Load Testing as a Part of the Software Development Cycle

PHILACMG

November 19, 2004

Alexander Podelko

Hyperion Solutions

Alexander_Podelko@hyperion.com

Agenda

- ***Performance Testing***
- **“Record and Playback“**
- **Alternatives**

Performance 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 what your business is

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

Hyperion Solutions

- Presentation is based on Hyperion performance team experience

- Hyperion Solutions is a vendor of Business Performance Management software
  - Revenues of $622 millions in fiscal 2004
  - 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 makes it own performance testing

All Stages of Software Life Cycle

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

Load Testing Process

- Define (design) what you want to test
- Fill it with test data
- Create workload
- Apply workload to the system
- Analyze results

Workload Development

- Workload development is the most non-trivial part of work according to our experience

- We need to create meaningful and realistic workloads in a timely manner
  - Usually in the development cycle timeframe
  - Different for each product / interface

Workload

- A workload should reproduce the typical stress on a system

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

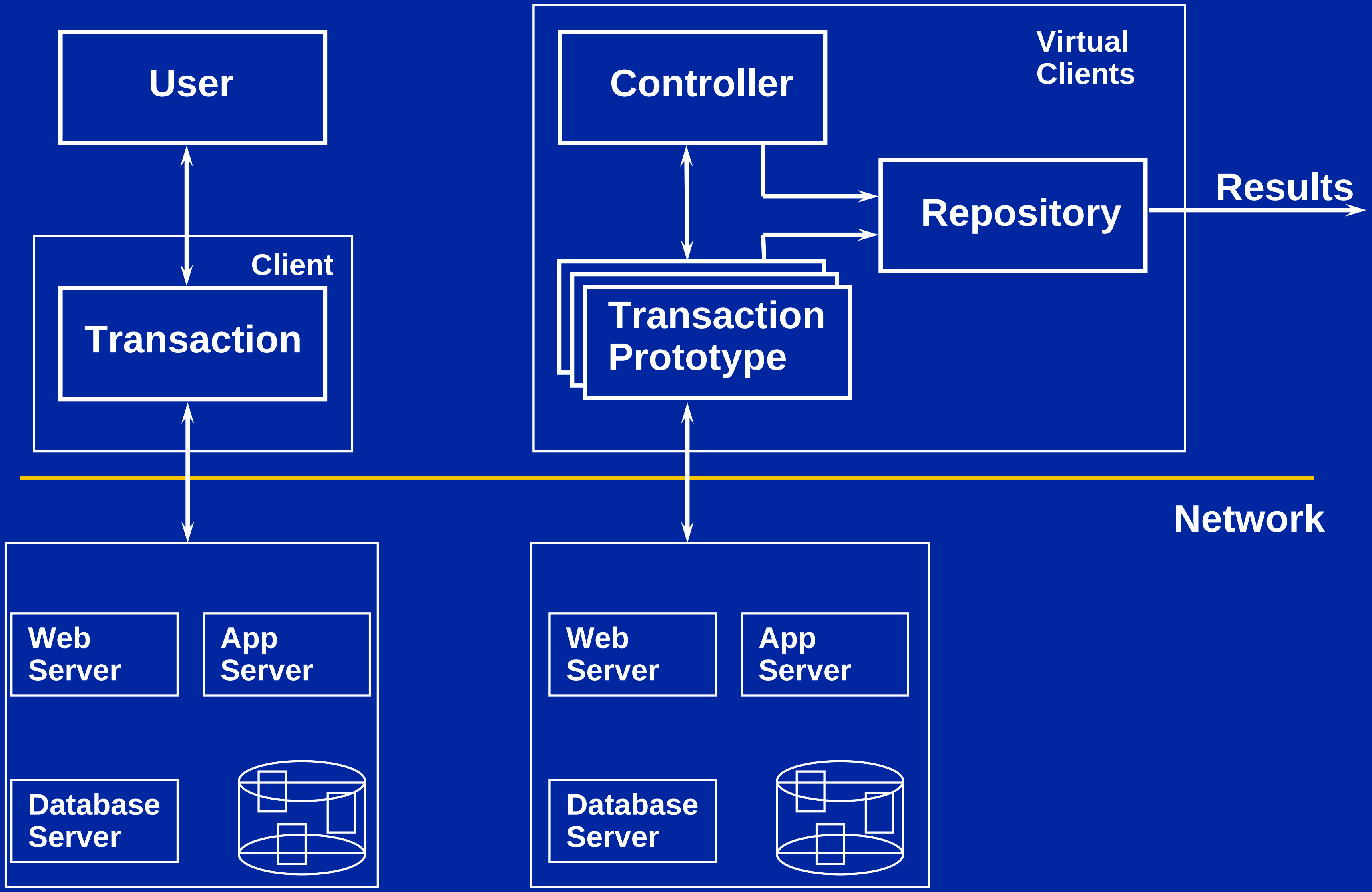
Agenda

- Performance Testing
- *“Record and Playback“*
- Alternatives

"Record and Playback"

- Virtual users: record communication between two tiers and then playback an automatically created script
- We successfully used this approach in most project since 1997
- We 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 Test (www.rational.com)
  - Compuware QA Load (www.compuware.com)
  - Mercury LoadRunner (www.mercury.com)

*All brands and trademarks are the property of their owners

Features of Universal Tools

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

Features of Universal Tools

- Advanced script language for workload development
- Ability to call external functions
- Ability to simulate GUI users as well as virtual users

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

Open Source

- OpenSTA
  - HTTP/S
  - www.opensta.org
- Apache JMeter
  - Web, JDBC
  - jakarta.apache.org/jmeter
- www.opensourcetesting.org/performance.php
  - List of 18 open source tools

Other Ways

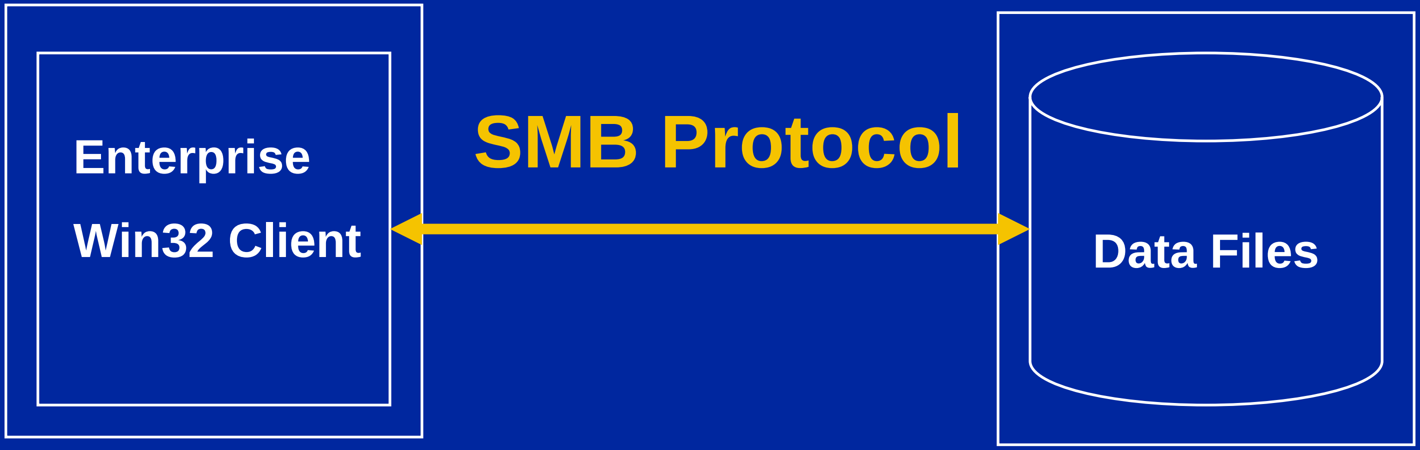
- Appliances
  - For example, Spirent Avalanche
  - can be useful for simulation a big number of simple Web users
  - Limited parameterization
- Outsourcing

