2002

freedom TO CREATE

Rational User Conference 2002

Custom Load Generation with Rational TestManager

Alexander Podelko
Arno Sokk
Hyperion Solutions

Rational
the software development company

Session # AT30

Agenda

- *Performance Testing*
- Workload Generation
- “Record and Playback “
- Custom Test Harness
- Custom Load Generation
- Implementation with External DLL
- Implementation with Java Language Test Script

Performance Testing

- Testing multi-user applications for performance is a must today
  - Failure cost could be very high
- You never know how an application will work with 1,000 users until you test
- Manual test usually isn't an option
- What you need to do significantly depends on your business is

Typical Questions

- What would be response times for 500 users?
  - Performance testing
- How many users could be supported?
  - Stress testing
- What configuration do we need for 500 users?
  - Capacity planning

Hyperion Solutions

- This presentation is based on Hyperion's performance team experience
- Hyperion Solutions is a vendor of business performance management software
  - Revenues of more than $500 millions in fiscal 2001
  - Employs more than 2,300 people in 20 countries
  - Business Intelligence Platform: Essbase
  - Packaged Applications: Performance Scorecard, Business Modeling, Planning, Financial Management
  - Used by 6,000 organizations worldwide
    - More than 80 of the Fortune 100

Performance Testing At Hyperion

- Performance group was created in 1997
- Shared between all development groups
- Supplements the functional testing done by QE groups for each product
- Lab environment
- Numerous configuration to test
  - About a dozen of different products
  - Different platforms (Win32 and UNIXes)
  - Different versions, interfaces, etc.

Performance Testing Procedure

- Define (design) what you want to test
  - Usually server part of an application
- Fill it with test data
  - Size of data could change the picture completely
- Define the workload
  - Define user transactions and scenarios
- Apply workload to the SUT
  - Monitoring SUT could give a lot of information
- Analyze results

Agenda

- Performance Testing
- *Workload Generation*
- “Record and Playback “
- Custom Test Harness
- Custom Load Generation
- Implementation with External DLL
- Implementation with Java Language Test Script

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
  - Separate for each product
- Only real way is using automation
  - Manual testing isn't an option even for a few users
    - Although could be used in some cases for one user or to verify correctness of automated workload for a few users

Workload

- A workload should reproduce the typical stress on a system
- A good workload for performance testing should be:
  - Measurable
  - Reproducible
  - Representative

Workload Characteristics

- Measurable - a quantifiable metric which represents performance can be defined
  - Response time
  - Throughput
- Reproducible - the measurement is repeatable and consistent, does not vary with time
- Representative - the work being performed is typical for normal operating conditions

Approaches To Workload Development

- "Record and playback" approach
  - Using commercial test tools
  - Record communication and then playback an automatically created script
    - Usually, of course, after proper parameterization
- Custom test harness
  - Special program to generate workload
- Custom load generation
  - Mix of two above

Agenda

- Performance Testing
- Workload Generation
- *“Record and Playback “*
- Custom Test Harness
- Custom Load Generation
- Implementation with External DLL
- Implementation with Java Language Test Script

"Record And Playback"

- Virtual users: record communication between two tiers and then playback an automatically created script
  - Usually, of course, after proper parameterization
  - The main way to create heavy workload
- GUI users: record and playback communication between user and client GUI
  - More accurate data (real client, end-to-end)
  - Requires a real machine for each user. Doesn't fit our requirements: won't be discussed further
  - Could be useful in combination with virtual users

