Software Test & Performance Conference 2005

Load Generation in Complex Environments

Alexander Podelko
apodelko@yahoo.com

Agenda

- ***Load Testing***
- **Record and Playback, Virtual Users**
- **Alternatives**

Load Testing

- Testing multi-user applications for performance is a must today
- You never know how an application will work with 1,000 users until you test
- What you need to do significantly depends on your environments

Typical Questions

- What would be response times for 100 concurrent users?
  - Performance / load testing
- What happens under excessive load?
  - Stress testing
- What hardware do we need for 100 users?
  - Capacity planning

Terminology

- Multi-user load on the system
  - Load testing
  - Performance testing
  - Stress testing
  - Scalability testing
  - Volume testing
  - Reliability testing

Hyperion Solutions

- Presentation is based on Hyperion performance team experience

- Hyperion Solutions is a vendor of Business Performance Management software
  - Revenues of $703 millions in fiscal 2005
  - Packaged applications and tools

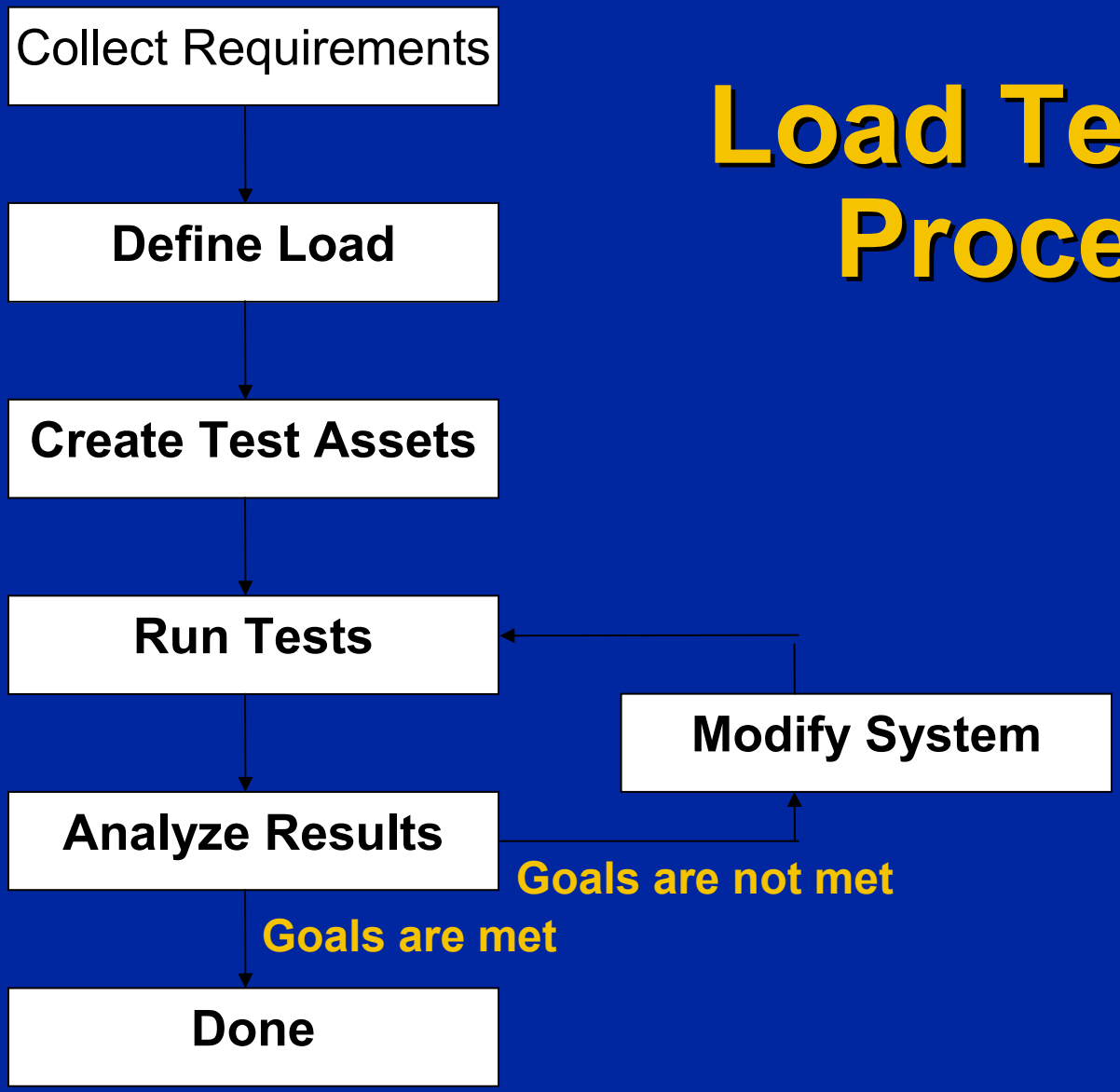
Performance Testing at Hyperion

- Centralized Performance Engineering Group was created in 1997
- Lab environment & customer sites
- Numerous products and configurations
- Now each development group does its own performance testing