Problems

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

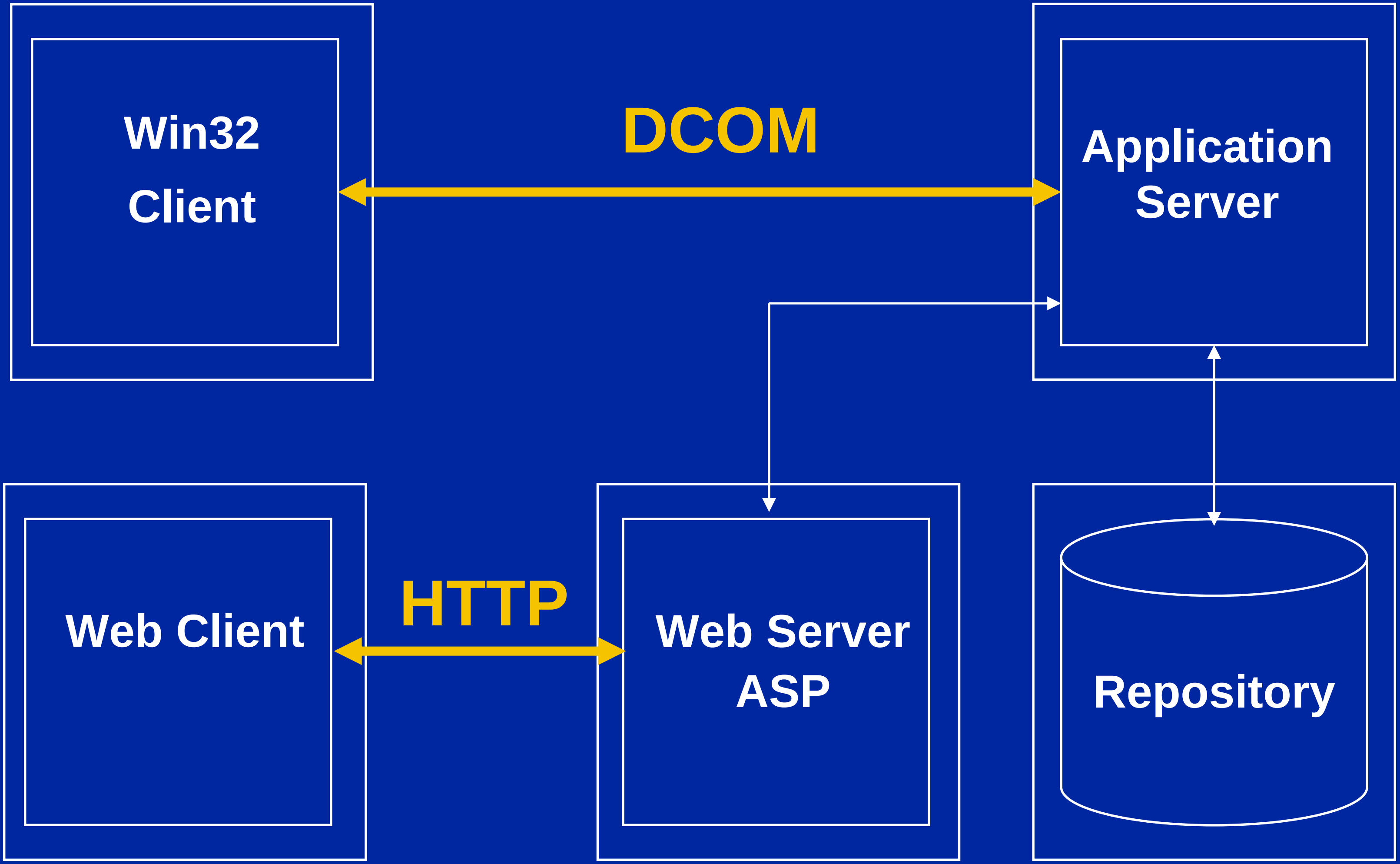
Hyperion Enterprise

