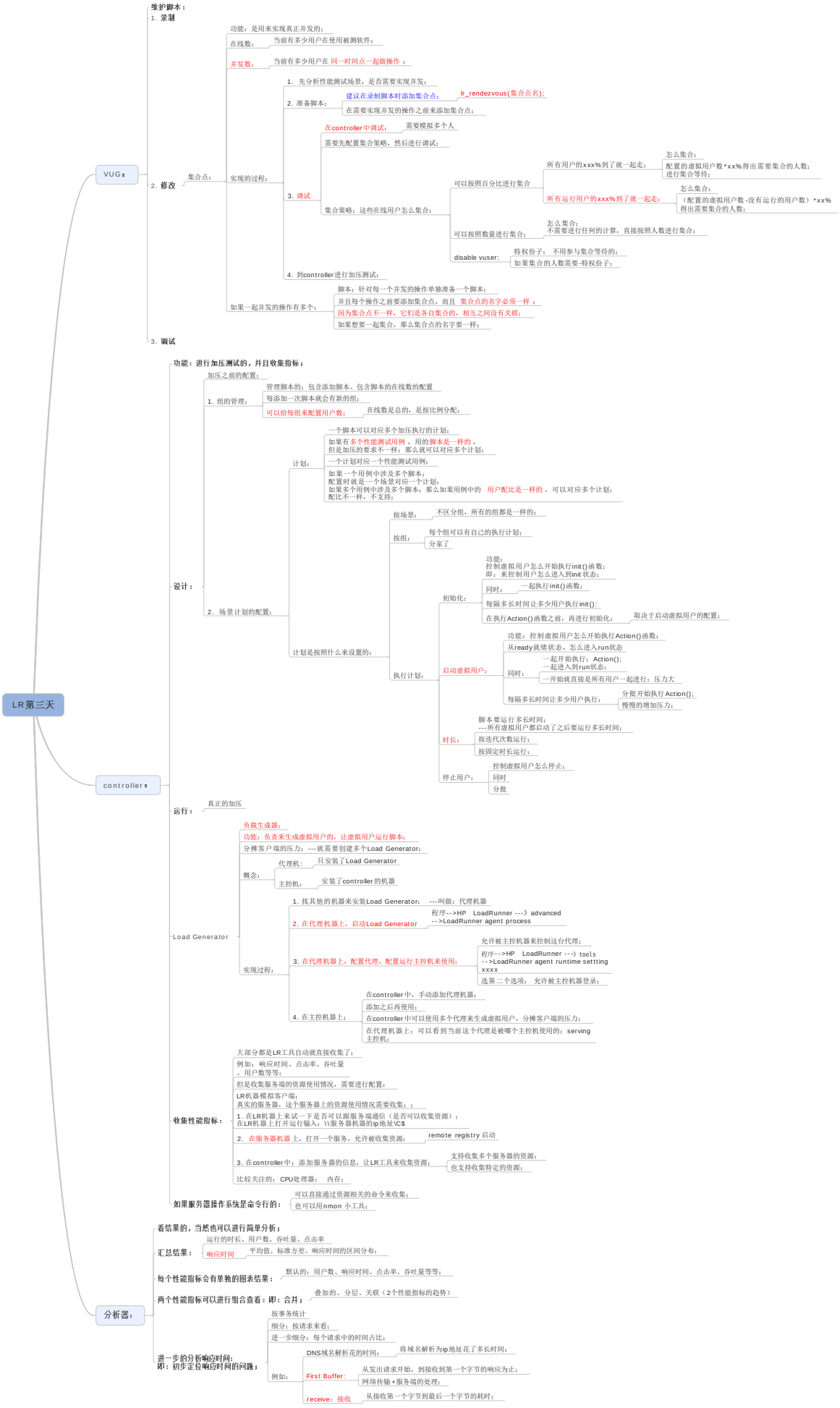
LR第三天
VUG：
维护脚本：
1. 录制
2. 修改
集合点：
功能：是用来实现真正并发的：
在线数：
当前有多少用户在使用被测软件：
并发数：
当前有多少用户在 同一时间点一起做操作 ：
实现的过程：
1. 先分析性能测试场景，是否需要实现并发：
2. 准备脚本：
建议在录制脚本时添加集合点：
lr_rendezvous(集合点名)；
在需要实现并发的操作之前来添加集合点：
3. 调试
在controller中调试：
需要模拟多个人
需要先配置集合策略，然后进行调试：
集合策略：这些在线用户怎么集合：
可以按照百分比进行集合
所有用户的xxx%到了就一起走：
怎么集合：
配置的虚拟用户数*xx%得出需要集合的人数；
进行集合等待：
所有运行用户的xxx%到了就一起走：
怎么集合：
（配置的虚拟用户数-没有运行的用户数）*xx%
得出需要集合的人数；
可以按照数量进行集合：
怎么集合：
不需要进行任何的计算，直接按照人数进行集合：
disable vuser:
特权份子： 不用参与集合等待的：
如果集合的人数需要-特权份子：
4. 到controller进行加压测试：
如果一起并发的操作有多个：
脚本：针对每一个并发的操作单独准备一个脚本；
并且每个操作之前要添加集合点，而且 集合点的名字必须一样 ；
因为集合点不一样，它们是各自集合的，相互之间没有关联；
如果想要一起集合，那么集合点的名字要一样；
3. 调试
controller：
功能：进行加压测试的，并且收集指标；
设计：
加压之前的配置：
1. 组的管理：
管理脚本的：包含添加脚本、包含脚本的在线数的配置
每添加一次脚本就会有新的组；
可以给每组来配置用户数：
在线数是总的，是按比例分配：
2. 场景计划的配置：
计划：
一个脚本可以对应多个加压执行的计划：
如果有多个性能测试用例 ，用的脚本是一样的，
但是加压的要求不一样；那么就可以对应多个计划；
一个计划对应一个性能测试用例；
如果一个用例中涉及多个脚本；
配置时就是一个场景对应一个计划；
如果多个用例中涉及多个脚本；那么如果用例中的 用户配比是一样的 ，可以对应多个计划；
配比不一样，不支持；
计划是按照什么来设置的：
按场景：
不区分组，所有的组都是一样的；
按组：
每个组可以有自己的执行计划；
分家了
执行计划：
初始化：
功能；
控制虚拟用户怎么开始执行init()函数；
即：来控制用户怎么进入到init 状态；
同时：
一起执行init()函数；
每隔多长时间让多少用户执行init()；
在执行Action()函数之前，再进行初始化：
取决于启动虚拟用户的配置；
启动虚拟用户：
功能：控制虚拟用户怎么开始执行Action()函数；
从ready就绪状态，怎么进入run状态
同时：
一起开始执行：Action()；
一起进入到run状态；
一开始就直接是所有用户一起进行：压力大
每隔多长时间让多少用户执行：
分批开始执行Action()；
