

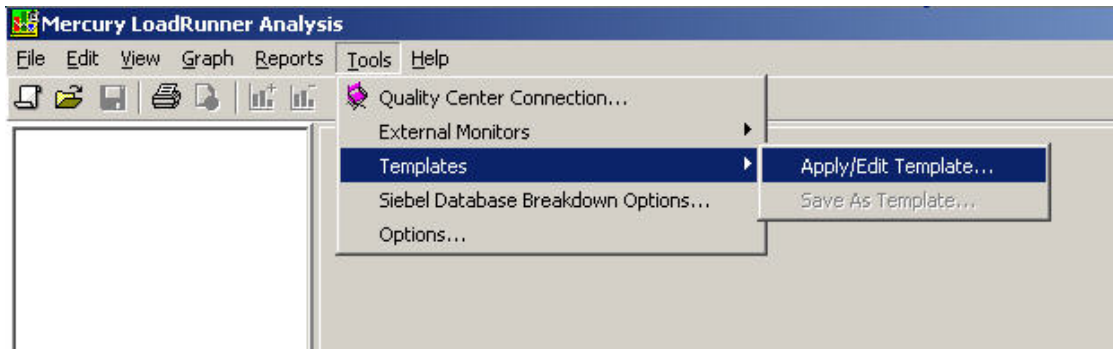
性能测试经验总结

第一步：计划测试

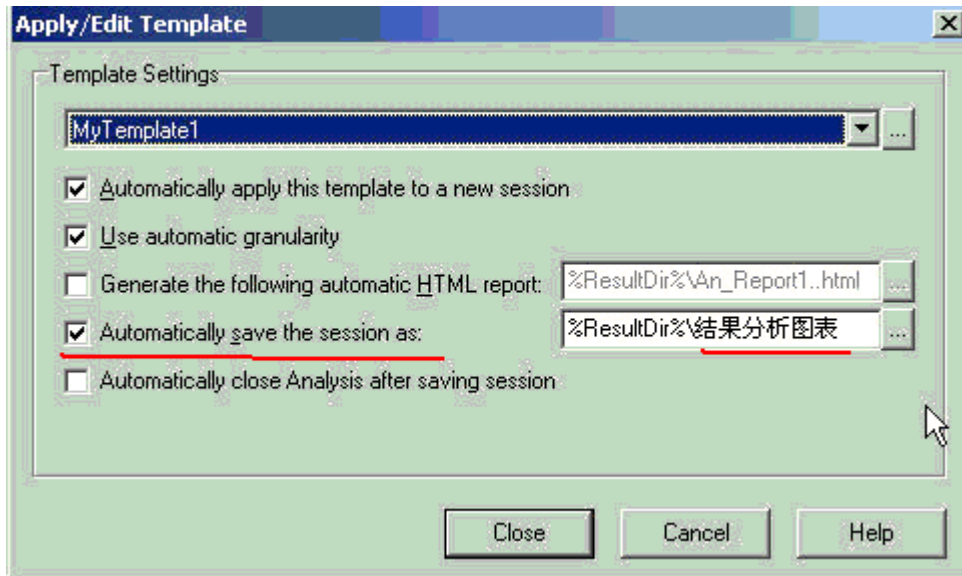
- 1、明确压力点，根据压力点设计多少种场景组合
- 2、把文档（包括多少种场景组合、场景与场景组合条件的对应表）写好
- 3、如果监测 UNIX 机器，在被监测的机器需要安装监测 Unix 的进程
- 4、让开发人员帮助我们准备测试数据或他们写相关的文档我们来准备数据
- 5、让开发人员做一个恢复数据的脚本，以便于我们每次测试的时候都能够有一个相同的环境
- 6、针对每一个模块包括四个子文件夹：如模块 A 下包括“脚本”“场景”“结果”“图表”四个子文件夹，每个子文件夹储存对应的文件，如下表所示

模块名	场景名	结果名	图表名
用户层障碍	1 场景	1 场景 0	1 场景 0
		1 场景 1	1 场景 1
		1 场景 2	1 场景 2
	2 场景	2 场景 0	2 场景 0
		2 场景 1	2 场景 1
		2 场景 2	2 场景 2