Proven financial consolidation, reporting, and analysis



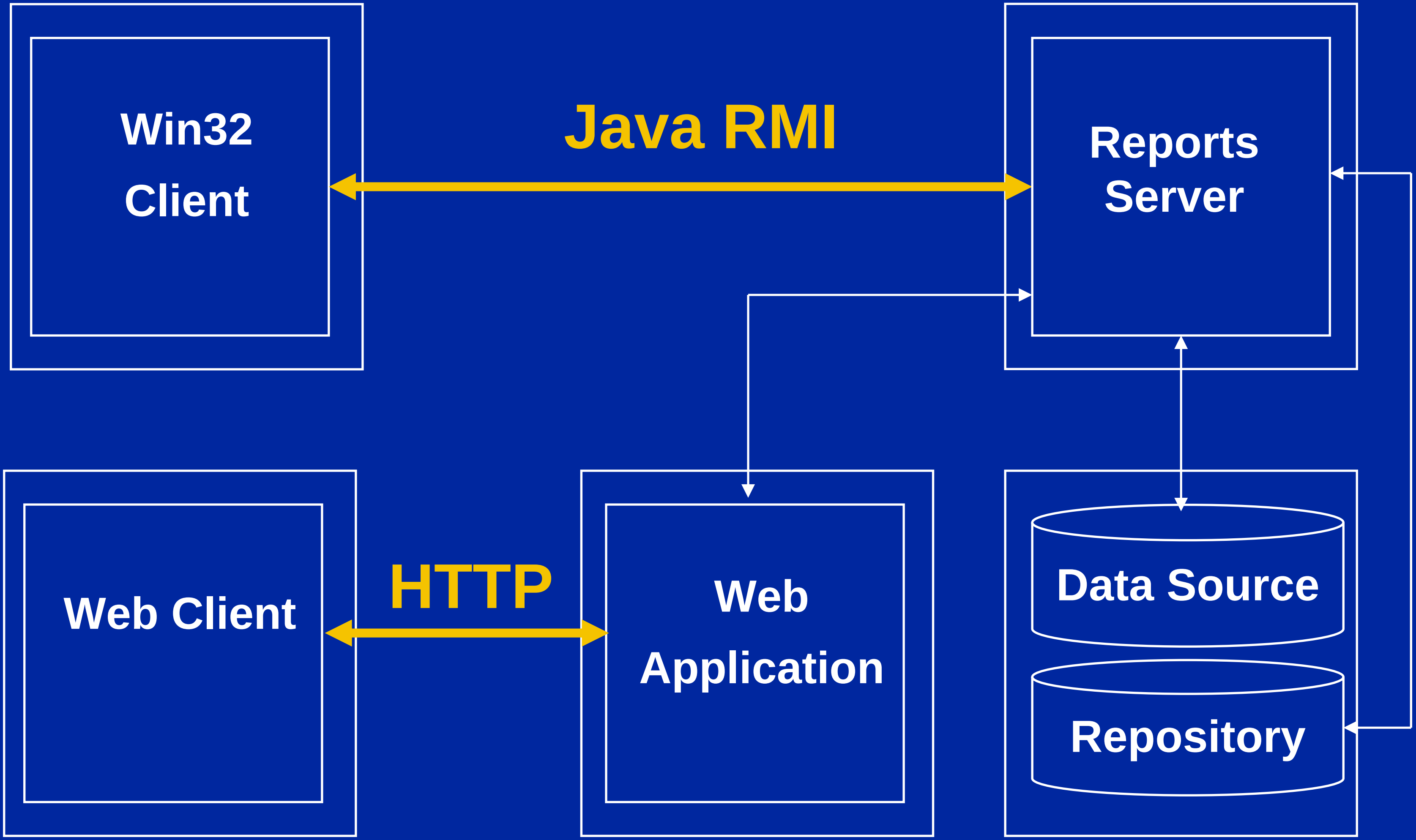
*All brands and trademarks are the property of their owners

