

# 性能测试

## --性能测试基础知识

# 知识回顾

---

## ■ 什么是性能

## ■ 什么是性能测试

- 是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

## ■ 为什么进行性能测试

# 知识回顾

## ■ 性能测试术语

### — 吞吐量

- 单位时间内能够处理的事务数目

### — 资源使用率

### — 点击率

■ 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

# 知识回顾

## ■ 性能测试分类

### — 负载测试

- 在一定的软件、硬件及网络环境下，运行一种或多种业务，在**不同虚拟用户数量**的情况下，测试服务器**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**

# 知识回顾

## ■ 压力测试

- 指在一定的软件、硬件及网络环境下，模拟大量的**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**

# 知识回顾

## ■ 容量测试

- 在一定的软件、硬件及网络环境下，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（包含对未来几年的处理能力及扩展能力）

# 知识回顾

## ■ 配置测试

- 指在不同的软件、硬件及网络环境配置下，运行一种或多种业务，在一定的虚拟用户数量情况下，**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。通过产生不同的配置来得到系统性能的变化情况

# 知识回顾

---

## ■ 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



# 知识回顾

## 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

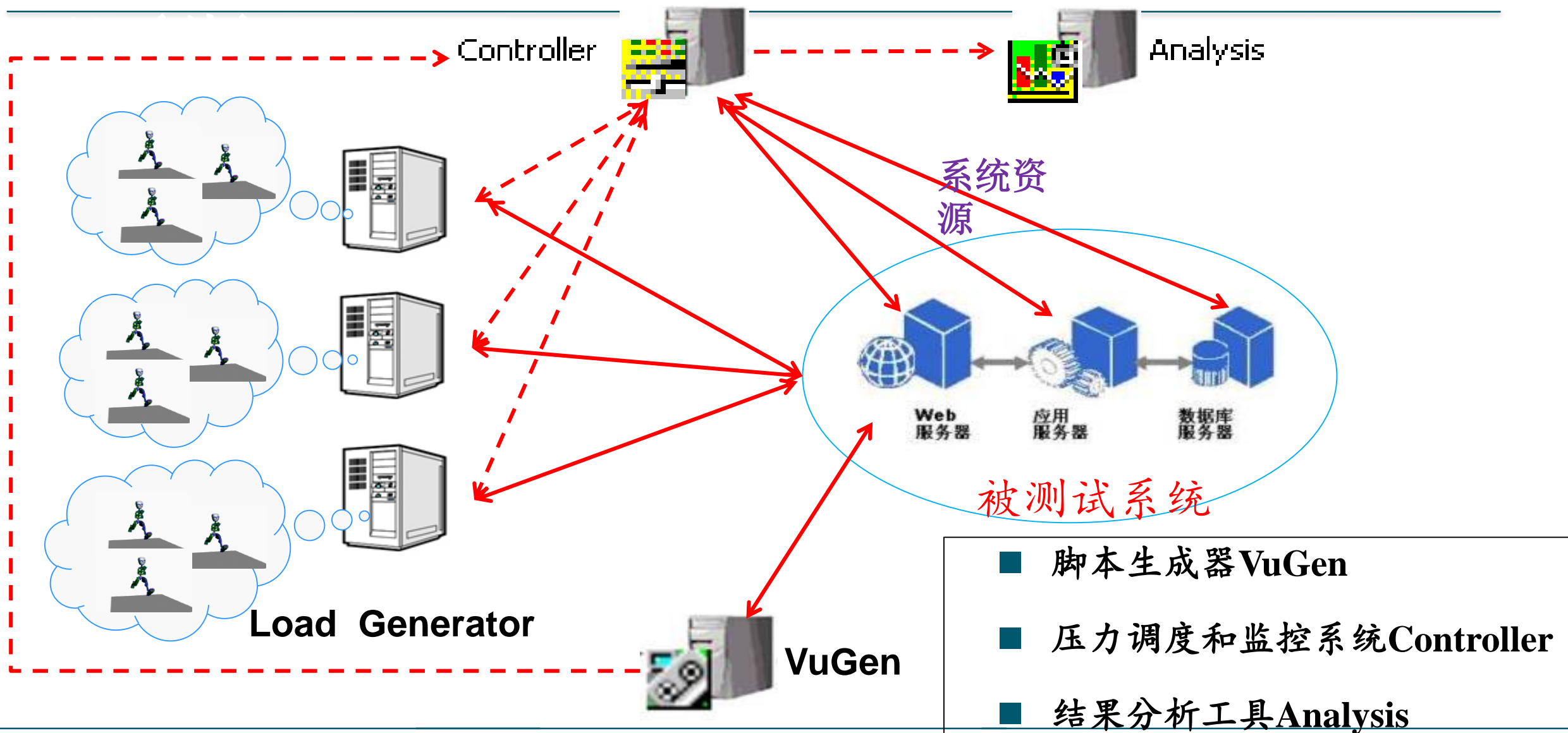
性能调优

# 知识点回顾

---

- HTTP协议
- 抓包工具使用
- LoadRunner安装和初体验

# LoadRunner 工具组成



# 目录

---

- LoadRunner开启
- 协议选择
- 录制选项设置
- 录制登录操作、手动书写函数
- Run-time Settings

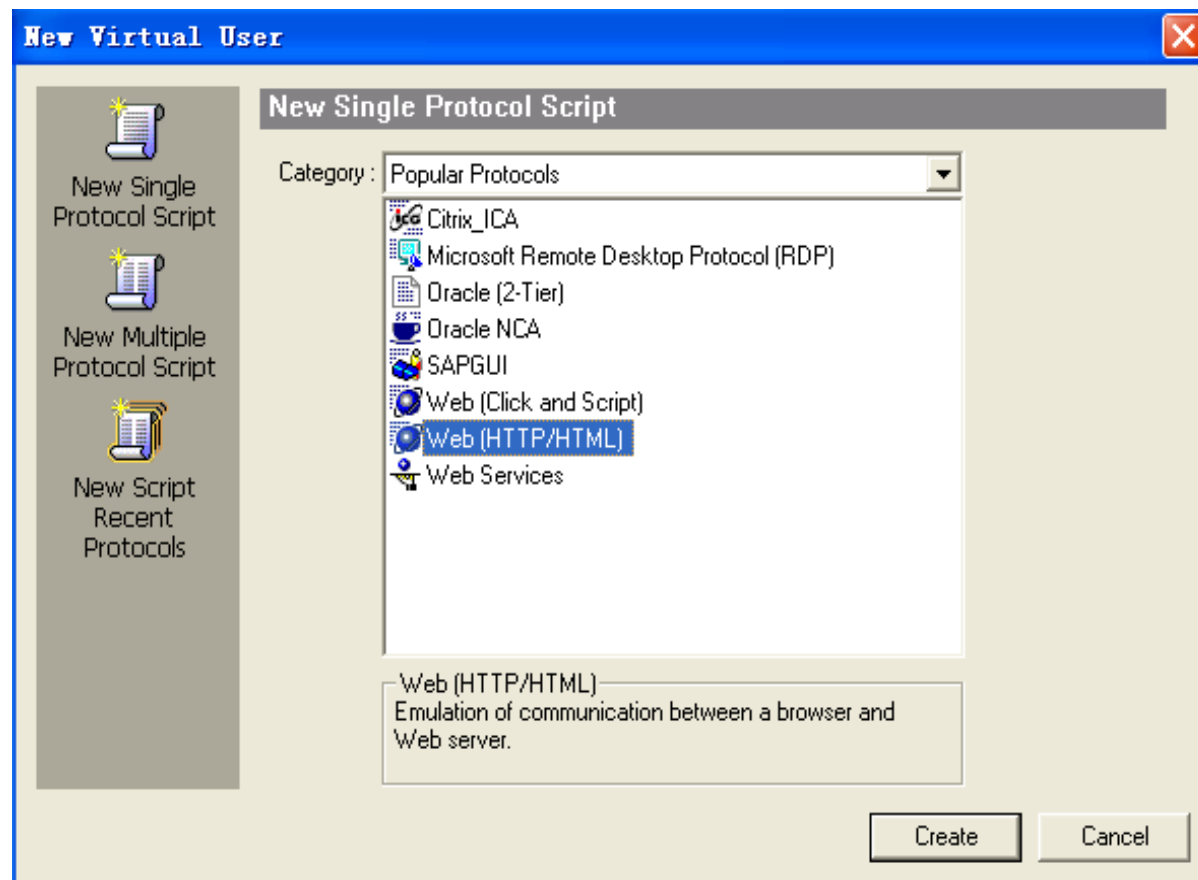
# LoadRunner 开启



# LoadRunner 开启



# 如何选择协议



# 目录

---

- LoadRunner开启
- 协议选择
- 录制选项设置
- 录制登录操作、手动书写函数
- Run-time Settings



# 如何选择协议



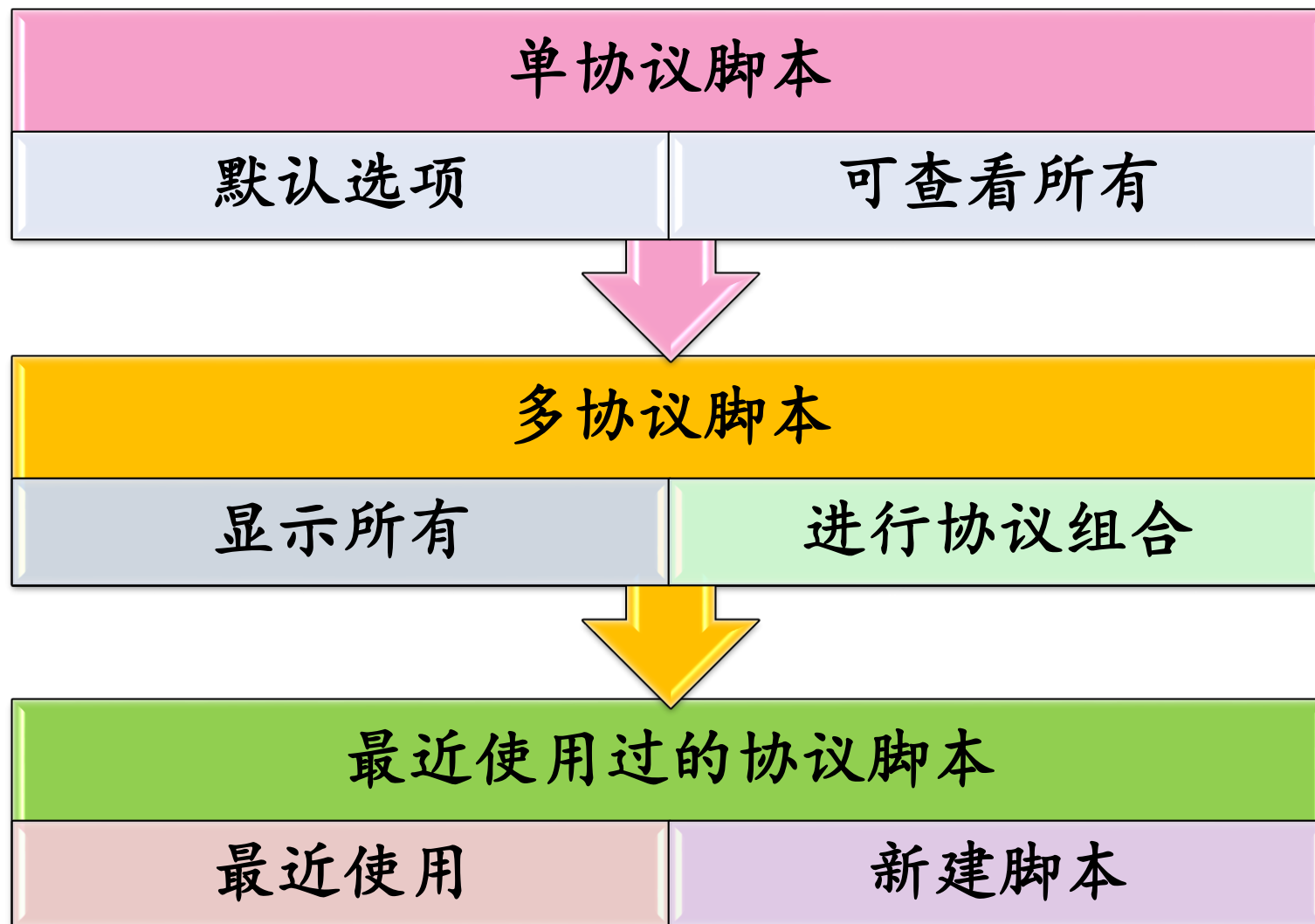
## 如何选择协议—协议分类

协议类型	可支持的协议
应用程序部署解决方案	Citrix ICA、RDP
客户端/服务器	DB2 CLI、DNS、Informix、Microsoft .NET、MS SQL Server、ODBC、Oracle (2层)、Sybase Ctlib、Sybase Dbllib和Windows Sockets
自定义	C Vuser、Java Vuser、JavaScript Vuser、VB Script Vuser、VB Vuser、VBNet Vuser
分布式组件	COM/DCOM、Microsoft .NET
电子商务	AMF、AJAX (Click and Script)、FTP、Flex、LDAP、Microsoft .NET、Web (Click and Script)、Web (HTTP/HTML) Web Services