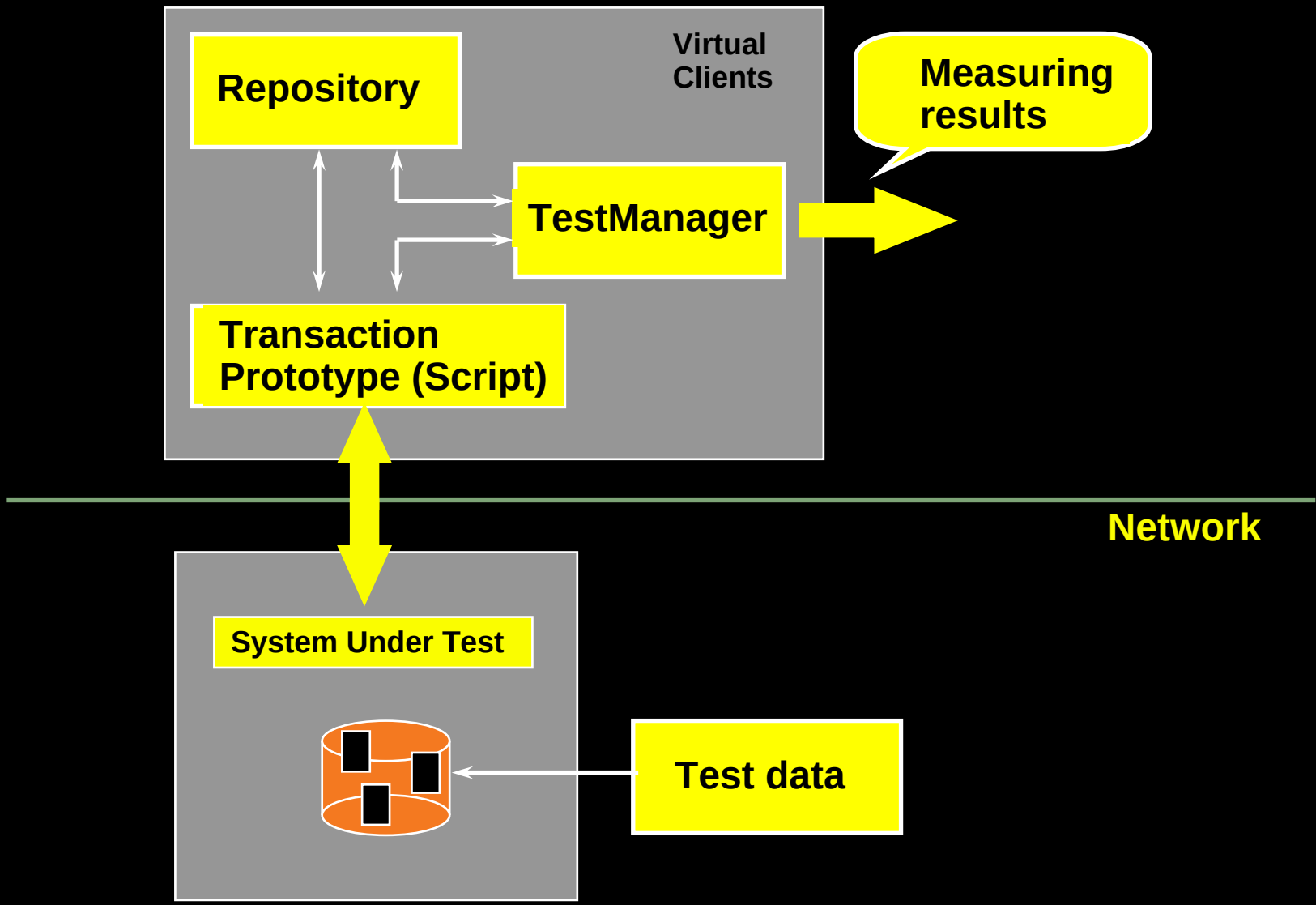
Our Experience

- Software, generating workload by simulation of numerous virtual users, is usually referenced as a load testing tool
- We successfully use load testing tools from Rational since 1997 in our group: preVue, Rational Suite Performance Studio, Rational Test
- Choosing the right tool is a large topic that is outside the range of this presentation

Rational Test Features

- Ability to record scripts automatically for different protocols
- Advanced script language for workload development
- The number of simulated users is limited mainly by available hardware
- Centralized test management and result analysis
- Coordinated test execution from several computers
- Ability to simulate GUI users as well as virtual users

