第一章：性能测试基本概念、分类、性能指标

性能测试：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的各项性能指标进行测试。

系统用户数：该系统的注册用户数；在线用户：登录系统的用户数；并发用户：同时对服务器进行操作的用户数。

响应时间：客户端发送请求到接收到服务器端的回应所需要的总时间。吞吐量：单位时间内能够处理的事务数目，也称为TPS（每秒事务数）。事务：一般指要做的或所做的事情。资源使用率：对CPU、内存、网络等资源的使用量

点击率（HPS）：每秒钟内，用户向服务器提交的HTTP请求数；点击率越大，表明对服务器产生的压力也越大。

资源利用率不是越低越好，系统允许的情况下，合理的尽可能多的利用资源，满足系统需要。尽可能大的吞吐量和可以接受的响应时间。

性能测试分类：负载测试、压力测试、容量测试、配置测试、基准测试、并发测试。

负载测试：在一定条件下，运行一种或多种业务，在不同虚拟用户数量的情况下，测试系统性能指标是否在用户的要求范围内。以此确定系统所能承载的最大用户数，以及不同用户数下的系统响应时间及服务器资源利用率等。压力测试：在一定条件下，模拟大量虚拟用户使服务器产生负载，使服务器资源处于极限状态下并长时间连续运行，以测试服务器在高负载情况下是否能够稳定工作。容量测试：在一定条件下，在数据库中构造不同数量级别的数据记录，在一定虚拟用户数量的情况下运行一种或多种业务，获取不同数量级别的服务器性能指标，以确定数据库的最佳容量和最大容量。配置测试：在一定条件下，运行一种或多种业务，在不同虚拟用户数量的情况下，获得不同配置的性能指标，用于选择最佳的设备及参数配置。基准测试：在一定条件下，模拟一定数量虚拟用户运行一种或多种业务，将测试结果作为基线数据，在系统调优过程中，通过运行相同的业务场景比较测试结果，确定调优结果是否达到预期效果。并发测试：通过模拟多用户并发访问同一个应用、存储过程或数据记录及其他并发操作，来测试是否存在死锁、数据错误等故障。

性能测试流程：制定性能测试目标、选择性能测试工具、设计性能测试、监控分析系统、性能调优。

第二章：HTTP协议、抓包工具 URL：统一资源定位符

HTTP协议：一种按照URL指示，将超文本文档从一台主机（web服务器）传输到另一台主机（浏览器）的应用层协议，以实现超链接的功能。

HTTP协议特点：支持客户/服务器模式；简单快捷；灵活；无连接：HTTP协议对于事务处理没有记忆能力。

HTTP主要请求方法：GET和POST。GET：用于获取页面指定信息，主要用于获取网络资源；POST：用于向服务器发送大量数据，主要用于表单提交。

HTTP请求协议：HEAD请求获取由Request-URL所标识的资源的响应信息报头；PUT：请求服务器存储一个资源，并用Request-URL作为其标识；DELETE：请求服务器删除Request-URL所标识的资源；TRACE：请求服务器回送收到的请求信息，朱啊哟用于测试或诊断；CONNECT：保留将来使用；OPTIONS：请求查询服务器的性能，或者查询与资源相关的选项和需求。

HTTP响应分为响应行：协议和状态码、响应头、响应正文：HTML数据。

**1. 1xx：信息提示，表示请求已被成功接收，继续处理。范围：100~101**

**2. 2xx：服务器成功处理了请求。范围：200~206**

**3. 3xx：重定向。访问资源被移动，告诉客户端新资源地址，浏览器重新对信资源发起请求。范围：300~305**

**4. 4xx：客户端错误。如：格式错误，URL不存在等。范围：400~415**

**5. 5xx：服务器错误。服务器内部错误。范围：500~505**

session与cookie区别：

* **Cookie的数据保存在客户端浏览器，Session保存在服务器**
* **服务端保存状态机制需要在客户端做标记，所以Session可能借助Cookie机制**
* **Cookie通常用于客户端保存用户的登录状态**