## 如何选择协议——协议分类

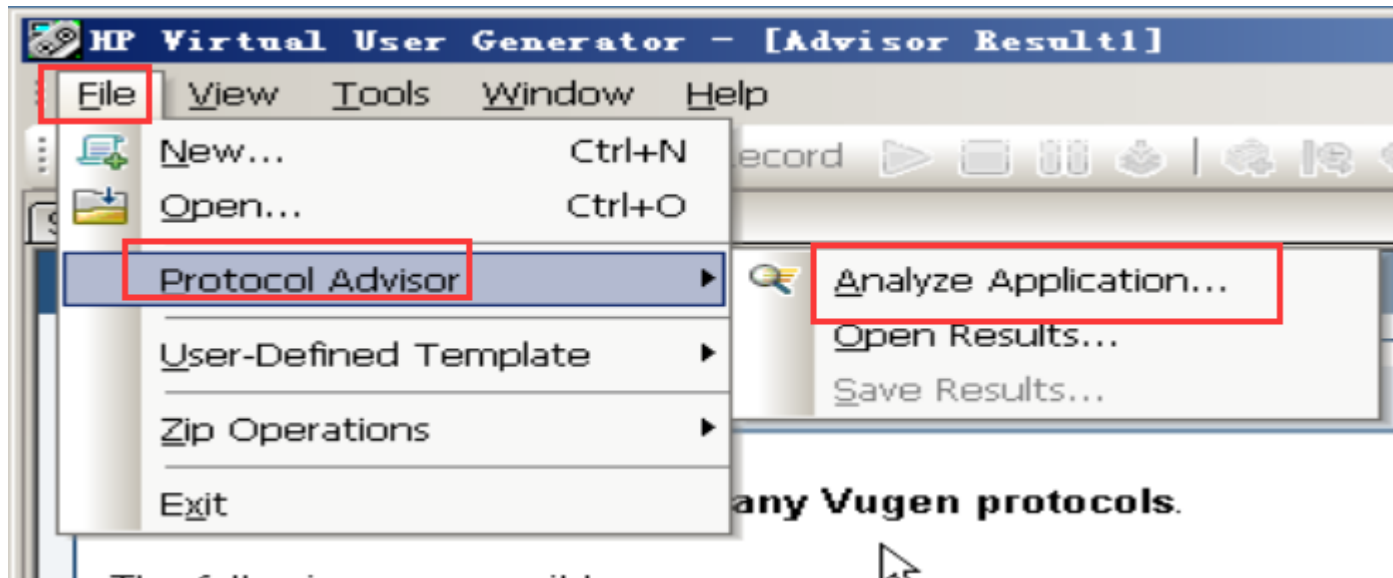
ERP/CRM	Oracle NCA、Oracle Web Applications 11i、Peoplesoft-Enterprise、Peoplesoft-Tuxedo、SAP-Web、SAP (Click and Script)、SAPGUI、Siebel-Web
Enterprise Java Bean	EJB
Java	JAVA Record Replay
传统	终端仿真 (RTE)
邮件服务	Internet邮件访问协议(IMAP)、MS Exchange (MAPI)、POP3和SMTP
中间件	Tuxedo和Tuxedo6
流数据	Media Player (MMS)和Real协议
无线	i-Mode、Multimedia Messaging Service (MMS) 和WAP

# 如何选择协议—创建协议方式



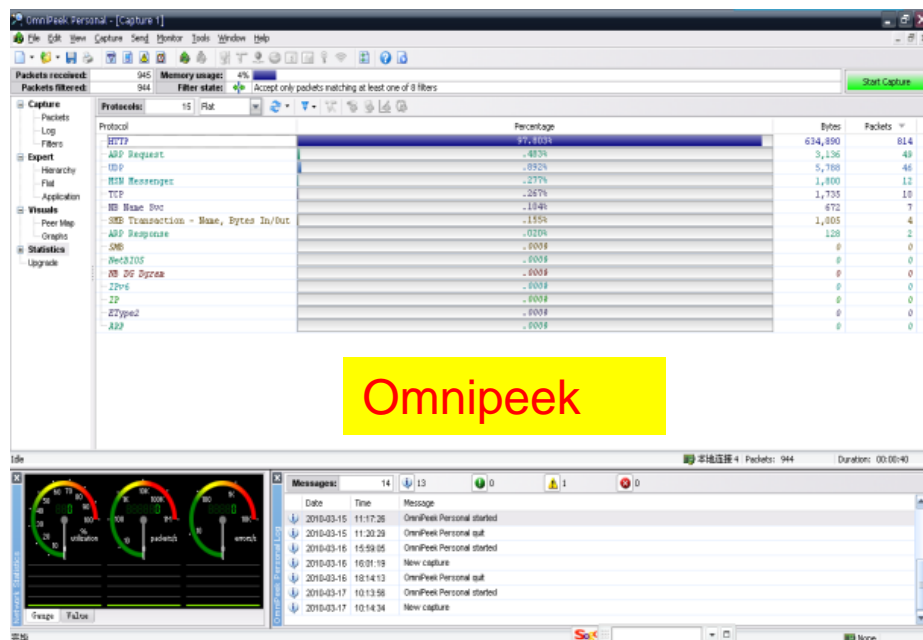
## 如何选择协议—确定方法

- 通过询问开发人员获知所使用的协议；
- 通过LR自带的“Protocol Advisor”——建议采用；
- 通过概要或详细设计手册获知所使用的协议；



# 如何选择协议—确定方法

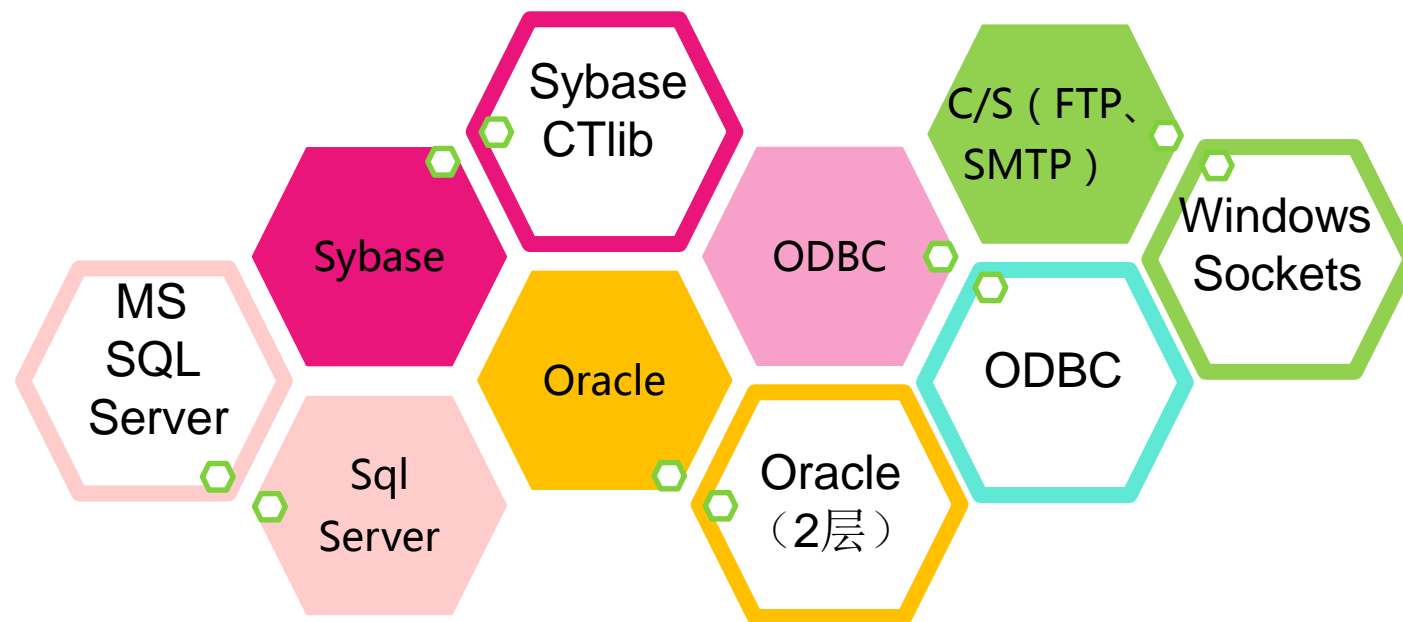
- 通过协议分析工具（如：Omnipeek）捕包分析；
- 通过以往经验确定被测对象所使用的协议。



