

所以，参数化操作其实由两部分组成：

1. 性能测试脚本和测试数据的分离；
2. 事先建立性能测试的数据。

也就是说，参数化的过程往往与性能测试数据准备密不可分。

第三，完成脚本的关联操作

关联操作，是 LoadRunner 虚拟用户脚本开发过程中最关键的部分，直接关系到脚本是否可以回放成功。

从概念上讲，关联的主要作用是，取出前序调用返回结果中的某些动态值，传递给后续的调用。是不是听起来很拗口，不太好理解？我们来看一个具体的例子吧。

假设，每次客户端连接服务器端时，服务器端都会用当前的时间戳（Time Stamp）计算 CheckSum，然后将 Time Stamp 和 CheckSum 返回给客户端。然后，客户端就把 Time Stamp + CheckSum 的组合作为唯一标识客户端的 Session ID。录制脚本时，录制得到的一定是硬编码（hardcode）的 Time Stamp 值和 CheckSum 值。

图 5 展示了这个交互过程，录制得到 Time Stamp 的值是 TS，而 CheckSum 的值是 CS。

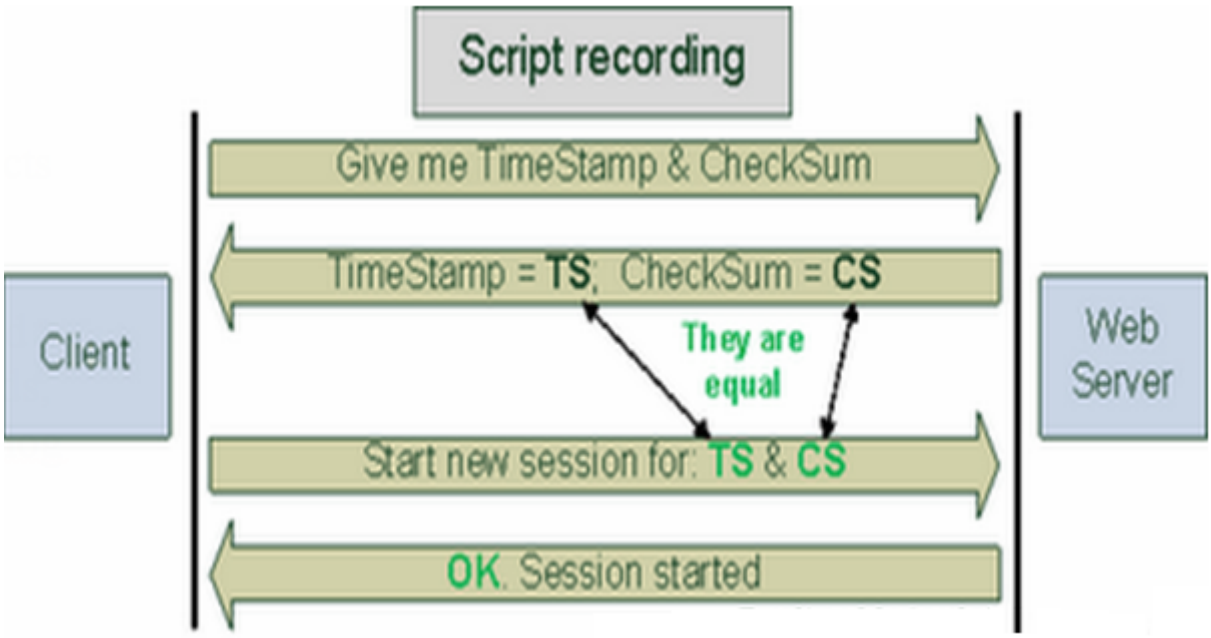

图 5 关联原理图 - 脚本录制过程

采用 Time Stamp + CheckSum 的组合作为 Session ID 的方式，在我们回放这个脚本的时候就有问题了。因为回放时，这段硬编码已经有了新的 Time Stamp 值和 CheckSum 值，并且显然与之前的值不同，所以服务器无法完成 Session ID 的验证，也就导致了脚本回放失败。