All Stages of Software Life Cycle

- Technology evaluation
- Prototypes / POC
- Component / unit
- Pre-release / release
- Benchmarking
- Before going live
- Performance issues in production

Load Testing Process



Load Generation

- Create tests assets – run test
- A “must” task for load testing
- “Tests assets” - usually scripts or programs in load testing
- Time constraints can make it very challenging
  - Different for each product / interface

Workload

- A good workload for performance testing should be:
  - Measurable
  - Reproducible
  - Static
  - Representative

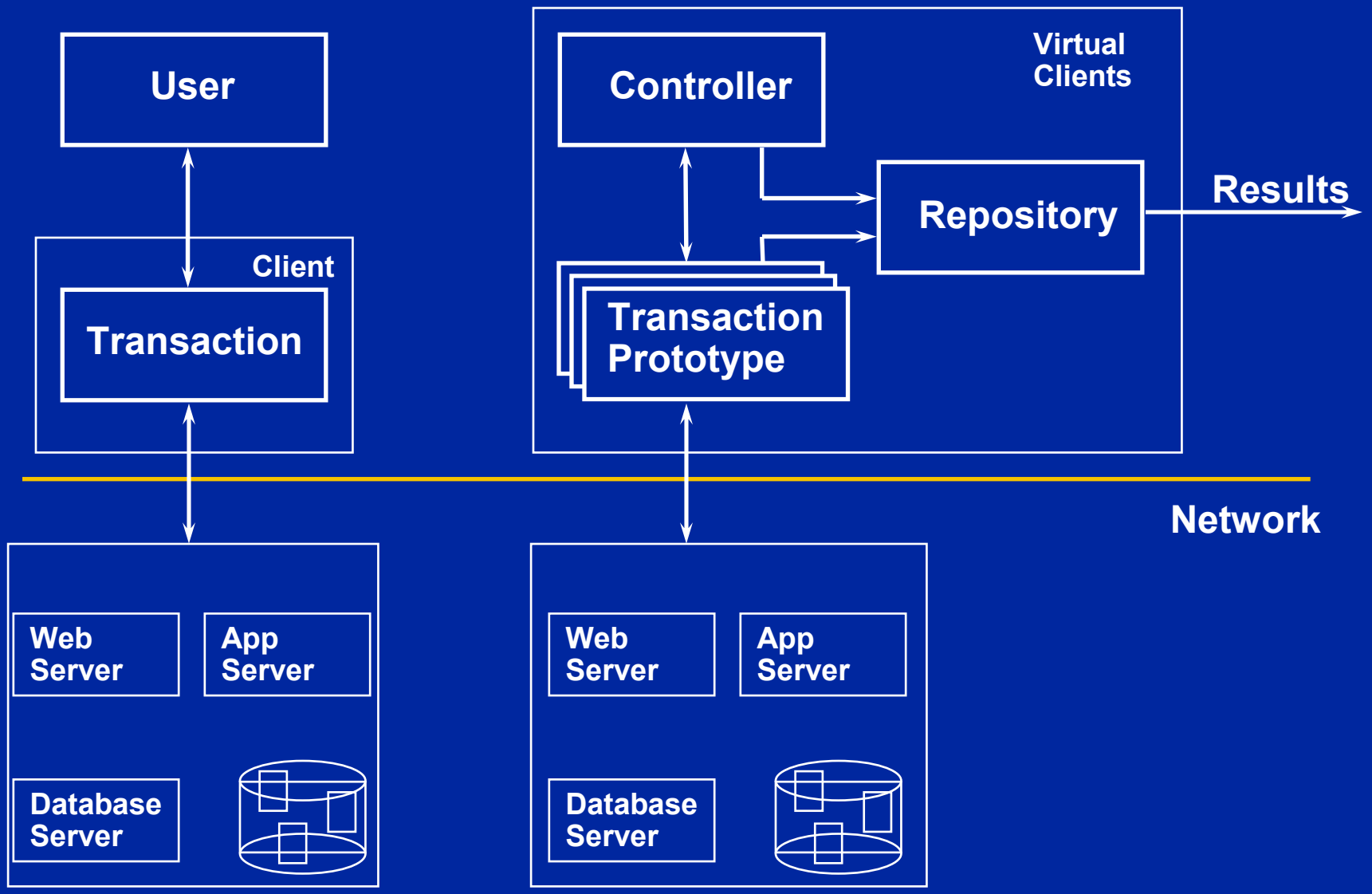
Agenda

- Performance Testing
- ***Record and Playback, Virtual Users***
- Alternatives

"Record and Playback"