抓包工具：浏览器自带、Fiddler、HTTPWatch、Charles

为什么需要HTTP协议？

为什么需要抓包？

性能测试工具：LoadRunner、JMeter

LR工具组成：虚拟用户生成器Vitual User Generator：集成开发环境，用于录制业务流程并创建脚本；压力调度和监控中心Controller：框架程序和监控程序，**用于组织、驱动、管理并监控负载测试**；压力结果分析工具Analysis：**数据分析工具，用于查看、剖析和比较性能结果。**

VuGen录制流程：

创建脚本、选择协议、设置录制项、开始录制、插入命令、停止录制。

**如何选择协议—确定方法：**

**通过询问开发人员获知所使用的协议；**

**通过LR自带的“Protocol Advisor”；**

**通过概要或详细设计手册获知所使用的协议；**

* + **录制选项设置**

**录制选项设置：**

**HTML-based script：基于 HTML 的脚本级别**

**URL-based script：基于 URL 的脚本级别**

**代码执行顺序：**

* + **vuser\_init ：初始化**
  + **Action ：具体操作，可重复**
  + **vuser\_end ：相当于析构函数**

思考时间：lr\_think\_time（）；

**局部变量和全局变量：**

**在init、action、end中定义的变量就是局部变量**

**在globals.h中定义的变量是全局变量**

**函数类型说明：**

* + **web:基于HTTP协议（与协议相关的函数）**
  + **lr:通用函数**
  + **lr. :java或.net中使用的函数**
  + **自定义函数**

五、LoadRunner的使用—参数化

1.什么是参数化：用参数替代常量，可以更加真实的模拟实际用户操作并简化脚本。

2.为什么进行参数化：每次使用的数据不同时，参数化更方便脚本执行。

3.怎样进行参数化：右键替换并设置参数属性（参数列表、类型、读取下一行数据的方式，参数值更新方式）；

①鼠标右键参数化方式：先替换常量再建参数化列表；

②建参数化列表方式：先建参数化列表再替换常量。

**参数化中多个参数设置**

* + **参数列表指向同一个文件**
  + **按列名称读取**
  + **参数读取方式，第一个参数设置如之前，第二个参数设置：**

**same line as XXX**

4.每次触发迭代才更新数据，不迭代不更新数据；只有action迭代时，数据才更新；只有run迭代才更新数据；once是无论程序怎么执行，参数值都不更新；随机的做法，让每次读取都是随机的，这样可以保证相对离散。

5.生成大规模数据：Excel拖拽、数据库插入。

6.哪些可以参数化：Tree视图方式下，每条访问记录属性中，各个被填项最后有ABC字样的都可以参数化。

六、LoadRunner的使用—关联、事务、检查点

1.脚本查看方式：

①脚本视图（Script View）：基于文本视图；VuGen将在编辑器中显示带有颜色编码的函数及其变量值的脚本；

②树视图（Tree View）：基于图标的视图；VuGen将在测试树中为录制期间所执行的每一步骤生成一个图标和标题；大多步骤有快照。

2.运行结果查看：

脚本运行后，点击View—Test Results

3.录制飞机订票系统中，使用正确的用户名和密码登录，为什么回放时结果不成功？因为服务器有动态生成的数据返回给客户端。

4.关联：脚本回放过程中，客户端发出请求，通过一些规则，在服务器所响应的内容中查找，得到相应的值，以参数的形式替换录制时的静态值，从而向服务器发出只能却的请求，这种动态获得服务器响应内容的方法被称为关联。

5.什么情况下需要关联：当脚本中的数据每次回放都发生变化时，并且这个动态数据在后面的请求中需要发送给服务器，那么这个内容需要通过关联来询问服务器，获得该数据的变化结果。例如：

登录字符串。带有会话 ID 或时间戳等动态数据的登录字符串

日期/时间戳。使用日期或时间戳或者其他用户凭据的任意字符串

常见前缀：如SessionID或CustomerID

6.怎样进行关联：

①自动关联：最简单，有局限性。常用于在非常标准的动态数据处理中；使用自动关联前，脚本必须要先运行一次。原理：对同一个脚本运行和录制时的服务器返回进行比较，来自动查找变化部分，并且提示是否生成关联。

