ISSN 2320 – 2602
Volume 3, No.10, October 2014
International Journal of Advances in Computer Science and Technology
Available Online at http://warse.org/pdfs/2014/ijacst043102014.pdf

Testing Matrix when Desktop Applications move to Virtualized Environments

Tanvi Dharmarha

Adobe Systems, India, tbajajdh@adobe.com

ABSTRACT

Application virtualization has existed for almost a decade and more and more organizations are setting up their own virtual labs with one or more available virtualization vendors. Choice of vendor depends largely on their current deployment and maintenance setup, time and cost of switch, skill level of administrators and size of enterprise. If a product company states that its desktop products are supported on virtualized environments, it means that the company is ready to offer support on all available vendors, versions, configurations and their various combinations. From a testers standpoint the amount of flavors he has to test a particular product on is vast. Certifying an entire product means testing all product workflows and any third party dependencies involving middleware/runtimes on all possible virtualization technologies and hypervisor combinations.

Apart from product features, a lot of other issues crop up when application move from desktop environment and enter into the virtualized space. Firstly, application is not tuned to support multiple users accessing the same machine and same installation of a product at the same time leading to issues in multiple user access scenarios. Secondly, application licensing may get ugly as product license in tied to the system and not to a particular user. Last but not the least; security might get compromised as access restrictions may not apply correctly in the new setup.

Through this paper we will unveil the major testing parameters that comprise testing in virtualization space and the product under test, scenarios and workflows that get affected when we move towards virtualization and finally how testers can smartly choose a few combinations from an infinite matrix and confirm application certification.

Keywords: Testing Virtualization, Application Virtualization, Product Testing

1. STATUS QUO

In most organizations, testing is still in a phased manner with separate environments for development, unit tests, QA, staging and production. In each phase, testing is focused on specific areas relevant to that phase. Development and unit tests focus on testing classes, functions and components that are being developed and updated. The requisite infrastructure resides on one developer machine. QA stage focuses on testing features and workflows in a production subset environment with minimal infrastructure, most of which is provided in house. Staging phase primarily involves testing subset of functional tests in production like environments. With applications getting more and more complex, with multiple clients, servers, configurations, tokens, templates, the challenge to understand the impact of a system's change increases as the number of pieces, parts, and inter-dependencies increase. With each successive testing phase, it becomes very difficult to maintain configuration compatibility between the pre-production systems and the production system.

Additionally it becomes difficult to simulate a real world environment which includes virtual and non-virtual infrastructure interaction when testing applications. Absence of such hybrid environments in test phases often leads to bugs for identified scenarios later in development cycle thereby increasing development costs and time.

Apart from test infrastructure, most traditional tools used by product QA, whether third party or in house, are not optimized to run in virtual environments.

Additionally, organizations test how their desktop product behaves with different memory settings and make recommendations to the customers about minimum requirements, optimum amounts, etc. But, in a virtualized environment, the end user does not have any control. So, it is becomes difficult for the tester or the virtualization provider to determine what is an optimal amount that will meet the needs of most (or all) of their expected users.

2. DEMYSTIFYING APPLICATION VIRTUALIZATION

At a high level, application virtualization is just redrawing the application on the client just the way it is in its current state on the server (as shown in Figure 1). Every user input of this app's image on the client is passed to the server for processing and the changes are again redrawn on the client. [1]

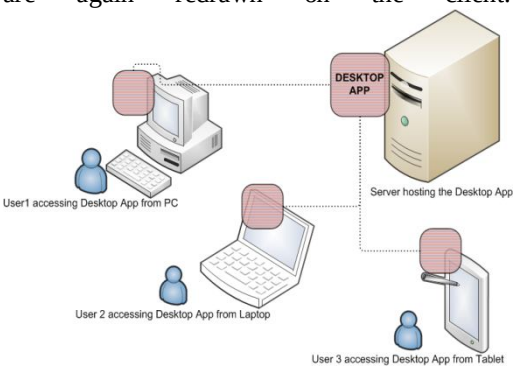

Figure 1: Explaining ApplicationVirtualization

Application virtualization in simplest terms refers to separating the application from the underlying operating system. This separation is usually achieved by a layer called the virtualization layer. Virtualization layer is implemented differently by different vendors and managed by hypervisors that are sometimes vendor specific or third party. With virtualization, organizations are moving to green computing and saving on data center space.

Application virtualization makes an application available to one or more end users by just installing/deploying the application on one server. Virtualization solutions enable on-demand delivery of applications on any device. To enable applications running seamlessly in virtualized environments, product companies need to set up different kinds of virtual labs and deploy the product in every virtual lab setup. The challenge arises when a tester(s) needs to certify the product in all available virtual setups.