# 如何选择协议（续）——原则

## ■ 协议选择原则

- B/S结构，选择Web（HTTP/HTML）协议；
- C/S结构，根据后端数据库的类型来选择；



# 目录

---

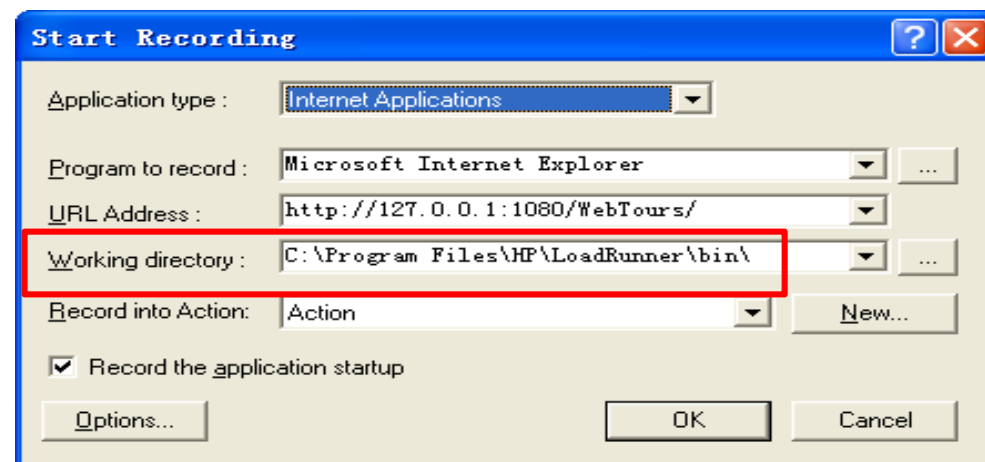
- LoadRunner开启
- 协议选择
- 录制选项设置
- 录制登录操作、手动书写函数
- Run-time Settings



# 录制前设置录制界面

## ■ 字段含义

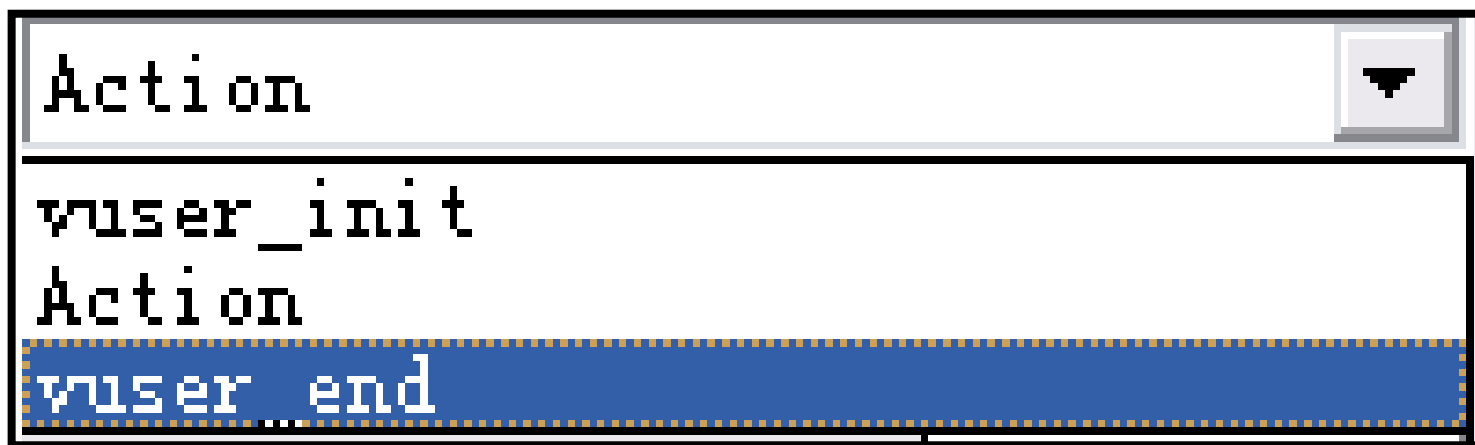
- **【Application type】**：应用程序类型
- **【Program to record】**：要录制的程序
- **【URL Address】**：待测试的URL
- **【Working directory】**：设置工作目录
- **【Record into Action】**：录制到操作，即选择把录制的脚本存放于哪一个函数部分



# Record into Action

## ■ 分为三部分：

- `vuser_init` : 初始化
- `Action` : 具体操作, 可重复
- `vuser_end` : 相当于析构函数



```
Action
vuser_init
Action
vuser_end
```

