M-eux Test

Whitepaper

Performance Testing and Monitoring of Mobile Applications

Abstract

The testing of a mobile application does not stop when the application passes all functional tests. Testing the performance of your mobile application is an important step before releasing the application. This document describes practices and tools you can use to test and monitor the performance of your mobile application, to ensure your end-users have the best possible mobile end-user experience.

Part Number MT-PT
Revision 2468, August 2013

Contents

Introduction

Performance testing is an important part of the overall testing of your mobile application. Proper performance testing helps you make sure that:

- The response times experienced by the end-user of your application are within the boundaries of acceptable performance, ensuring customer satisfaction and retention. In this paper, we will refer to this as **UI-based performance testing**;
- The load your mobile application generates on your back-end infrastructure (datacenter or the cloud) is manageable for your back-end infrastructure. In this paper we will refer to this as **load generation** or **stress testing**;
- You can monitor the performance of your application after it has been released and adoption of your application rises. In this paper, we will refer to this as **performance monitoring**;

"M-eux Test extends industry-standard performance management tools"

This documents gives an overview of the approaches and products that are available on the market to implement these features. Various industry-standard tools exist to implement these test, but they are limited to non-mobile applications. M-eux Tests extends these tools and allows you to re-use your exiting processes and procedures to extend your testing capabilities to the mobile device area. In general, M-eux Test extends the products from Hewlett Packard (HP) and Microsoft. The table below gives an overview of the products that you can use, in conjunction with M-eux Test, for the performance testing activities:

Capability	Hewlett Packard	Microsoft
UI-based Performance Testing	Unified Functional Test (UFT)	Visual Studio Test Manager (MTM)
Load Generation	LoadRunner	Visual Studio
Test & Application Lifecycle Management	Application Lifecycle Management (ALM) Quality Center (QC)	Team Foundation Server
Performance Monitoring	Application Performance Management (APM)	Operations Manager (OpsMgr)

This document describes how you can implement UI-based performance testing, load generation and performance monitoring using standard industry tools, leverage M-eux Test.

UI-based performance testing

Measuring how fast the user interface of your mobile application responds to the input from a user is what we will refer to as the **UI-based performance**. Testing of the UI-based performance is **UI-based performance testing**.

Why mobile devices are different

While most of the performance testing techniques found on the PC platform can be re-used in the mobile platform, the mobile platform does offer a wide range of new challenges that you may need to take into account when testing the performance of your application on a mobile device.

An important difference between the PC and mobile platforms is that mobile platforms are **battery-powered**. Batteries in the mobile devices are significantly smaller than those found in laptops so **battery consumption** is an important performance measurement. M-eux Test gives you access to the performance counters of the mobile platforms, allowing you to measure the battery consumption of your application.

Processor and memory constraints are more present on mobile platforms. Mobile devices run on a different processor architecture (ARM vs x86) than PC devices and the processors and less processing power and memory capacity is available. Testing the performance of your application therefore becomes more important on the mobile platform. M-eux Test gives you access to the performance counters of the mobile platforms, allowing you to measure the CPU and memory consumption of your application.

> "With M-eux Test, you can re-use your functional tests as performance tests"

Finally, the **network connectivity** between the mobile devices and your datacenters or the cloud are more constrained than a PC connected to a wired network. The overall bandwidth is lower, roundtrip times are higher, and packet loss is more frequent, just to name a few. M-eux Test allows you to measure the response times of your application on different network types, giving you insight in how well your application will perform on unreliable networks.

Simulating the network conditions between the device and the back-end system

When testing the response times of your UI, it is important to take into account the impact of the network layer. Most mobile devices connect to back-end systems over GSM or WiFi networks which can be slow and unreliable. If you execute the tests close to your datacenter on high bandwidth, low latency networks, you may not see the same performance as your end users.

M-eux Test offers integration with Shunra Network Virtualization for Mobile which allows you to simulate the network conditions of various mobile networks across the globe, allowing you to see what the performance of your application will be in a specific geography even if you do not have access to devices in that geography.

Re-use your existing functional tests

With M-eux Test, you can re-use the existing functional tests you have written and re-use them as performance tests. All you need to do is to add instrumentation code to your test scripts that will measure the response times of your application.

For more information on how to use your functional tests as performance tests, please see the M-eux Test User Guide.

Monitoring the performance of your application over time

Through our integration with HP ALM, HP QC and Microsoft Team Foundation Server, we allow you to store the performance metrics of your application of each individual run. You can then upload these results to HP ALM, HP QC or Microsoft Team Foundation Server. Using these tools, you can track and manage the performance of your application over time.

Load Generation

In many cases, you want to test how your system behaves as the popularity grows and more and more users start connecting to your back-end servers. Load generation allows you to simulate the load coming from these users. You can use the same techniques as used in UI-based performance testing to measure the response times your end users will see. You can use the monitoring infrastructure for your servers to identify potential performance bottlenecks.

You can generate load at two levels: at the protocol level and at the device (User Interface) level. When choosing between these two, you will have to choose between the ease of setup and configuration and the preciseness of your test results.

In general terms, device level load generation will be a more precise match of the load that you will see in production, but protocol-level load generation is, in certain scenarios, easier to set up and configure.

Capability	Protocol-Level Load Generation	Device Level Load Generation
Setting up the required infrastructure	**Low**. Most organizations already have a load generation tool in place.	**Medium to High**. If you want to test with 100's to 1000's of concurrent users, you will need to configure physical or virtual versions of these devices.
Configuring the scripts	**Medium to high**. You will need to generate scripts that simulate the load coming from your application, duplicating existing work.	**Low**. You can re-use your functional test scripts and you can re-use your UI-based performance testing scripts.
Quality of simulated load	**Medium to high**. Application-level specific patterns in the load may not be captured.	**High**. You test the load as it will be generated by an actual application on an actual device.
Details of results	**Medium**. You get the response times at the protocol level but not at the user interface (UI) level.	**High**. You get the response times as experienced by the end-user.