Virtual User Simulation



Problems

- "Record and playback" approach doesn't work for testing components
  - We developed a custom test harness to test components
  - There are some tools for performance testing of components now
- A particular load testing tool supports the limited number of technologies (protocols)
  - New or exotic technologies are not on the list
  - We hadn't been able to use this approach for several technologies

Problem 1

- SMB (Server Message Block) protocol, later succeeded by Common Internet File System (CIFS)
  - Used when two Microsoft systems communicate over network
  - Its commands are embedded within the transport protocol like TCP/IP
  - Rational Test tools do not support automated record & playback of SMB and CIFS traffic

Problem 2

- Microsoft DCOM
  - Used for communication between two remote COM components
  - At the moment of evaluation (1999) only Mercury claimed DCOM support, but it didn't work in our environment
  - Rational Test tools provide automated DCOM recording now
  - Since "custom load generation" approach worked fine, we didn't return to the evaluation of DCOM "record and playback" approach

Problem 3

- Java RMI (Remote Method Invocation)
  - Used for communication between two remote Java programs
  - At the time of evaluation (1999) nobody supported Java RMI over JRMP (Java Remote Messaging Protocol)
  - Rational Test does not currently support RMI recording

Agenda

- Performance Testing
- Workload Generation
- “Record and Playback “
- *Custom Test Harness*
- Custom Load Generation
- Implementation with External DLL
- Implementation with Java Language Test Script

Custom Test Harness

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

Advantages

- Doesn’t require any special tool
  - Except for development environment, of course
- Starting version could be quickly created by a programmer familiar with API
  - Simple harness could spawn some threads and each thread will simulate a user
- Should work if API works
- You don't care what protocol is used for communication
  - It is hidden behind API

Disadvantages

- Efforts to update and maintain harness increase drastically as soon as you need:
  - Complex user scenarios
  - Centralized test management and result analysis
  - Coordinated test execution from several computers
  - Ability to run GUI as well as virtual users
- When you have numerous products you really need to create something like a commercial load testing tool
  - Probably isn't the best choice for a small group