# 录制订票页面的登录操作

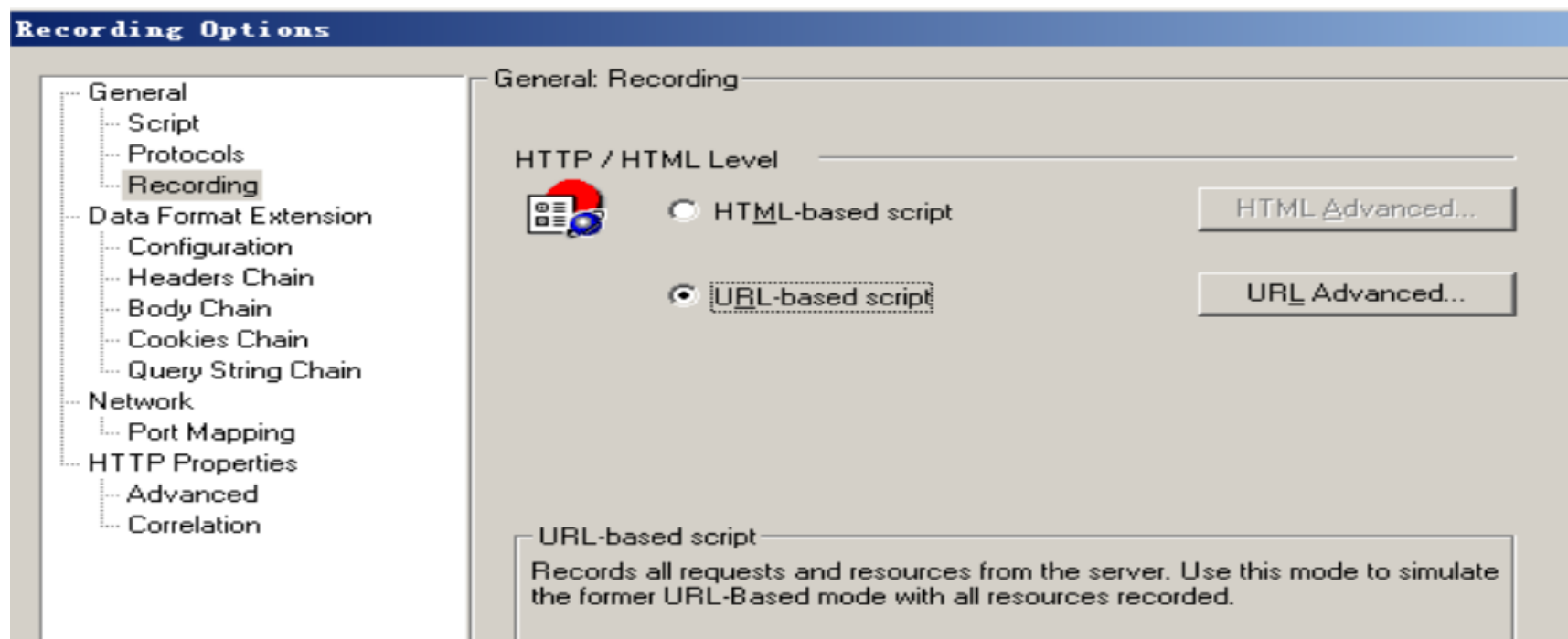
---

- 1 新建脚本
- 2 设置相关选项，开始录制
- 3 录制后，保存脚本

# 录制订票页面的登录操作

## 4 修改设置项，重新录制

—将Recording options 中 HTML-based script 改为URL-based script



# 录制选项设置

## ■ HTML-based script

- 基于 HTML 的脚本级别：为每个 HTML 用户操作生成单独的步骤和函数。步骤直观且脚本容易理解和维护。

## ■ URL-based script

- 基于 URL 的脚本级别：录制“客户端向服务器发送请求后，服务器返回给客户端的所有浏览器请求和资源”。它自动将每个 HTTP 资源（即所有操作）录制为 URL 步骤（即由 web\_url 语句构成的脚本）。

较上一方式，记录了更详细的客户端操作信息，甚至可捕获非 HTML 形式应用程序，如小程序、非浏览器程序。但生成的脚本内容长且多，显示不直观

# 目录

---

- LoadRunner开启
- 协议选择
- 录制选项设置
- 录制登录操作、手动书写函数
- Run-time Settings

# 录制脚本分析

---

- `web_URL()`
- `web_submit_data()`
- `web_custom_request()`
- 手动书写如上三个函数，实现售票系统登录操作

# 脚本分析

```
web_url("test", "URL=http://127.0.0.1:1080/WebTours", LAST);
```

```
web_submit_data("web_submit_data",  
  "Action=http://127.0.0.1:1080/WebTours/",  
  "Method=POST",  
  "TargetFrame=",  
  "Referer=",  
  "Mode=HTML",  
  ITEMDATA,  
  "Name=username", "Value=jojo", ENDITEM,  
  "Name=password", "Value=bean", ENDITEM,  
  LAST);
```

```
web_custom_request("web_custom_request",  
  "URL=http://127.0.0.1:1080/WebTours/",  
  "Method=POST",  
  "TargetFrame=",  
  "Resource=0",  
  "Referer=",  
  "Mode=HTML",  
  "EncType=utf-8",  
  "Body=username=jojo&&password=bean",  
  LAST);
```



