1. 开场

这是一次同行的聚会，分享一些大家共同关心的技术问题，一起学习和提高。请大家在听的同时也提出你的问题。

正式开始之前，作一个调查，多少人用过 LR？
多少人超过 1 年？
超过 2 年？
超过 3 年？

论坛上的问题集中在以下几种：

- LR 的安装、卸载，license 等等
- 基本概念，使用，比如某个菜单如何设置等，某段代码如何写
- **策略的设置，测试设计，测试场景的组织等**
- **结果的分析和定位**

我们今天重点探讨的不是 1 和 2，而是 3 和 4。

性能测试工具只是工具，重要的还是测试设计，测试的思想。特别是对于性能测试。如果思想不对，再好的工具测试出来的结果也是无用的。

性能测试还有他自己的特殊性，不是一次测试是结束了，是一个不断的迭代不断优化的过程，不像功能测试。而且没有标准的测试结果。

- 理解需求和测试设计

需求的准确理解是性能测试成功的前提
测试设计是性能测试成功的关键。

性能测试的需求一般来说有这么几种分类：

1 模拟大量用户，测试服务器最大负载和性能指标
2 不知道用户规模，测试最大承载用户
3 给定目标事务响应时间，测试最大用户

反应在正式的需求文档中，往往是：

100 用户下，90%的事务平均响应时间不超过 3 秒，最大不超过 10 秒
500 用户下，90%的事务平均响应时间不超过 10 秒，最大不超过 20 秒

有些时候大数据量的情况下页面响应时间、系统资源占用等也是我们性能测试关注的重点。

确定业务流程：
选择典型的业务流程，把需要测试的业务流程的执行步骤形成文档。

1. 使用频率高的
2. 尽量覆盖交易路径的
3. 对系统性能产生影响的

通过一些方法：

- 用户现场调查
- 分析系统日志

等

并发用户数量设计：

- 极限法
- 用户趋势分析法
- 经验评估法

等

系统承载规模：

1. 并发在线用户是实时在线用户的 10%-20%
2. 峰值计算 8/2 原则，80%的人在 20%的时间完成了操作。

举例如下：
某网站统计每天用户数目 10，000，000 人次
峰值时间是 11：00-14：00 用户数目是 4，000，000 人次
一天 86400 秒
那么可以认为每秒 10,000,000/86400=116 人
峰值并发用户数是：4，000，000/3600X3=371 人

还有一种情况大概是不知道并发用户，这时候就要考虑一个序列下系统的表现情况，比如 50，100，150 等，以此加压看系统是否满足业务部门提出的响应时间要求，从而定位最佳使用用户数目。还有一种方法，稍后介绍。

最大用户数、同时在线用户计算公式

针对不同行业和应用，不尽相同，大概是同时在线用户=最大用户数 x 20%
比如一个系统要求支持的最大用户是 10 万，那么根据经验同时最大在线数就是 2 万

- 测试用例的设计

性能测试方案的设计比功能测试更严格，需要详细进行设计测试设计、包括数据库，带宽
需要详细准备测试环境

性能测试环境的准备不到位，往往导致结果大相径庭。测试出来的数据不准确，没有代表性，或者错误的数据。

性能测试并发用户 test case 范例

并发用户数	事务平均响应时间	事务最大响应时间	事务成功率	平均流量
100				
150				
200				

- Virtual User Generator

浏览器设置
常见问题是系统有多个浏览器，导致找不到默认浏览器，两种方法进行修改：更改系统设置或者指定浏览器目录。

代理设置
请在录制之前确认你的应用是否需要代理，并在录制选项里面进行必要的设置。

HTML based vs URL based
两种不同的代码格式，适应不同的应用。
有以下原则：

1. 基于浏览器的应用推荐使用 HTML based

2. 不是基于浏览器的应用推荐使用 URL based
3. 如果基于浏览器的应用，包含 javascript，并且该脚本向服务器发生请求，也使用 URL based
4. 基于浏览器的应用使用 HTTPS 安全协议，使用 URL based