In most scenario's, we see our customers using a mix of both. For example, you can use protocol-level load generation to generate 80% to 99% of the anticipated load, and use device level load generation to generate the remaining load. You can use the load generated on the device to measure the exact response times.

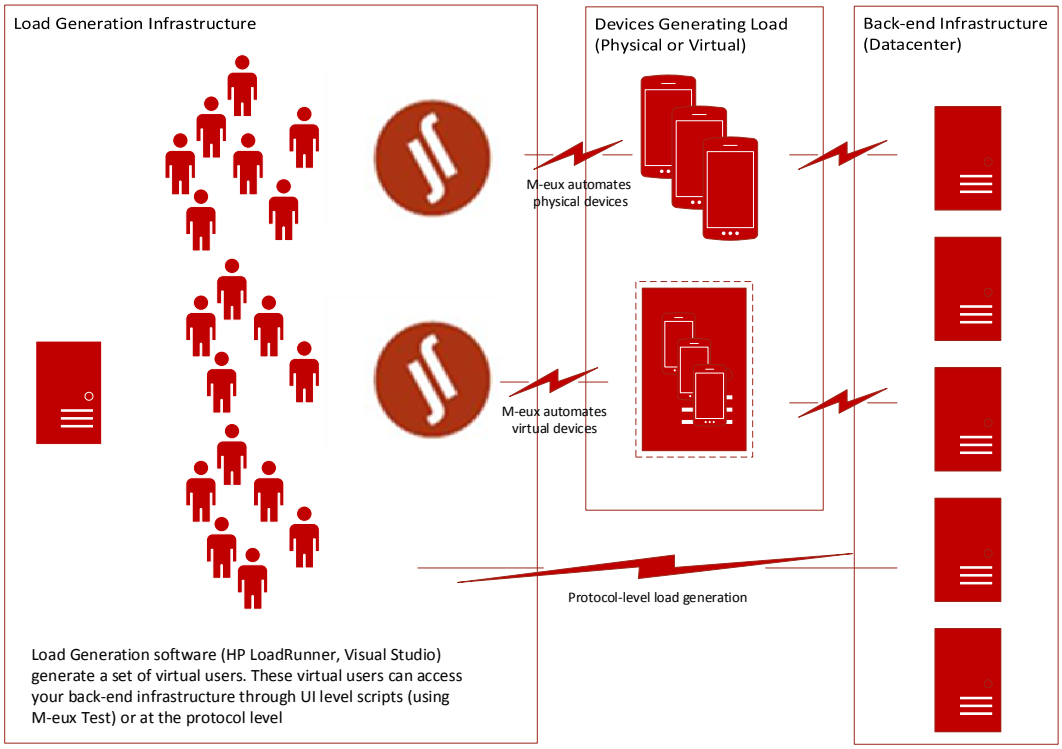
Protocol-Level Load Generation

Protocol-level load generation does not use devices and simulates the traffic at the protocol level. For example, if your application executes a series of HTTP requests against your back-end infrastructure, you can intercept these HTTP requests and script them using your protocol-level load generation tool (such as HP LoadRunner and Microsoft Visual Studio).

For further information on protocol-level load generation, we refer you to the documentation of these tools.

Device (User Interface) level Load Generation

To generate load against your system using devices, you need two principal components: a **device pool** which will be generation the load and a **load generation system** which will use your devices to generate that load.



From a high-level point of view, this setup works as such:

- **Load generation software** (HP LoadRunner or Microsoft Visual Studio) creates a set of virtual users. Due to think times and concurrency rates, only a limited number of these virtual users are active at any time.
- A set of **agents** execute the actions of the active virtual user. These actions are recorded as M-eux Test Scripts in HP UFT or Microsoft Visual Studio.
- Each agent has a **dedicated mobile device** on which it can execute its scripts. The mobile devices connect to your back-end network.

A couple of considerations come into play when using devices to generate load against your back end systems. The most important choice will be the choice between physical and virtual devices.

Choosing between physical and virtual devices

While on the PC platform, you will see most performance tests being executed on virtual devices, the choice between physical and virtual devices has not been settled yet on for the mobile platforms. Using M-eux Test, you can use both physical devices and virtual devices (emulators or simulators) for generating your load.

If you choose to use virtual devices, you will need to choose a **virtualization layer** on which you want to host the emulators for your devices. Popular choices include products include VMWare vSphere, Microsoft Hyper-V and the OpenStack infrastructure.

Managing the performance of your application over time

Through our integration with HP ALM, HP QC and Microsoft Team Foundation Server, we allow you to store the performance metrics of your application of each individual run. You can then upload these results to HP ALM, HP QC or Microsoft Team Foundation Server. Using these tools, you can track and manage the performance of your application over time.

Performance Monitoring

Performance testing of your application does not stop when you release your application. As the popularity of your application increases, your back-end infrastructure may become overloaded and the performance of your application may degrade. Other events, such as reconfiguration of the network or changing user patterns, may also impact the performance of your application.

M-eux Test integrates with the performance monitoring tools from HP and Microsoft, allowing you to monitor the performance of your application using your organization's standard performance monitoring tools.

Through our HP UFT (QTP) and Microsoft Visual Studio integration, we allow you to re-use your performance tests in your monitoring infrastructure. This allows you to set up monitoring of your application in the wild without the need for re-implementing the same test scripts.