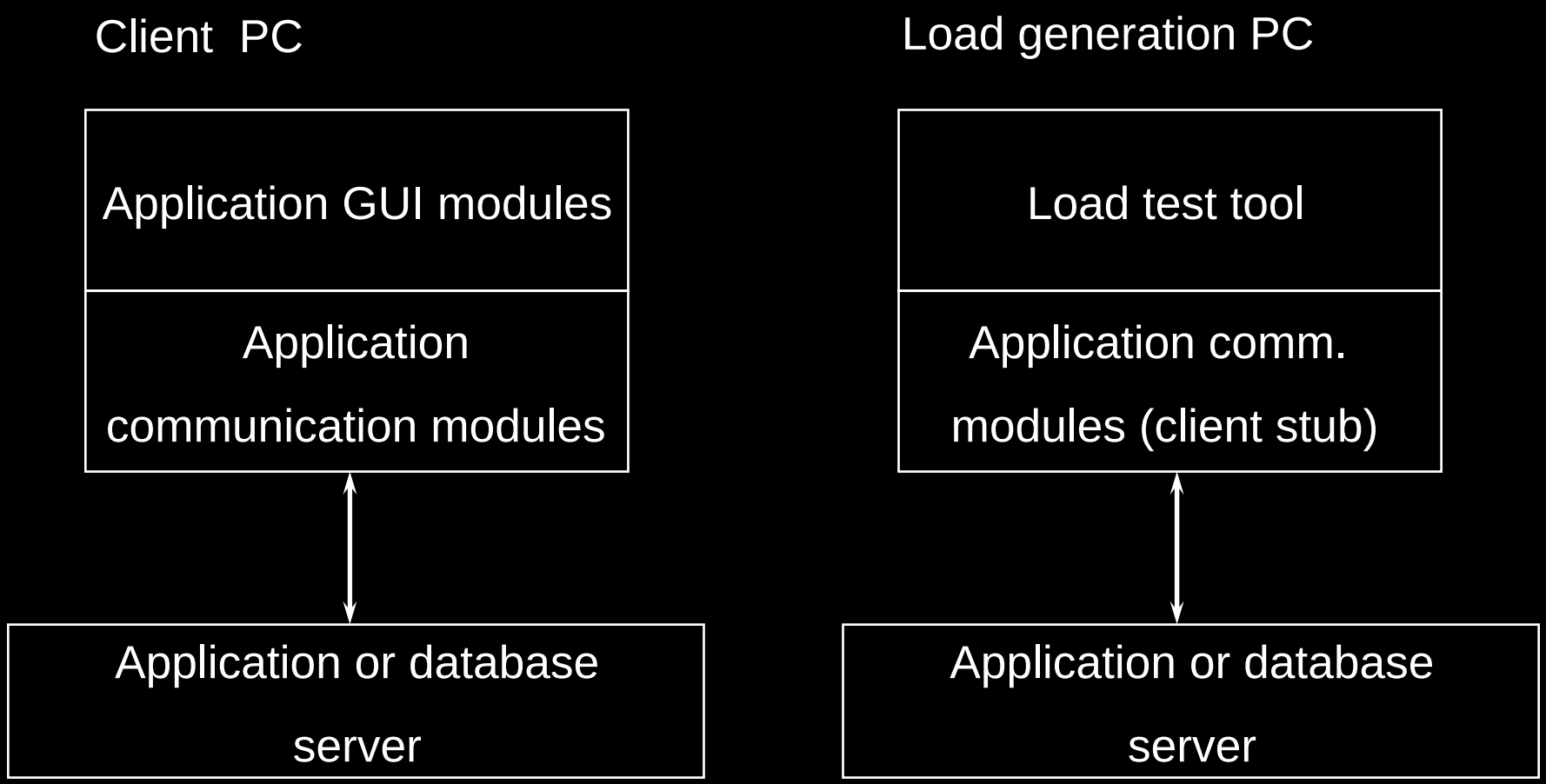
Agenda

- Performance Testing
- Workload Generation
- “Record and Playback “
- Custom Test Harness
- *Custom Load Generation*
- Implementation with External DLL
- Implementation with Java Language Test Script

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
- There are several ways to implement it with Rational TestManager
  - External DLL
  - Test Script Services (TSS)
    - Java Language Test Script
    - VB Language Test Script
    - Command Line Test Script
- We successfully use external DLLs and Java Language Test Scripts

Custom Load Generation



Advantages

- Eliminates dependency for third-party tool to support specific protocols
  - As it was for SMB, DCOM and RMI in our case
  - Allows to keep pace with new technologies
- Leverages all the features of a load testing tool and allows using it as a test harness
  - No need to implement multi-user support, data collection and analysis, reporting, scheduling, etc.
- Sometimes allows managing the workload in a more user-friendly way

User-Friendly Way

- Recording and parameterization of a recorded script could be time-consuming
- Example: running a query
- If you want to change the query in the workload or use a new build you need to record and parameterize the script again !!!
  - It happens very often in our environment
- With "custom load generation" the real query could be read from file
  - You just change the file without any changes in the script

User-Friendly Way #2

- When you change a build you probably need to recompile custom client software
  - It is usually less time consuming than recording and parameterizing
  - If API changed probably some changes in the source code would be necessary

Considerations

- Requires access to API or source code
- Requires additional programming work
- Requires a commercial tool license for necessary number of virtual users
- Minimal transaction that could be measured is an external function
- Usually requires more resources on client machines
- Results should be cautiously interpreted
  - Insure that there is no contention between client stubs

Agenda

- Performance Testing
- Workload Generation
- “Record and Playback “
- Custom Test Harness
- Custom Load Generation
- *Implementation with External DLL*
- Implementation with Java Language Test Script

Implementation With External DLL

- The mature way to create a custom client stub is an external DLL
- Should be C functions compiled as DLL
  - Functions can be written in C++ and declared as *extern "C"*
- DLL should be placed into specific directory and specified in the environment
- Only limited set of types can be function arguments
- If agents are used, external DLL as well as client software (API) should be placed on agent machines too