- Virtual users: record communication between two tiers and then playback an automatically created script
- Hyperion successfully used this approach in most project since 1997
- Hyperion used two load testing tools: Mercury LoadRunner and Rational Test (Performance Studio, preVue)

Virtual User Simulation



Load Testing Tools

- List of supported features differs significantly from tool to tool
- Universal powerful tools:
  - Segue SilkPerformer (www.segue.com)
  - Rational Performance Tester (www.rational.com)
  - Compuware QA Load (www.compuware.com)
  - Mercury LoadRunner (www.mercury.com)

Features of Universal Tools

- Ability to record scripts automatically for different protocols
- Advanced script language
- A number of simulated users limited mainly by available hardware
- Centralized test management and result analysis

Features of Universal Tools

- Ability to monitor environment
- Ability to support other approaches to load generation
  - Ability to call external functions
  - Ability to simulate GUI users as well as virtual users
- Interfaces to other software

Other Load Testing Tools

- A lot of specialized tools
  - www.softwareqatest.com/qatweb1.html
  - testingfaqs.org/t-load.html
- Empirix (Web)
  - Same scripts for functional and performance testing
- Microsoft Application Center Test (ACT) comes with Visual Studio .Net
  - Visual Studio 2005 Team System for Testers

Open Source

- OpenSTA (www.opensta.org)
  - HTTP/S
- Apache JMeter (jakarta.apache.org/jmeter)
  - Web, JDBC
- www.opensourcetesting.org/performance.php
  - List of 22 open source tools
- Eclipse Test & Performance Tools Platform (www.eclipse.org/tptp)

Other Ways

- Appliances
  - For example, Spirent Avalanche, Antara FlameThrower, and Ixia products
  - can be useful for simulation of a large number of simple Web users
  - Limited parameterization
- Outsourcing / Services

Problems

- "Record and playback" approach often doesn't work for testing components
- Each load testing tool support a limited number of technologies (protocols)
- Hyperion had several problems back in 1999

Hyperion's Problems Back in 1999

- Hyperion Enterprise - SMB (Server Message Block) protocol
- Hyperion Financial Management - DCOM
- Hyperion Reports - Java RMI

Agenda

- Performance Testing
- Record and Playback, Virtual Users
- ***Alternatives***

Alternatives

- Manual
- Record and Playback, GUI Users
- Programming
- Mixed / Custom Load Generation

Manual

- Not an option for a large number of users
- Always variation in human input times
- Can be a good option to simulate quickly a few users
- Can be used with other methods to verify correctness

GUI Users

- Functional / regression testing tools
  - WinRunner, QuickTest Pro, Rational Robot, etc.
- Record and playback communication between user and client GUI
- Don't care about communication protocols / internals
- Accurate data (real client, end-to-end)

GUI Users

Requires a real machine for each user

- Mercury can use one Windows Terminal session per user, so running several GUI users on the box
- Another workaround from Mercury is using low-level graphical Citrix protocol

Custom Test Harness

- Special program to generate workload
- Requires access to the API or source code
- Requires programming
- Could be cost effective solution in some simple cases

Advantages

- Doesn’t require any special tool
- Starting version could be quickly created by a programmer familiar with API
- Should work if API works
- You don't care what protocol is used for communication