慢慢的增加压力；
时长：
脚本要运行多长时间；
---所有虚拟用户都启动了之后要运行多长时间；
按迭代次数运行：
按固定时长运行：
停止用户：
控制虚拟用户怎么停止：
同时
分批
运行：
真正的加压
Load Generator
概念：
负载生成器：
功能：负责来生成虚拟用户的，让虚拟用户运行脚本；
分摊客户端的压力：---就需要创建多个Load Generator；
代理机：
只安装了Load Generator
主控机：
安装了controller的机器
实现过程：
1. 找其他的机器来安装Load Generator：---叫做：代理机器
2. 在代理机器上，启动Load Generator
程序-->HP LoadRunner --->advanced
-->LoadRunner agent process
3. 在代理机器上，配置代理，配置运行主控机来使用：
允许被主控机器来控制这台代理：
程序-->HP LoadRunner --->tools
-->LoadRunner agent runtime settting
xxxx
选第二个选项： 允许被主控机器登录：
4. 在主控机器上：
在controller中，手动添加代理机器：
添加之后再使用；
在controller中可以使用多个代理来生成虚拟用户，分摊客户端的压力；
在代理机器上：可以看到当前这个代理是被哪个主控机使用的：serving
主控机：
收集性能指标：
大部分都是LR工具自动就直接收集了：
例如：响应时间、点击率、吞吐量
、用户数等等；
但是收集服务端的资源使用情况，需要进行配置：
LR机器模拟客户端；
真实的服务器，这个服务器上的资源使用情况需要收集；；
1. 在LR机器上来试一下是否可以跟服务端通信（是否可以收集资源）；
在LR机器上打开运行输入：\\服务器机器的ip地址\C$
2. 在服务器机器上，打开一个服务，允许被收集资源：
remote registry 启动
3. 在controller中：添加服务器的信息，让LR工具来收集资源：
支持收集多个服务器的资源；
也支持收集特定的资源；
比较关注的：CPU处理器； 内存；
如果服务器操作系统是命令行的：
可以直接通过资源相关的命令来收集；
也可以用nmon 小工具；
分析器：
看结果的，当然也可以进行简单分析；
汇总结果：
运行的时长、用户数、吞吐量、点击率
响应时间
平均值、标准方差、响应时间的区间分布；
每个性能指标会有单独的图表结果：
默认的：用户数、响应时间、点击率、吞吐量等等；
两个性能指标可以进行组合查看：即：合并；
叠加的、分层、关联（2个性能指标的趋势）
进一步的分析响应时间：
即：初步定位响应时间的问题；
按事务统计
细分：按请求来看：
进一步细分：每个请求中的时间占比：
例如：
DNS域名解析花的时间：
将域名解析为ip地址花了多长时间；
First Buffer:
从发出请求开始，到接收到第一个字节的响应为止；
网络传输+服务端的处理；
receive：接收
从接收第一个字节到最后一个字节的耗时；