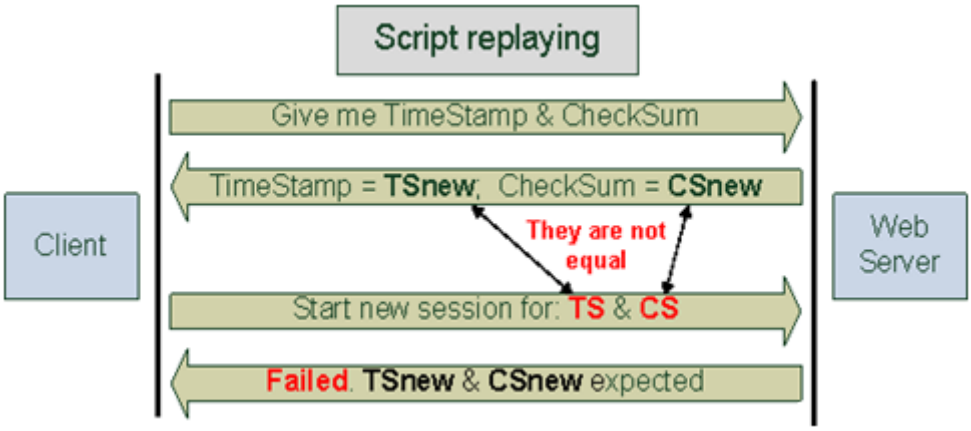

图 6 关联原理图 - 脚本回放过程

其实，这种情况几乎存在于所有的虚拟用户脚本中，所以我们必须要解决这个问题。

解决方法就是，在脚本回放的过程中，实时抓取 Time Stamp 值和 CheckSum 值，然后用实时抓取到的值替换后续需要使用这两个值的地方。这个过程就是“关联”。

如图 7 所示，**关联就是解析服务器端的返回结果，抓取新的 Time Stamp 值和 CheckSum 值，然后后续的操作都使用新抓取的值，这样脚本就能回放成功了。**

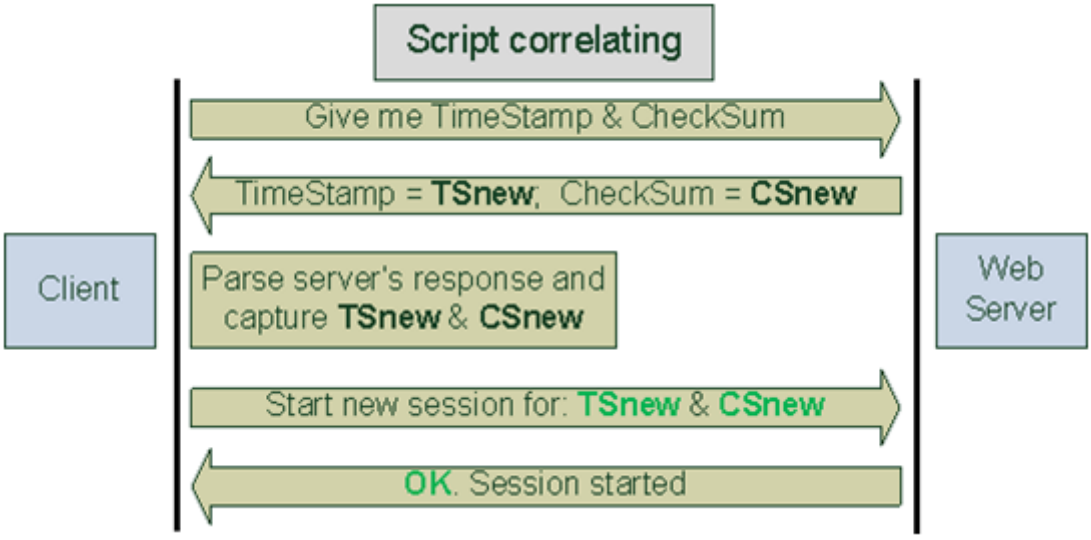

图 7 关联原理图 - 使用“关联”后的脚本回放过程

理解了关联操作，在脚本中处理关联就比较简单了，LoadRunner 提供了功能强大的关联函数 web_reg_save_param()。这个关联函数支持多种动态值的获取方式，用得最多的是基于“前序字符串匹配”加上“后续字符串匹配”的方式。其中，字符串匹配，支持正则表达式。

我们一起来看个具体的例子吧。

假设，服务器端返回的结果是“LB=name=timestamp value=8888.LB=name=CheckSum”，那么为了能够获取到“8888”这个动态值，我们就可以用“前序字符串 =LB=name=timestamp value=”和“后续字符 =.LB=name=CheckSum”来“框出” 8888”这个动态值。

另外，需要特别注意的是 web_reg_save_param() 函数是注册型函数，必须放在获取动态值所属的请求前面，相当于先声明，后调用。

更多的关联函数用法，你可以参考 LoadRunner 官方文档。

第四，加入检查点

检查点，类似于功能测试中的断言。但是，性能测试脚本，不像功能测试脚本那样需要加入很多的断言，往往只在一些关键步骤后加入很少量的检查点即可。这些检查点的主要作用是，保证脚本按照原本设计的路径执行。