②边录制边关联：无需操作，系统自带常见应用需要做的关联规则；【Recording Options】—【Correlation】启用选项；适合于提前将关联规则定义好，然后依据其进行关联。

③手动关联：最有效手段，能处理特殊的动态数据；典型实例：论坛中置顶帖子和非置顶帖子中的顶端帖子ID。通过手动关联函数 web\_reg\_save\_param()将想要的字符串保存到一个参数中。 手动关联是关联应用中的最有效手段。

手动关联步骤：

①录制测试脚本，录制两遍（两遍操作内容必须一致）；

②使用脚本比较工具（Tools—compare with script）找出两次脚本的不同，判断是否需要进行关联；

③在日志中查找脚本不同的内容

④在日志中找到查找内容的地方点击鼠标右键，选择“跳至原行”，跳到脚本位置，在该请求前插入关联函数；插入关联函数需要参数名称和左右边界，在日志中找到其左右边界填入参数设置框中

注意：关联函数必须写在被关联数据所在的请求前（或所有请求前），原因：

①发出请求前，申明要捕获的数据边界，这样在数据流返回时，直接过滤；

②否则需要将上一个请求的所有内容保存后，再去单独读取遍历，效率降低。

⑤将动态数据替换成变量名。

⑥验证：查看日志、查看运行结果。

7.事务：模拟用户的一个相对完整的、有意义的业务操作过程。

8.什么情况下使用事务：查看某系列操作的使用时间。

9.怎样使用事务：

①手动输入函数

②使用菜单插入

③录制过程中插入

lr\_start\_transaction("事务名称")：事务开始

lr\_end\_transaction("事务名称","事务状态")：事务结束，结束状态

注意：开始与结束函数必须成对出现；事务的名称必须一样。

10.事务函数：

LR\_AUTO是指事务的状态有系统自动根据默认规则来判断，结果为PASS/FAIL；LR\_PASS是指事务是以PASS状态通过的，说明该事务正确的完成了，并且记录下对应的时间，这个时间就是指做这件事情所需要消耗的时间。

LR\_FAIL是指事务以FAIL状态结束，该事务是一个失败的事务，没有完成事务中脚本应该达到的效果，得到的时间不是正确操作的时间，这个时间在后期的统计中将被独立统计。

LR\_STOP将事务以STOP状态停止。事务的PASS和FAIL状态会在场景的对应计数器中记录，包括通过的次数和事务的响应时间，方便后期分析该事务的吞吐量以及响应时间的变化情况。

11.检查点：检查服务器返回的页面是否正确。

12.为什么使用检查点：脚本运行成功，登录运行通过，但登陆成功后返回的页面不正确。

13.插入检查点函数：

web\_find（）：在HTML页面中查找相应的内容；web\_find函数用于查找HTML页面中的内容，故须放在待查找内容的后面。

web\_reg\_find（）：web\_reg\_find（）函数是在HTML的源文件中查找相应的内容，故需插入在待查找内容之前。

注意：尽量使用web\_reg\_find函数；检查的字符尽量不要是中文。

七、LoadRunner的使用—Controller

1. Controller作用：设计场景（并发用户的设置）、运行场景（分发模拟）、监控场景（服务器的资源）

2.场景主要用来模拟真实世界的用户是如何产生压力的。

3.设计场景流程：新建场景、设计Schedule、添加Load Generator、运行设置、系统监控。

场景类型：

目标场景：设置一个运行目标，通过Controller的Auto Load功能进行自动化负载，如果测试的结果达到目标，则说明性能符合目标，否则LR提示无法达到目标。

手动场景：**自己设置虚拟用户的变化，通过设计用户的添加和减少过程，来模拟真实的用户请求模型，完成负载的生成。**

用户组模式（数字）

用户组模式（百分比）