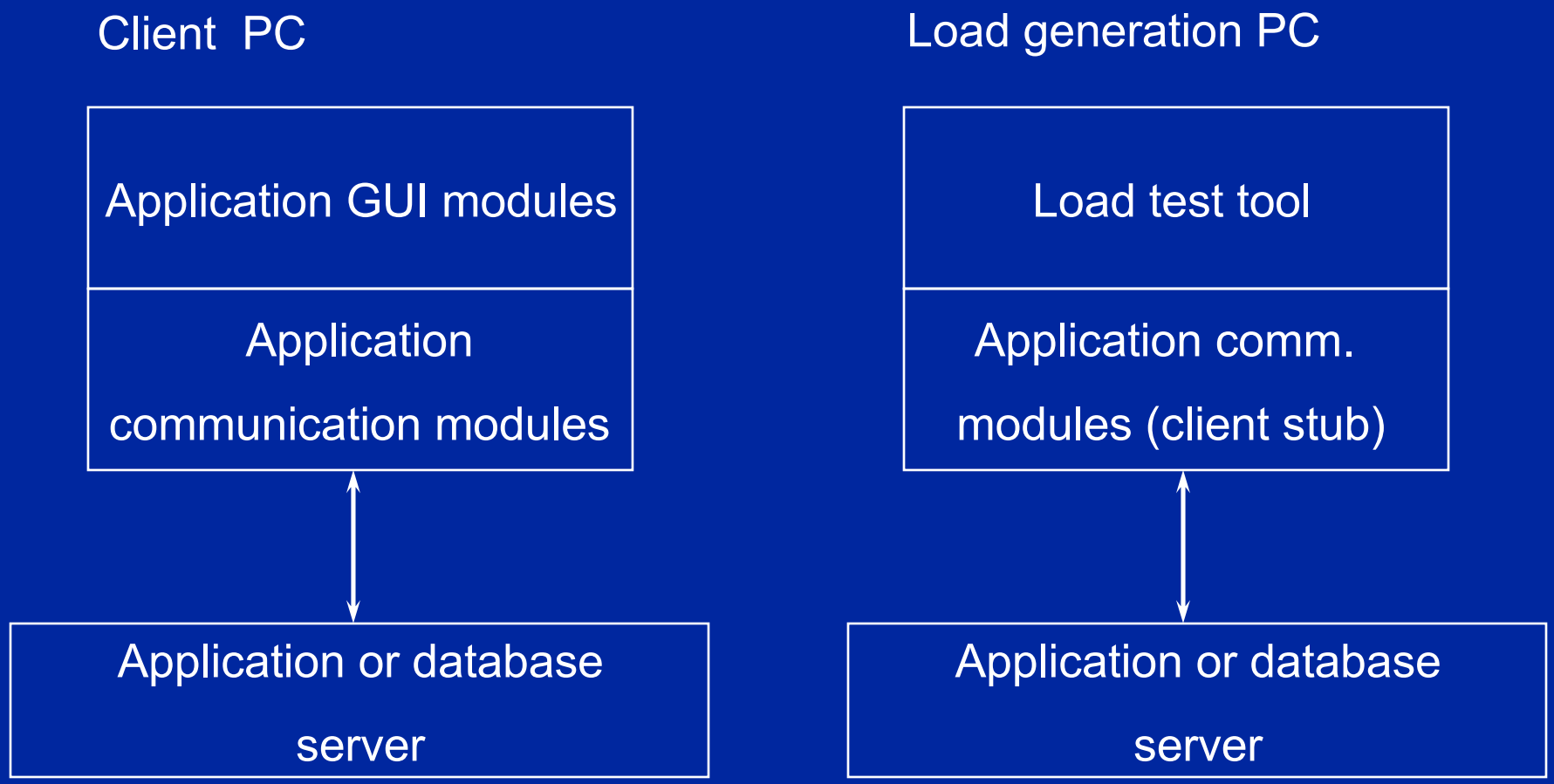
Disadvantages

- Efforts to update and maintain harness can increase drastically
- When you have numerous products you really need to create something like a commercial load testing tool

Custom Load Generation

- Mixed approach
  - Lightweight custom client stubs to work with an application
  - Commercial load testing tool to manage these stubs and analyze results
- Implementation depends on the particular tool
  - Hyperion used Rational Test and Mercury LoadRunner

Custom Load Generation



Implementation

- We did it for LoadRunner and Rational Test
- Standard external DLL in C/C++
- API calls directly inserted into scripts – for scripts in Java, for example

Advantages

- Eliminates dependency on supporting specific protocols
- Leverages all the features of the load testing tool and allows using it as a test harness
- Sometimes simplifies work with difficult to parameterize protocols

Considerations

- Requires access to API or source code
- Requires programming
- Minimal transaction that could be measured is an external function
- Requires understanding of internals

Recording vs. API

- **RMI recording**

```
_integer =
   _ireportserver.executeJob(_designjobobject);
_ireportserver.getStatus(new Integer(3));
_ireportserver.getStatus(new Integer(3));
_ireportserver.getStatus(new Integer(3));
_iinstance =  _ireportserver.getInstance(new Integer(3));
```