最常用的检查点函数是 web_reg_find()，它的作用是通过指定左右边界的方式“在页面中查找相应的内容”。这里需要注意的是，这个函数也是注册型函数，即需要放在所检查的页面之前，否则会检查失败。更多的检查点函数以及用法也请参考 LoadRunner 官方文档。

步骤 4：验证脚本的正确性

完成了脚本开发后，根据我的个人经验，我强烈建议你按照以下顺序检查脚本的准确性：

1. 以单用户的方式，在有思考时间的情况下执行脚本，确保脚本能够顺利执行，并且验证脚本行为以及执行结果是否正确；
2. 以单用户的方式，在思考时间为零的情况下执行脚本，确保脚本能够顺利执行，并且验证脚本行为以及执行结果是否正确；

3. 以并发用户的方式，在有思考时间的情况下执行脚本，确保脚本能够顺利执行，并且验证脚本行为以及执行结果是否正确；
4. 以并发用户的方式，在思考时间为零的情况下执行脚本，确保脚本能够顺利执行，并且验证脚本行为以及执行结果是否正确。

只有上述四个测试全部通过，虚拟用户脚本才算顺利完成。

至此，我们完成了第二个阶段的“录制并增强虚拟用户脚本”的工作，顺利拿到了虚拟用户脚本。那么接下来，我们就会进入第三个阶段，使用开发完成的虚拟用户脚本创建并定义性能测试场景。

阶段 3：创建并定义性能测试场景

还记得我在分享《工欲善其事必先利其器：后端性能测试工具原理与行业常用工具简介》这个主题时，介绍过的性能测试场景的内容吗？如果有点忘记了，我建议你先回顾一下这篇文章的内容。

这个阶段的工作，就是在 LoadRunner Controller 中设置性能测试场景。由于整个设置过程，都是基于 Controller 的图形用户界面的操作，本身没什么难度，所以我就不再详细展开了，如果有这方面的问题，你也可以自行百度或者给我留言。

阶段 4：执行性能测试场景

完成了性能测试场景的设计与定义后，执行性能测试场景就非常简单了。

这个过程一般是在 LoadRunner Controller 中完成。你可以通过 Controller 发起测试、停止测试、调整性能测试场景的各种参数，还可以监控测试的执行过程。

阶段 5：分析测试报告

执行完性能测试后，LoadRunner 会根据自己的标准并结合性能测试场景中定义的系统监控器指标，生成完整的测试报告。在 Analysis 中，不仅可以以图形化的方式显示单个指标，也可以将多个指标关联在一起进行比较分析。

图 8 展示了使用 LoadRunner Analysis 展示事务平均响应时间的界面，我们可以看到图片右下角各个事务的最小响应时间、最大响应时间和平均响应时间。