# 目录

---

- LoadRunner开启
- 协议选择
- 录制选项设置
- 录制登录操作、手动书写函数
- **Run-time Settings**

# Run-time Settings

---

## ■ 代码执行顺序

- **vuser\_init**
- **Action**
- **vuser\_end**

## ■ 增加Action2后执行顺序

## ■ 在每个函数中调用lr\_output\_message(“init/action/end”);

# Run-time Settings

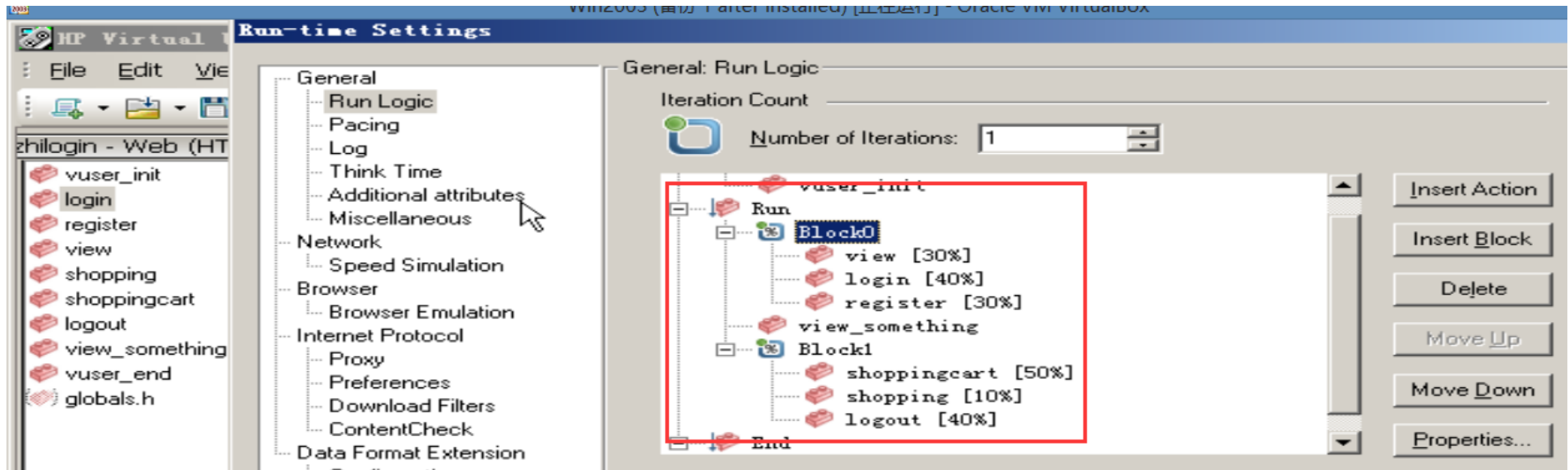
## ■ Run Logic

- action 设置迭代次数，然后执行
- 两个Action分别设置不同的百分比

- 编写一个逻辑30%用户登录，40%用户注册，30%直接访问，然后查看具体商品，之后，10%的用户购买商品，50%的用户将商品加入购物车，40%的用户退出

# Run-time Settings

## ■ Run Logic



# Run-time Settings

- 思考：50%的用户添加购物车后，继续选择商品（具体的数值暂估不出来），或者用户反复做某些操作，但不知道次数，怎么办？

— 扩展逻辑开发

```
Action(){  
    int v;  
    v = rand()%100 ;  
    if(v < 50){ Action2();}  
    else{ Action3();}  
}
```

如果脚本能100%模拟用户，那么结果就会代表真实情况

# Run-time Settings

---

## ■ pacing

- 迭代与迭代之间的时间差

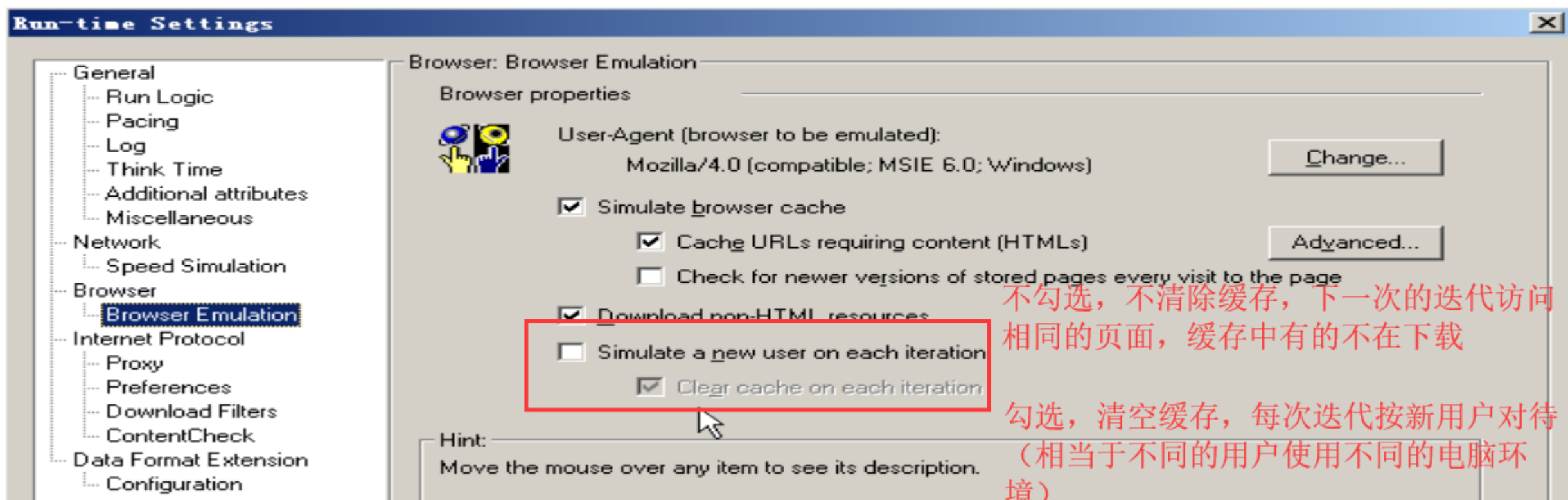
## ■ Think time

- 思考时间
- `lr_think_time ()`
- 为什么要用think-time?
- Think Time设置多长时间合适?