其中：结果名“1 场景”是在场景中的“Results Setting”中设置的，具体的设置见“建立场景”部分，这里也可以有另外一种方法：在打开模板设置，如下：



选中“Automatically save the session as:”并且在“%ResultDir%”后面填写你想保存的文件名，当你打开某个 lrr 文件时，系统自动在当前目录中生成一个文件保存分析图表，如下图所示：




第二步：生成测试脚本

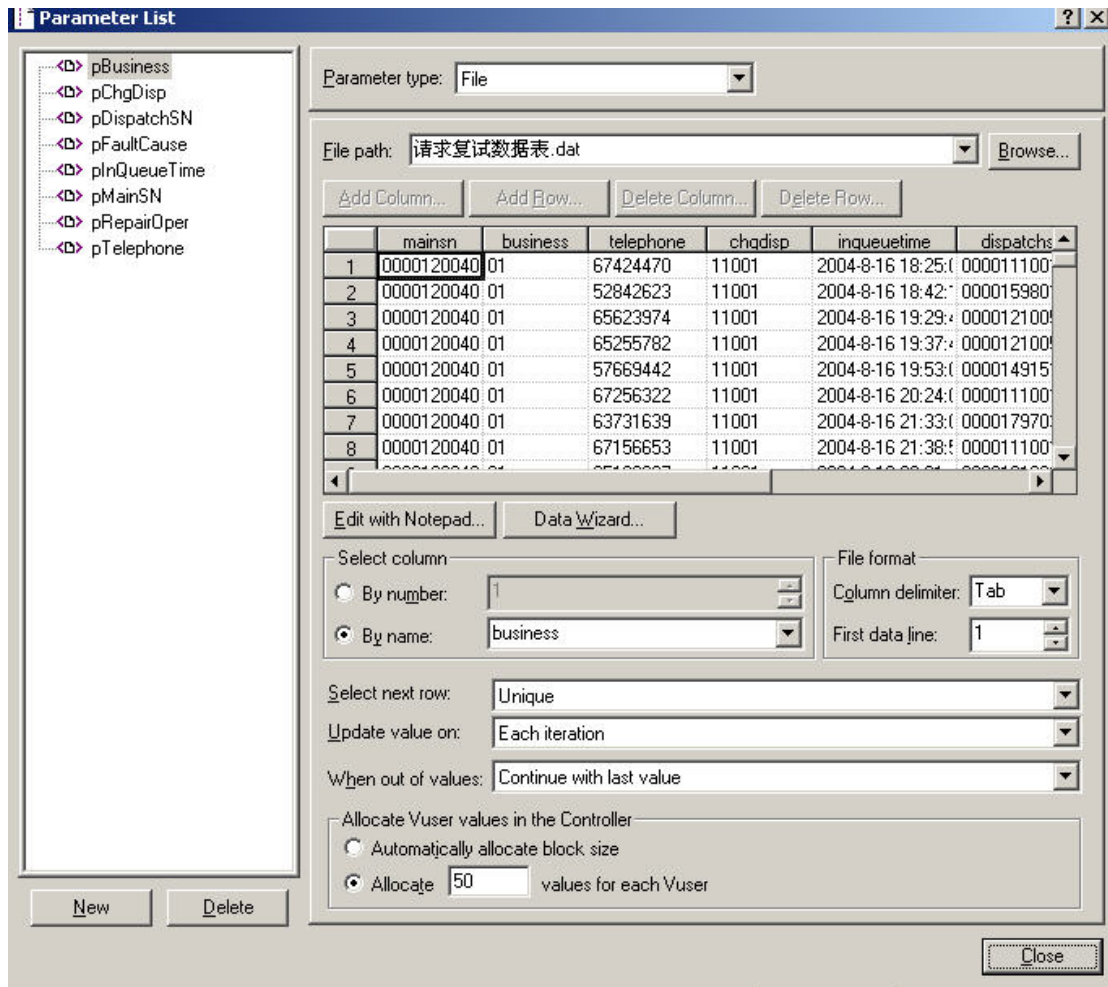
- 1、把登陆部分放到“vuser_init”部分,把需要测试的内容部分放到“Action”部分执行；但是如果是模拟多个用户登陆系统，则要把登陆部分放到 Action 部分来实现
- 2、录制脚本后，想查询某个函数的原型，按“F1”键
- 3、确认脚本中哪些参数是需要进行参数化的(最好能可以和开发人员一起确认)
- 4、在脚本参数化时把函数 web_submit_data()中的 ITEMDATA 后面的数据参数化，因为这些数据是传递给服务器的，当然也可以把一个函数中的所有相同变量都替换掉
- 5、脚本中无用的部分用“/*”“*/”“//”注释掉，但最好不要删除
- 6、调试脚本遵循以下原则：
 - 确认在 VU 里 SUSI（单用户单循环次数 single user & single iteration）
 - 确认在 VU 里 SUMI（单用户多循环次数 single user & multi iteration）
 - 确认在 controller 中 MUSI(多用户单循环次数 multi user & single iteration)
 - 确认在 controller 中 MUMI(多用户多循环次数 multi user & multi iteration)
- 7、事务的名称取的有意义便于事务之间的区分，把所有的事务名都记录在一起，便于在测试结果概要中区分它们，这要写成一个表：某次测试有哪些模块，每个模块中有哪些事务(见对应的“关系表”)
- 8、在  Param List “Parameter List” 中可以选择参数类型“Random Number”，使某一个参数取设定的范围内的随机值