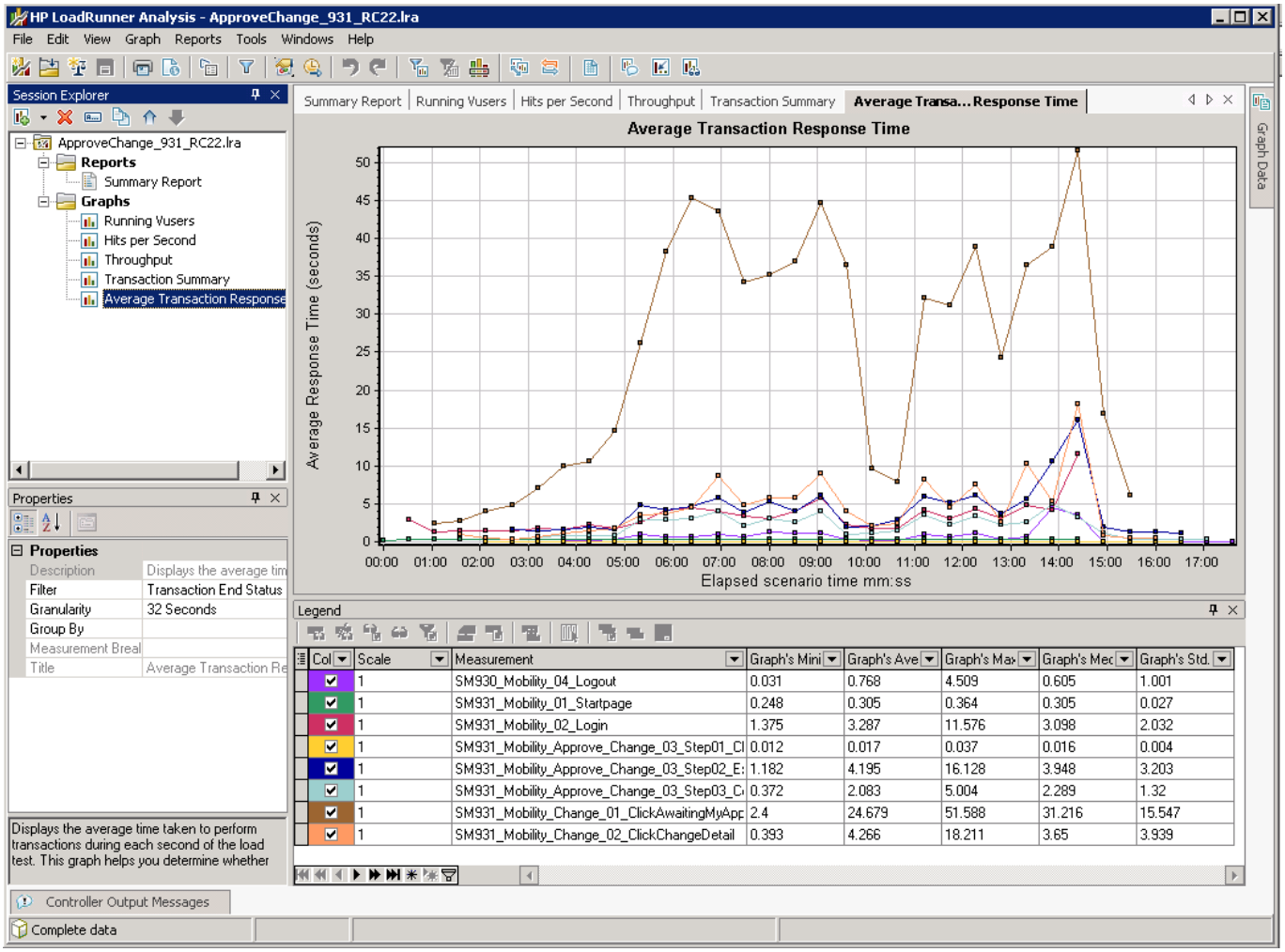

图 8 性能测试报告的分析

性能测试报告的分析，是一项技术含量非常高的工作。优秀的性能测试工程，通过报告中的数值以及数值之间的相互关系，就能判断出系统中可能存在的问题。这就好比医生看验血报告，经验丰富的医生可以根据验血报告对病情做出八九不离十的判断。

性能测试报告的解读，需要丰富的系统架构、性能理论以及大量实战经验的积累。这个话题已经超出了我今天要分享的范围，所以我也就不再继续展开了。

总结

今天接着上一篇文章，我和你分享了企业级后端性能测试的后四个阶段的内容，包括录制并增强虚拟用户脚本、创建并定义性能测试场景、执行性能测试场景，以及分析测试报告。现在，我再为你总结一下每一个阶段的重点内容。

录制并增强虚拟用户脚本，这个阶段的工作又可以分为识别被测应用使用的协议、录制脚本、完善录制得到的脚本、验证脚本的正确性四步。其中，完善录制得到的脚本这一步，涉

及到了很多概念和基础知识，所以我进行了重点讲解，希望帮你克服性能测试的难点。

创建并定义性能测试场景，以及执行性能测试场景，这两个阶段的工作都是在 LoadRunner 的 Controller 模块中完成的，也都比较简单。你可以参考我在《工欲善其事必先利其器：后端性能测试工具原理与行业常用工具简介》这篇文章分享的内容，完成这两个阶段的工作。

分析测试报告，这个工作的技术含量非常高。深入解读性能测试报告的能力，需要丰富的系统架构、性能理论，以及大量实战经验。所以，我们需要在平时工作中，不断地丰富自己的知识体系。

思考题

你们公司的性能测试是否使用 LoadRunner，在使用过程中遇到了什么难题？你们又是如何解决的呢？

感谢你的收听，欢迎你给我留言。

上一篇 33-实战案例篇：基于LoadRunner实施企业级服务器端性能测试的实践（上）
下一篇 34 | 站在巨人的肩膀：企业级实际性能测试案例与经验分享

精选留言 (15)

写留言

海 3
2018-09-12

老师，是否可以写一篇关于性能测试报告的分析和性能瓶颈的定位过程?