Supported Arguments Types

- C types
  - int
  - char *
  - char *
  - int *
  - char **
- VU script types
  - int
  - string /*read only */
  - string:maxsize /*writable*/
  - int[ ], int[ ][ ], int[ ][ ][ ]
  - string[ ],string[ ][ ],string[ ][ ][ ]

External DLL

```
extern "C“ {
__declspec(dllexport) void DoSomething(); }
#include "windows.h"
void DoSomething (int n)
{
...             //some processing
}
```

Requirements For VU script

- VU script is a C-like language to program in Rational Test
- External modules should be specified in the environment
- Functions from external DLL should be described as *external_C* in the VU script
- Then the functions from external DLL could be called in the script body just as any built-in function

External DLL Call From VU Language Script

```
#include <VU.h>
external_C proc DoSomething (n)
int n;
{}
int p=3000; //a parameter
{
start_time[“T1"];
DoSomething(p);
stop_time[“T1"];
}
```

Agenda

- Performance Testing
- Workload Generation
- “Record and Playback “
- Custom Test Harness
- Custom Load Generation
- Implementation with External DLL
- *Implementation with Java Language Test Script*

Test Script Services For Java

- External DLL isn't good for other languages
  - To run a custom client written in Java the JVM was started each time with parameters to run the particular report by the Win32 CreateProcess function
- Rational 2001 (and later) TestManager could work with Java, VB and Command Line (Shell) Test Scripts
  - Just write a script in Java instead of VU script
  - Small Java Language Test Script instead of the awkward starting of JVMs by CreateProcess

Sample Java Language Test Script

```
import java.io.*;
import com.rational.test.tss.*;
public class hr
            extends com.rational.test.tss.TestScript
{
  public void testMain(String args[])
  {
    int n=3000;
    myClass mc;
    new hr();
    mc=new myClass();
    TSSMeasure.timerStart("T1");
    mc.DoSomething(n);
    TSSMeasure.timerStop("T1");
  }
}
```

Java Language Test Script

- Almost all functionality of VU Script available
  - Described in details in "Rational Test Script Services for Java" (tssjava.pdf)
- Eight main classes
  - Datapool
  - Logging
  - Measurement
  - Utility
  - Monitor
  - Synchronization
  - Session
  - Advanced

Development Of Java Language Test Scripts

- Nothing is included by default
- Timer start: TSSMeasure.timerStart(“xxx");
- Timer stop: TSSMeasure.timerStop(“xxx");
- Put in the correct place of the state histogram TSSMonitor.runStateSet(xxx);
- Get internal variable TSSMeasure.internalVarGetInt(IV_uid, userNum);
- Test Log contains what you explicitly put into it
  - You could also check output files Repository\TestDatastore\TMS_Builds\Build 1.Build\Default.LogFolder\xxx.log\perfdata\oNNN

Keep In Mind

- Test Script Services for Java are quickly improving
  - It is better to use the latest version
  - Some minor problems could still exist
    - Script selector in the "New Suite" Performance Testing Wizard only show VU language scripts. If you would like to use Java Language Test Scripts, use the "New Suite" Blank Performance Testing Suite option instead of the Wizard.
- Don’t put the Java Language Test Script on agent machines. Do set the complete environment.
  - Don't forget about jar files, properties files, CLASSPATH, etc.

Demo

Summary

- Performance testing is a must today
- To create workload you could use load testing tools or write a test harness yourself
- Mixed approach may give better results: using custom client stubs and Rational TestManager to manage these stubs and analyze results
  - Called "custom load generation" in this presentation
  - Helps to overcome technology limitations of recording
  - Could be implemented with external DLL or Test Script Services (Java, VB, or Command Line Test Scripts)

Questions?

2002

freedom TO CREATE

Rational User Conference 2002

Thank You!

Alexander Podelko
Arno Sokk

Alexander_Podelko@hyperion.com

This presentation will be posted by tomorrow at:
http://www.rational.com/ruc