- **Real code**

```
joID = poReportServer.executeJob(djo);
bStatus = true;
while (bStatus) {
   bStatus = poReportServer.getStatus (joID);
   Thread.sleep(300); }
 poReportServer.getInstance(joID);
```

More Considerations

- Requires a load test tool license for the necessary number of virtual users
- Environment should be set on all agents
- Usually requires more resources on agent machines
- Results should be cautiously interpreted

If Difficult to Parameterize…

- Recording and parameterization of a script could be time-consuming
- “Custom load generation" approach sometimes can be a better choice

Example 1: Essbase Query

- Multi-Dimensional Database
- C API
  - Used by many applications and middleware
  - Winsock scripts
- Quite difficult to parameterize and verify
- External DLL was made for major functions

Winsock Script

```
lrs_create_socket("socket0", "TCP", "LocalHost=0",
"RemoteHost=ess001.hyperion.com:1423",
LrsLastArg);
 lrs_send("socket0", "buf0", LrsLastArg);
 lrs_receive("socket0", "buf1", LrsLastArg);
 lrs_send("socket0", "buf2", LrsLastArg);
 lrs_receive("socket0", "buf3", LrsLastArg);
 lrs_save_searched_string("socket0",
      LRS_LAST_RECEIVED, "Handle1",
                "LB/BIN=\\x00\\x00\\v\\x00\\x04\\x00",
      "RB/BIN=\\x04\\x00\\x06\\x00\\x06", 1, 0, -1);
 lrs_send("socket0", "buf4", LrsLastArg);
 lrs_receive("socket0", "buf5", LrsLastArg);
 lrs_close_socket("socket0");
```

Winsock Script

```
send  buf22 26165
   "\xff\x00\xf0\a"
   "\x00\x00\x00\x00\x01\x00\x00\x00\x01\x00\x03\x00"
   "d\x00\b\x00"
   "y'<Handle1>\x00"
   "\b\r\x00\x06\x00\f\x00\x1be\x00\x00\r\x00\xd6\aRN"
   "\x1a\x00\x06\x00\x00\x00\x00\x00\x00\x00\x00\b"
   "\x00\x00\x00\xe7\x00\x00\x01\x00\x03\x00\x04\x00"
   "\x10\x00\xcc\x04\x05\x00\x04\x00\x80\xd0\x05\x00\t"
   "\x00\x02\x00\x02\x00\b\x00<\x00\x04"
   "FY04\aWorking\tYearTotal\tELEMENT-F\tProduct-P"
   "\x10<entity>\t\x00\x02\x00"
   …
```

Script Using External DLL

```
lr_load_dll("c:\\temp\\lr_msas2k.dll");
pCTX = Init_Context();
hr = Connect(pCTX, "ess01", "user001","password");
…
lr_start_transaction("Mdx_q1");
sprintf(report, "SELECT %s.children on columns,
   %s.children on rows FROM Shipment WHERE
   ([Measures].[Qty Shipped], %s, %s)",
   lr_eval_string("{day}"),  lr_eval_string("{product}"),
   lr_eval_string("{customer}"),
   lr_eval_string("{shipper}"));
hr = RunQuery(pCTX, report);
lr_end_transaction("Mdx_q1",LR_AUTO);
```

Example 2: EDS

- Essbase Deployment Services
- Middleware, no GUI interface
- Test scripts in Java from the QA group
- Solution - creation of LoadRunner scripts from the test script

EDS Java Script

```
import lrapi.lr;
import com.essbase.api.base.*;
import com.essbase.api.session.*;
…
public class Actions{
   public int init() {
         return 0;
   }//end of init
   public int action() {
    String s_userName = "system";
    String s_password = "password";
```

EDS Java Script

```
lr.enable_redirection(true);
try {
   lr.start_transaction("01_Create_API_instance");
      ess =
   IEssbase.Home.create(IEssbase.JAPI_VERSION);
   lr.end_transaction("01_Create_API_instance",
   lr.AUTO);
   lr.start_transaction("02_SignOn");
      IEssDomain dom = ess.signOn(s_userName,
   s_password, s_domainName, s_prefEesSvrName,
   s_orbType, s_port);
   lr.end_transaction("02_SignOn", lr.AUTO);
...
```

Summary

- Load testing is a must today for multi-user applications
- Load generation is a must step in load testing, can be challenging in complex environments
- No universal approach – you need to find your own way

Questions?

Alexander Podelko

apodelko@yahoo.com

To learn more check my collection of performance-related links and documents at www.alexanderpodelko.com