第三步：建立场景

- 1、把场景名称编号，并制定出一份场景名称和场景条件组合的对应表。比如，场景 m 对应于“某一模块_xx 个 vu_分 z 台 machine”（见“关系表”中的例子）
- 2、根据上面的对应表把场景设置好，需要设置的要素如下：总体多少个用户、分多少个组、

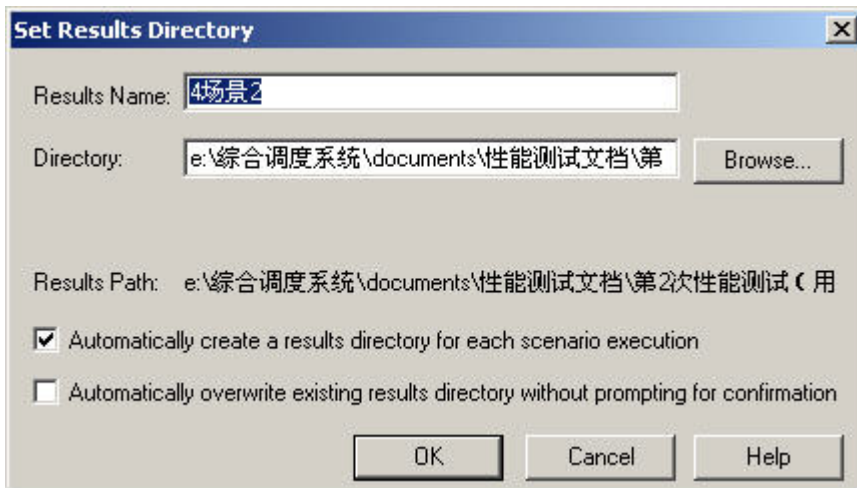
每个组有多少个用户、分几台机器运行、每个脚本迭代多少次、是否回放 think time 时间、检查 Parameter List 中每个参数设置是否正确、参数从表中取值间隔是否正确、是否选中 “Initialize all Vusers before Run”

- 3、测试结果应该保存为 “m 场景 0, m 场景 1,…”
- 4、把虚拟用户分散到几台机器上和在一台机器上面都要进行测试，因为有可能效果不同
- 5、场景中如果有需要改动的地方，必须新建一个场景(建议使用 “另存为”，然后再修改结果文件名，再选择相应的脚本)，并把场景按顺序编号，先维护好场景与场景组合条件的对应表，以便以后的查找，并且在结果  “Results Setting” 中设置的结果名与场景名相同。建议在 “Results Setting” 中选中 “Automatically create a results directory for each scenario execution” 让它每次自动累加，不建议选中 “Automatically overwrite existing results directory without prompting for confirmation”，因为我们不要覆盖掉以前的测试结果，把它保存下来以便有个根据。
- 6、需要注意的地方：当在 “Parameter List” 中的 “Select next row” 选中 “Unique” 时，如果再在 “Edit Schedule\Schedule by Scenario\Duration” 中选中第二项 “Run for XX after the ramp up has been completed” 时系统就会报错，提示 “Unique” 类型不相符。
- 7、在 “Run-time Setting” 设置中，“General” 中的 “Pacing” 非常有用，可以设置每次迭代之间相隔多少时间，也可以是随机的取值
- 8、建议：把 “Parameter List” 和 “Run-time Setting” 中的所有设置都搞熟悉，这样便于以后对脚本和场景进行设置
- 9、设计 “Parameter List” 时的小技巧：即在 “Allocate X values for each Vuser” 时，尽量