# Run-time Settings

## ■ Browser Emulation

### — 浏览器仿真

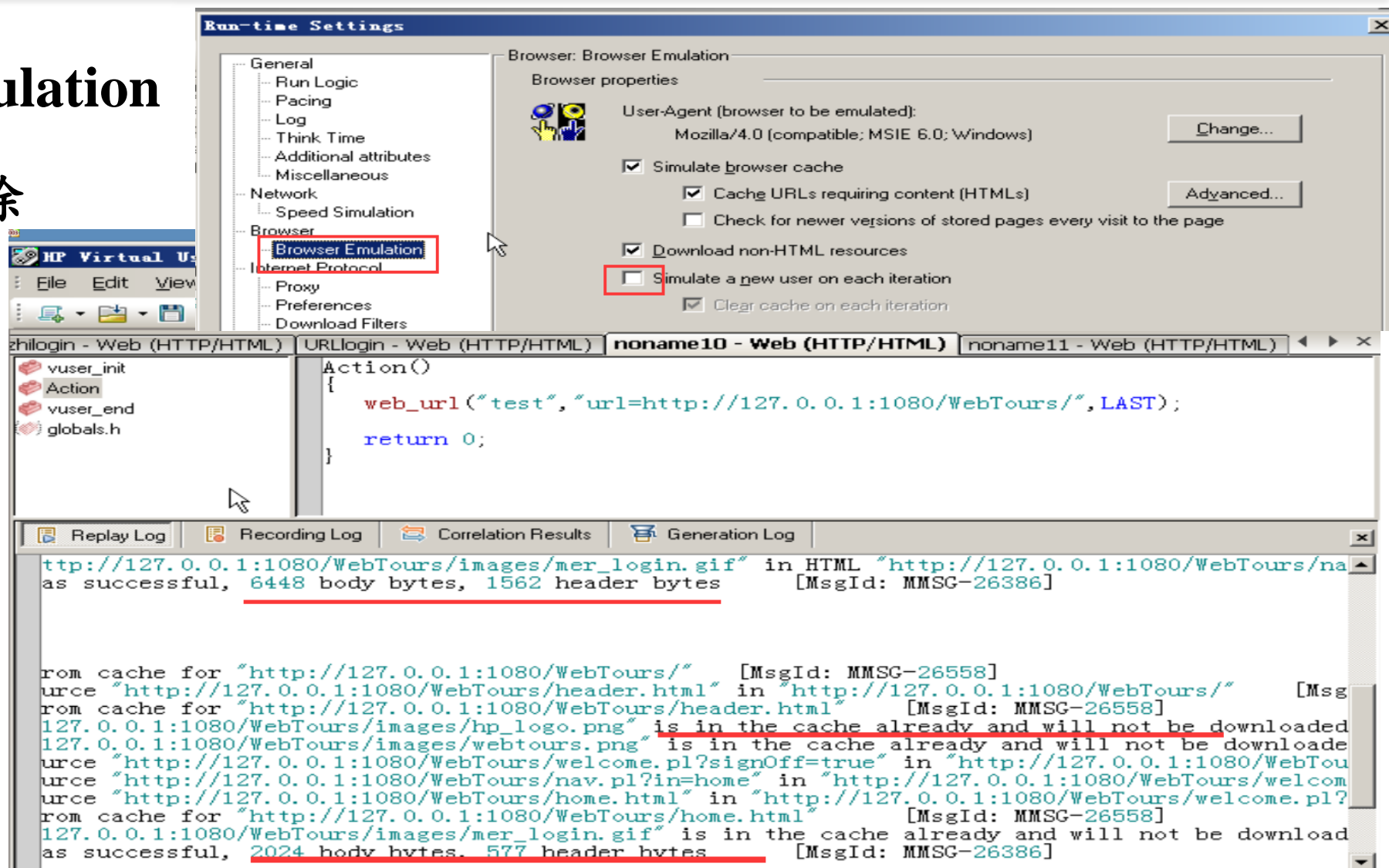


# Run-time Settings

## ■ Browser Emulation

— 不勾选清除

运行结果



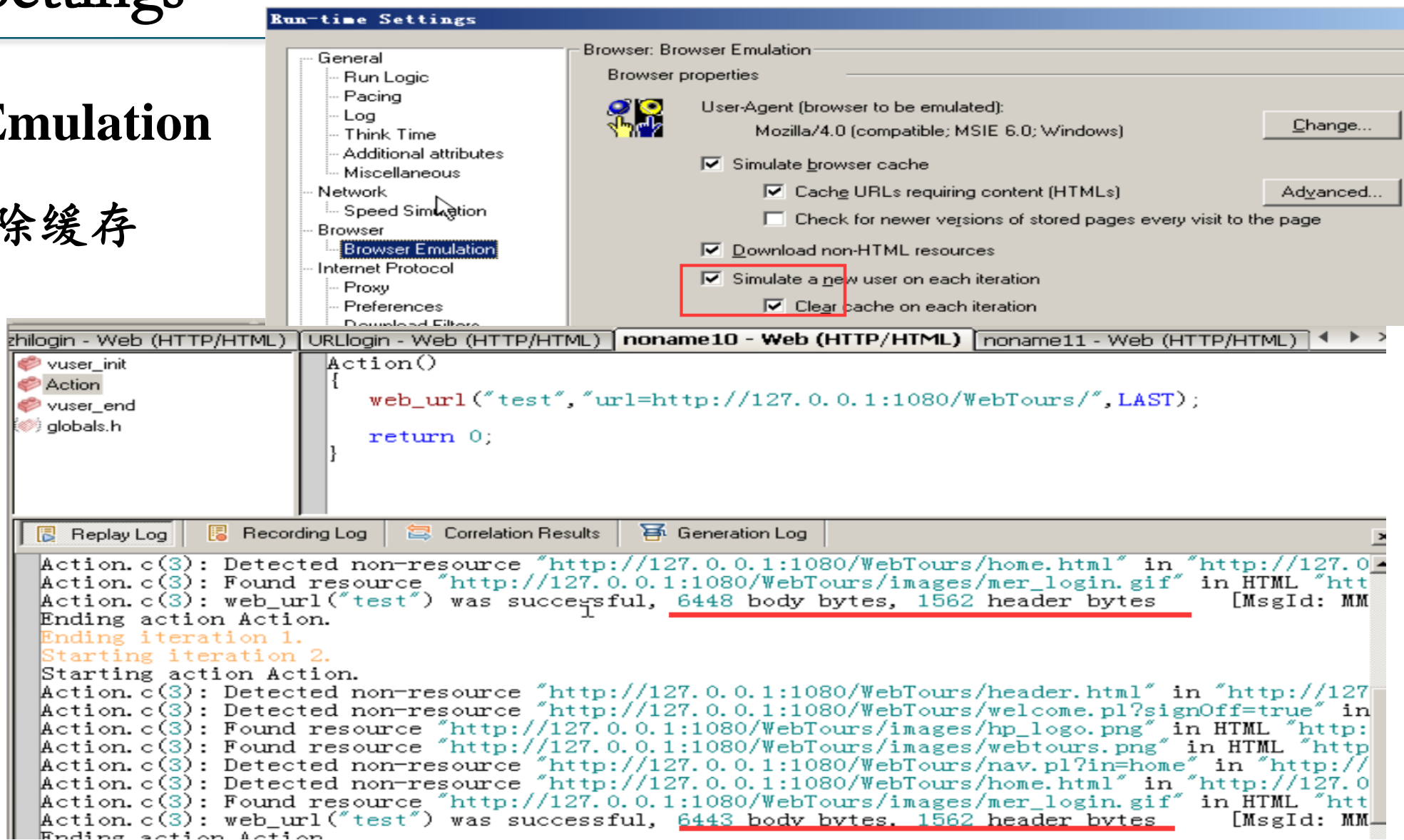


# Run-time Settings

## Browser Emulation

勾选清除缓存

运行结果



The screenshot displays the 'Run-time Settings' dialog box with the 'Browser Emulation' tab selected. In the 'Browser properties' section, the 'Simulate a new user on each iteration' and 'Clear cache on each iteration' options are checked and highlighted with a red box. Below the dialog box, the application interface shows a code editor with the following code:

```
Action()
{
    web_url("test", "url=http://127.0.0.1:1080/WebTours/", LAST);
    return 0;
}
```