web_url, web_link, web_image,web_submit_form 等
web_url, web_submit_data

对于 headers 按钮，如果录制的时候出现乱码，解决不了的情况，可以尝试 accept-language，这个对于 webspere 的情况特别有用。

日志设置
调试的时候进行的设置，单用户模式
运行负载的时候设置、压力环境的设置，设置不当可能导致磁盘空间不足

迭代设置
不同的迭代可以通过 block 进行组合设置，在 controller 中迭代设置进一步讲解。

思考时间
单用户模式下跟场景模式下的设置需要考虑对环境的压力，分析结果的时候可以刨去的时间

Cache 设置、模拟新用户
每个用户启动之前清空 cache，保持每次连接的时候都是采用的新连接，对系统的压力更大。

带宽
模拟不同的请求速度，控制对系统的压力。

自己定义或者激活关联规则
关联规则可以自己定义。

Token，sessionid 等需要进行关联
手动关联的步骤

web_reg_save_param()

web_find and web_reg_find 的区别

1. 前者只针对基于 HTML 录制的代码，后者可以针对 HTML 和 URL based
2. 前者可以针对 HTML 页面内容搜索，后者可以针对 source code 进行搜索
3. 前者是在打开页面或提交请求之后，后者是注册类，在之前

参数化规则定义和使用

参数化策略
每次迭代：顺序、随机，唯一等
何时更改：下一迭代、下次出现、once

- 录制代码

插入事务
不用 action 来作为事务的名称，自己插入事务，便于衡量。

插入集合点
不要为了集合点而插入集合点，要有目的的。

增加调试代码
增加一些必要的日志和 log 等，方便调试。

增加关联
切记，不要为了关联而关联，不要一出现错误就考虑用关联，看清楚错误的本质。
手动关联的时候采取的步骤。

参数化
不要为了参数化而参数化，考虑实际的应用限制，区别对待进行参数化。

- Controller 介绍

Group 的设置策略，按照组执行，而不是用户
每组进行执行的顺序，前后的间隔等，都可以通过组的策略进行设置。

通过组的设置，可以模拟多重用户角色，比如同样的脚本，思考时间的多少，网络带宽等等。这里面的 RTS 可以是共享的，也可以是各自独立的。

面向目标的场景可以有下面一些：

- 虚拟用户数
- 每秒点击次数（web user only）
- 每秒事务数
- 每分钟页面数（web user only）
- 事务响应时间

Load generator 的添加和设置
负载生成器不是任何时候都必须的，添加的时候需要先进行权限设置，比如 Windows 服务器作为负载生成器的时候，需要管理员的权限先进行设置。

首先保证被监视的 windows 系统开启以下二个服务 Remote Procedure Call(RPC) 和 Remote Registry Service (这里具体在那里开起服务就不说了)
被监视的 WINDOWS 机器:右击我的电脑,选择管理->共享文件夹->共享 在这里面要有 C$这个共享文件夹,(要是没有自己手动加)
然后保证在安装 LR 的机器上使用运行.输入\\被监视机器 IP\C$ 然后输入管理员帐号和密码,如果能看到被监视机器的 C 盘了,就说明你得到了那台机器的管理员权限,可以使用 LR 去连接了

说明: LR 要连接 WINDOWS 机器进行监视要有管理员帐号和密码才行,

其他问题相关可以参考：
http://www.rickyzhu.com/2007/03/15/monitor-windows-and-linux-using-loadrunner/

系统资源添加
添加的时候也需要进行确认可以连通。

集合点策略
多少用户达到集合点的时候释放、all vuser, running vuser 的区别。

目标场景
无法确定目标用户的时候，可以使用目标场景进行逐步递加，由系统自动增加用户。

- IP Snooper Ditto 介绍

- 控制 Controller 执行

加压、降压设置在测试用例中设计
首先对系统进行预热，一次不要加太多用户。

自然终止（迭代此时生效）
指定持续时间
永久执行，等待人工停止

监控执行

退出状态为什么有时候是 pass，有时候是 stop

自然终止-pass
设置运行时间-stop