把它的间隔在数据容许的范围内取大些，这样可以做从一次迭代到最大值迭代，而且对脚本没有什么影响

- 10、当一个脚本中有多个事务，在事务前面增加集合点时需要一点技巧。或者我们把脚本复制几个，或者我这样做：测试前面的事务的压力时，把后面的事务前的集合点设置为不激活状态；在测试后面的事务的压力时，把前面的事务的集合点设置为不激活状态，另外最好不选中 Initialize all Vusers before Run，具体参见 Controller 中的“Scenario/Rendezvous”，及用户手册(按 F1)
- 11、把持续时间从最后 60 秒改为整个场景的时间，右键单击某个图，选择“Configure”，修改 Graph Time 即可
- 12、每次从一个场景修改后保存为另一个场景时别忘记把结果保存文件名修改相对应的文件名。在设置结果保存文件名时有一个技巧：如果你打开这个窗口时，点击确定则系统会



默认以“4 场景 2”为基点向后加“4 场景 20”“4 场景 21”等等，但是如果你把结果文件名后面的数据去掉，改为“4 场景”，点击确定后，系统会自动搜索是以“4 场景”开头的文件名，并在它的后面继续增加，比如把它改为“4 场景”时，下次结果保存在“4 场景 3”中。而且他在搜索的时候搜索以“4 场景”开头的文件名，从 0 开始，有的话就不取代而跳过，没有的话就取代。

第四步：运行场景

- 1、运行场景前需要注意的事项：每个组的虚拟用户数、迭代次数、think time、参数化时的取值间隔、执行恢复数据的脚本、确认虚拟机的 LoadRunner Agent Service 打开
- 2、如果监测 Unix，运行场景前需要启动监测 Unix 进程，启动的命令“rpc.rstatd”、查看这个进程是否启动的命令“rpcinfo -p”
- 3、运行前使 Generator 机器处理 Ready 状态
- 4、确认被监测的机器已经连接上去，并且添加自己所需要的计数器
- 5、运行之前一定要确认系统中压力点的数据量是多少
- 6、确认以上都正确时再运行测试场景


第五步：监视场景

- 1、打开

Passed Transactions	0	🔍
Failed Transactions	0	🔍

 “Passed Transactions”或“Failed Transactions”，可以随时观察到事务的运行状态

第六步：分析测试结果

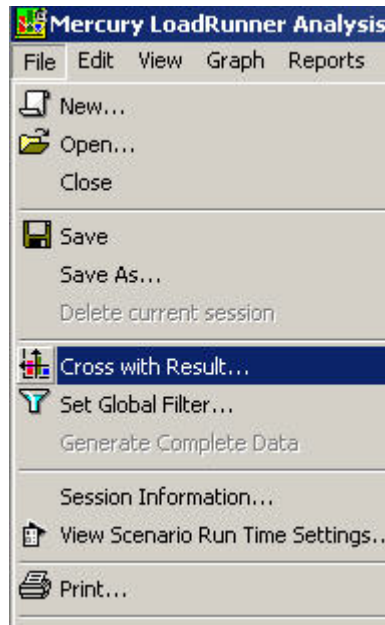
- 1、打开 Analysis 后，把经过数据处理的结果图表保存到“图表”文件夹，并且文件名和场景名、结果名相同，这样便于以后的查阅。也可以省去每次进行数据处理的时间。
- 2、可以通过点击界面上的  “View Run Time Setting”可以看到此场景运行时的一些场景

设置

- 3、在关联图表时可以自动调节每个元素的比例，点击右键，选择

View Measurement Trends 即可

- 4、每次测试结束后确认所做的操作是正确的，确认正确后再分析结果
- 5、在结果文件夹中为每个场景建立一个文档，把每次运行时的情況记录下来以便于写测试报告，尤其运行错误的原因记录下来，并把开发人员所做的修改也记录下来以便知道开发人员做了些什么修改
- 6、在分析运行结果时可以把几个结果合在一起进行比较，打开如下“Cross with Result...”



即可，然后增加一个运行结果，这样就可以把你所需要的结果放到一起比较了