Hyperion Financial Management



*All brands and trademarks are the property of their owners

Hyperion Reports



*All brands and trademarks are the property of their owners

Agenda

- Performance Testing
- “Record and Playback“
- ***Alternatives***

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

*All brands and trademarks are the property of their owners

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

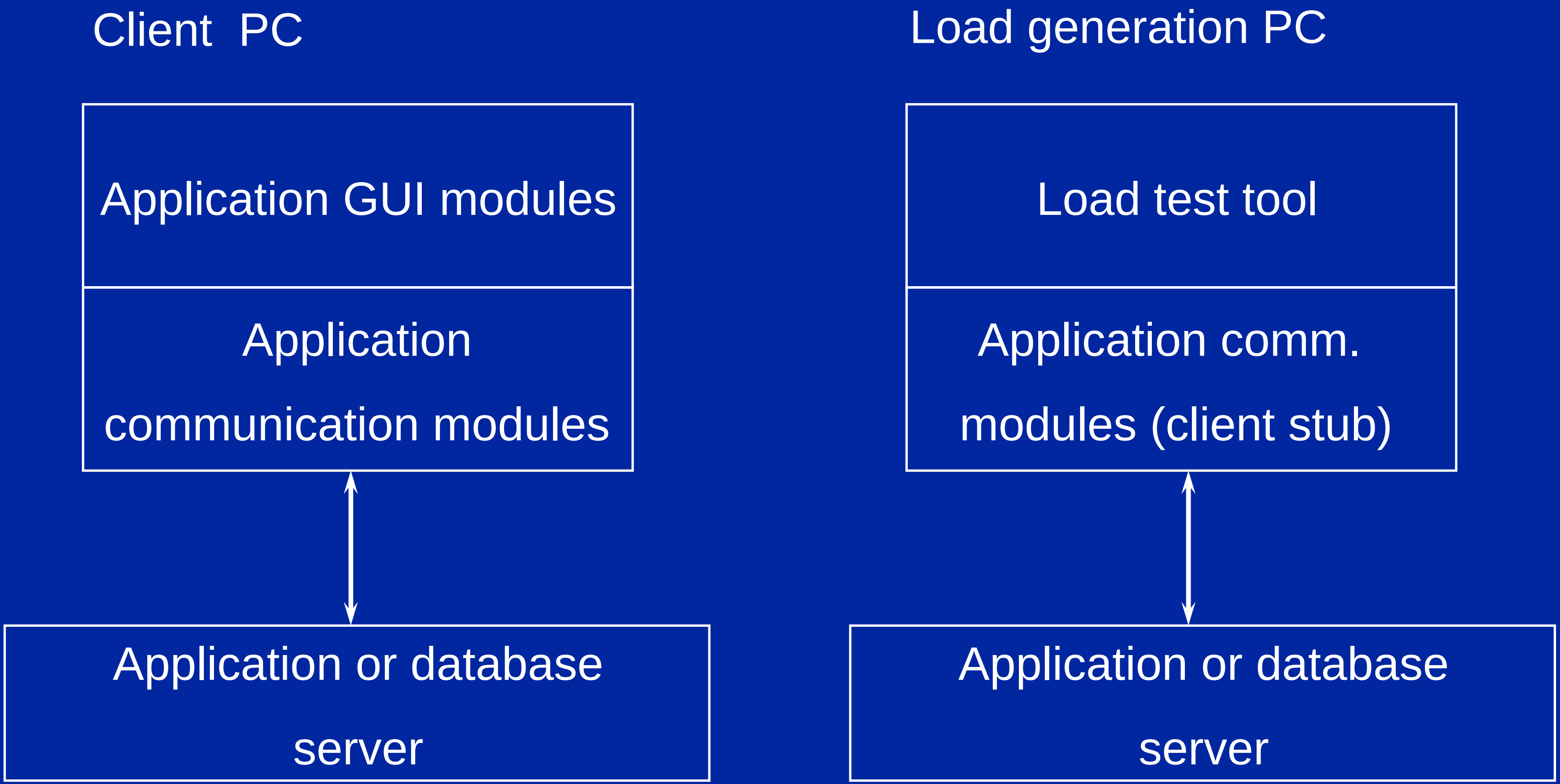
Disadvantages

- 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
  - We used Rational Test and Mercury LoadRunner

Custom Load Generation



Client PC

Application GUI modules

Application communication modules

Application or database server

Load generation PC

Load test tool

Application comm. modules (client stub)

Application or database server

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 LoadRunner 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: 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

```
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\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);
```

Summary

- Performance testing is a must today for multi-user applications

- Performance testing is not one-time action, it is continuous efforts during all software lifecycle

- No universal approach – you need to find your own way

To Learn More

- My collection of link (just started) www.alexanderpodelko.com
- www.perftestplus.com
- www.performancetester.com
- www.loadtester.com
- www.stickyminds.com

Questions ?