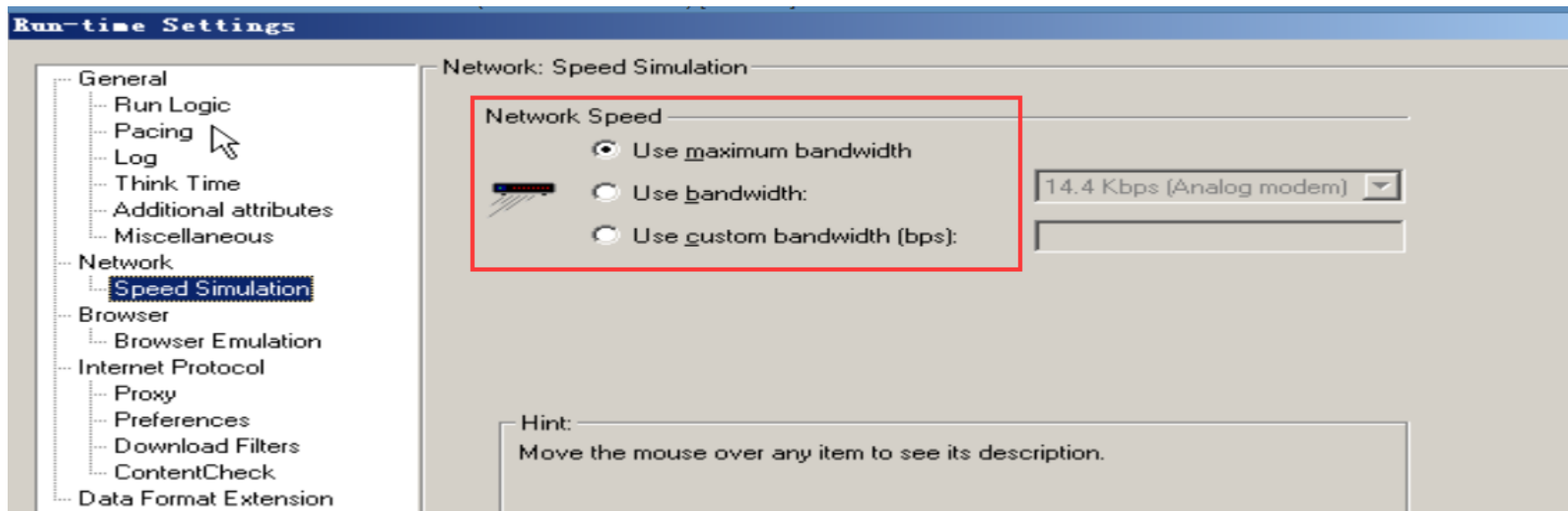
The bottom panel shows the 'Replay Log' with the following output:

```
Action.c(3): Detected non-resource "http://127.0.0.1:1080/WebTours/home.html" in "http://127.0.0.1:1080/WebTours/"
Action.c(3): Found resource "http://127.0.0.1:1080/WebTours/images/mer_login.gif" in HTML "http://127.0.0.1:1080/WebTours/"
Action.c(3): web_url("test") was successful, 6448 body bytes, 1562 header bytes [MsgId: MM]
Ending action Action.
Ending iteration 1.
Starting iteration 2.
Starting action Action.
Action.c(3): Detected non-resource "http://127.0.0.1:1080/WebTours/header.html" in "http://127.0.0.1:1080/WebTours/"
Action.c(3): Detected non-resource "http://127.0.0.1:1080/WebTours/welcome.pl?signOff=true" in "http://127.0.0.1:1080/WebTours/"
Action.c(3): Found resource "http://127.0.0.1:1080/WebTours/images/hp_logo.png" in HTML "http://127.0.0.1:1080/WebTours/"
Action.c(3): Found resource "http://127.0.0.1:1080/WebTours/images/webtours.png" in HTML "http://127.0.0.1:1080/WebTours/"
Action.c(3): Detected non-resource "http://127.0.0.1:1080/WebTours/nav.pl?in=home" in "http://127.0.0.1:1080/WebTours/"
Action.c(3): Detected non-resource "http://127.0.0.1:1080/WebTours/home.html" in "http://127.0.0.1:1080/WebTours/"
Action.c(3): Found resource "http://127.0.0.1:1080/WebTours/images/mer_login.gif" in HTML "http://127.0.0.1:1080/WebTours/"
Action.c(3): web_url("test") was successful, 6443 body bytes, 1562 header bytes [MsgId: MM]
Ending action Action.
```

# Run-time Settings

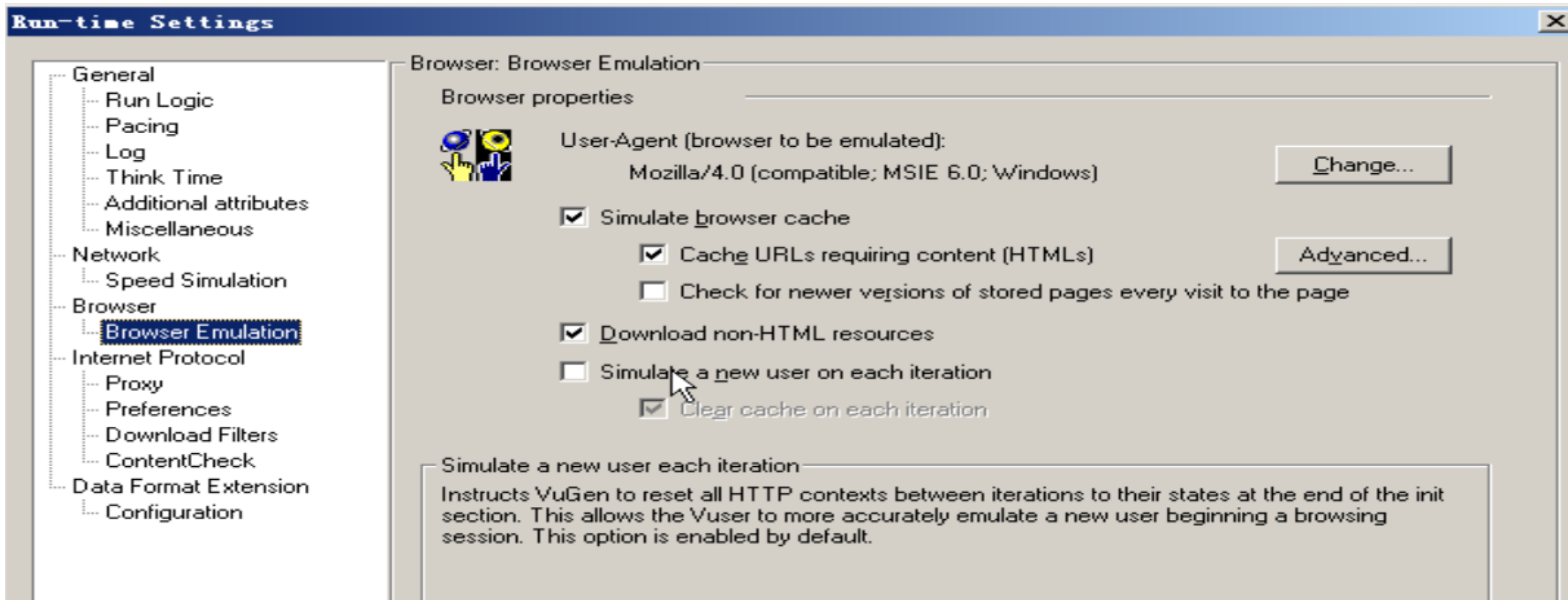
## ■ Speed Simulation

### — 带宽模拟



# Run-time Settings

## ■ Browser Emulation



## 思考

- 1 用户的登录写在vuser\_init，退出写在vuser\_end中，用户的实际操作写在Action中
- 2 用户的登录、退出和实际操作都写到Action中
- 3 模拟一个用户登录、购买、退出，然后下一个用户登录、购买、退出，直到模拟1000个用户后停止？还是模拟1000个用户在线买东西，这些用户有的在登录，有的在购买，有的在退出？

# 内容总结

---

- LoadRunner开启
- 协议选择
- 录制选项设置
- 录制登录操作、手动书写函数
- Run-time Settings



# Question

---