展开

作者回复: 这个我之前在做大纲的时候有想过专门来个完整分析过程的文章，但是后来放弃了，主要原因是这样的文章的确可以给大海带来性能分析到位主观感受，但是最大的问题是知识点非常零碎，而且不管用什么例子都不具有代表性，而且会需要一个实际的系统以及很多log的支持，会牵涉很多敏感信息，所以最后没有成文。不过可以在留言区多多讨论

海 2
2018-09-13

老师所有的文章，一路看过来每次都是意犹未尽，获益良多。这么好的文章，看一遍怎么够呢！非常感谢老师的辛苦用心。

伪专家 2
2018-09-12

现在公司都用jemter，也是jemter2次开发，又叫全链路压测平台

展开

作者回复: 是的，大型互联网企业特有的线上环境的全链路压测，牵涉到流量隔离，影子表，api改造，jmeter海量并发的二次开发等内容，是个专业性很高的领域

Xiye 1
2018-09-13

我们公司没有用Loadrunner，用得是Jmeter。在使用中，会有压力不够的问题，我们用一台机器做控制器，三台或四台机器做Agent端测试，经常一台机器超过200线程，性能就上

不去了，检查发现瓶颈在于测试的Agent端。特别是我们项目转到Sprint boot架构下，这个问题就更加明显了。后来其他项目的同事推荐我们使用wrk，确实能提高了不少测试压力。 ...

展开

Sunshine

2018-09-13

1

老师可以介绍一下，如何去实现每隔10s增加100个用户的方法吗？上一篇提的一些场景不太明白如何去做。感谢🙏

Cynthia🌸...

2018-09-12

1

对于这篇里面提到的录制脚本等相关工作自己还是比较熟悉的。而对于性能测试的理解和整体把握上还不够成体系。
需要好好琢磨文章，实践，总结，实践。

展开

作者回复: 性能测试系列的最后一篇文章可能会给你很多体系上的理解和帮助

口水窝

2019-04-26

想起4年前在前面一个公司的时候用过，但不知其中的原理，只是依葫芦画瓢，感觉不会写代码，不会百度，执行不起来，不会找原因，然后就没在看了，那时候可羡慕运行起来的高手了，现在在回头看，其实编程也不是很难，关键我们没有找对方向，没有找对下手点，以致于自己一直在这个范围之外彷徨！
这几年冷静下里，想了想，生娃带娃的两年，逐渐使我慢慢明朗起来，要知道怎么去学...

展开

李书红

2019-02-09

我们的性能测试脚本一般都采用java vuser协议，在lr的原生api上封装了个小框架，以屏蔽lr的诸多技术细节。用录制方式得到的脚本是c语言的，不好维护，遇到报文加解密和签名验签的场景就更尴尬了。

展开

人心向善

2018-11-27

检查点的作用说白了就是验证正确性，比如拿登录这个点来说：只有当用户成功登录系统后才会出现“welcome！admin”的这种提示，那么检查点就可以以“welcome！admin”这些信息作为check，原因就是因为只有用户登录后才会看到这个提示，不登录是看不到的！如果说页面上实在没有比较容易找到的检查点信息，可以以200 ok为检查点或者是具体返回值信息比如 返回success 0这些信息也可以，不过并不建议这样做

展开

作者回复: 性能测试脚本理论上是需要每一步都加检查点的，但是实际工程中这么做的人很少，只在一些关键路径加

落恒

2018-09-13

有个个问题想请教一下老师：
有使用过LR12录制过测试脚本，录制的脚本会把所有的请求都自动生成对脚本。想问一下这些请求图片/css样式url的是否会对服务器真实压测结果产生影响，是否需要删除无关的url

赵明月

2018-09-12

也想看jmeter的

展开

作者回复: 其实jmeter的功能相比loadrunner还是差了点的，但是由于免费，所以用的比较多，后续我们讲全链路压测的时候就会以jmeter来讲

伪专家

2018-09-12

测试迷茫

展开

伪专家

2018-09-12

招聘测试一般条件都要有测试工具开发经验...测试的基础，质量保证...质量度量过程都不是关注点...测试要的基础知识点比开发还要全面...工资待遇...重视程度...从目前的招聘信息对比差距越来越大.....

展开

Robert小七

2018-09-12

我看老师好像还是用的LR11，是否从这里可以看出后续jmeter是主流！LR只是利于理解

> 作者回复: 其实不完全是，lr的很多理念和方法都是比jmeter要先进很多的，当然最为概念讲解的辅助工具也可以更好的突出重点。

牛鬼蛇神VS...

2018-09-12

晚安

展开