3. KEY PARAMETERS

There are many parameters involved in testing of products in virtualized environments. Testing every combination is virtually impossible. Some high level parameters are listed below

1. Product version
2. Middleware version
3. Host machine Operating System and configuration
4. Guest machine Operating System and configuration
5. Host Device
6. Virtualization solution
7. Hypervisor
8. Latency
9. Application Provisioning

3.1 Product version

A full-fledged product encapsulates multiple binaries and libraries each of which is responsible for executing some product workflows. All these binaries and libraries are thoroughly tested by their domain experts/testers and are then versioned, signed and finally released to be integrated into the product. The final integration results in what we call a certified product. Any deviation observed while testing this product in virtualized environment will require a fix, rebuild and recertification of the requisite binary and finally a regression of the product post integration of the new binary.

It becomes extremely difficult to track which version of a product works well on which version of the virtualization setup.

For e.g. consider a product P v1.1 that work well in one virtual setup A but does not work in another setup B. Investigation reports a fix expected in a binary a.dll v1.1 resulting in a new binary a.dll v1.2. Integration of a new binary result in a minor release of product P v1.2, which works on the latter setup B. The problem arises when this new version P v1.2 is released and needs to be retested for regression on the former setup where the previous version was seamlessly running. This typical scenario sometimes leads to a deadlock where fix in one version causes a break in another setup.

3.2 Middleware Version

Some features of a product require specific middleware to be installed on the machine; failing to do so may result in erratic behavior. Middleware is simple terms is a piece of software that interacts with the Operating system and the Product. Common middleware include runtimes, plugins and other application software.

Some desktop applications (or products) come packaged with dependencies and automatically install requisite software during their own installation process while other might expect the middleware to be pre-installed on the machine.

For example, a desktop application may have a specific requirement of Java Runtime v1.6 or

Microsoft .Net framework v3.0 with Service Pack 1. It is very common that the product feature requiring Java Runtime v1.6 will not work as desired on one virtual setup because the setup may only support the latest version of Java Runtime. As a result the product feature would have to be tweaked/ re-implemented to work with the latest run time that is available in the virtualization setup.

3.3 Host machine Operating System and configuration

Although virtualization vendors claim that their virtualization solutions completely isolate the Host Operating system and the guess operating system but sometimes the behavior of host machines can override the guess machines capabilities. Different features of applications running in virtual space can behave differently on different OS flavors. A feature might work on windows OS but may be disabled for Android OS.

For example, Windows and Android tablets both offer zooming capabilities with touch screen. Some applications too offer zooming capabilities when running on touch devices. If the virtualization setup is same and user is accessing the desktop application from a windows tablet and Android tablet, the zooming effect may differ. In one case it may zoom only the document inside the application which means that the application zoom has precedence while in other case it may zoom the entire screen including the application under test which means that OS zoom has precedence.

3.4 Guest Machine Operating System and configuration

Guest machine or virtual machine usually has virtual components such as vCPUs (CPUs assigned to a virtual machine), cores, virtual sockets and virtual drives. Configuring a virtual machine depends largely on the requirements which can be compute intensive, memory intensive, storage or GPU intensive etc. An application deployed on the same operating system can behave differently if the choice of operating system configuration is compute intensive as opposed to memory intensive. Graphic intensive application features might fail to run on low powered machines with a non GPU intensive processor.

Some application software or middleware do not get installed on virtual environments. One such example is the coupon printing software that explicitly prevents installation on virtual environments for obvious reasons. Tweaks in registries are required to bypass the virtual instance detection.

Also, some virtualization vendors do not provide support for some operating systems that might be relevant for the desktop application. Testing the application in unsupported OS flavors also leads to multiple failures. For example, Windows Virtual PC does not officially support versions of windows earlier than Win 7 [2]. Likewise, some video editing applications such as Adobe Premier Pro require a lot of monitoring and Video I/O hardware add ons on different OS(s) that make virtualization of the application trickier than that of other traditional applications.

Guest machine and its device drivers dialog with a virtual device driver instead of the real hardware. Desktop applications may require printers, scanners, other I/O for its workflows and testing the application would thus requires testing interaction of the guess machine device driver with the virtual device drivers.

3.5 Host device

Desktop Applications by default may be expected to run seamlessly on PCs and Laptops but with the advent of on demand application delivery anywhere and on any device, it becomes critical that the desktop applications work on all devices ranging from PCs, laptops, netbooks, tablets and even smart phones. Often a desktop application fails to launch on a tablet or netbook due to unmet screen resolution requirements. For example, desktop applications may require a minimum screen size or resolution of 1024x786 and if the host device does not support that desired resolution, the application will fail to launch.

3.6 Virtualization Solution

There are numerous organizations offering virtualization solutions and the way they implement virtualization is also different. Earlier the major players in the market were Citrix, VMWare and Microsoft but now a lot of small medium enterprises (SMEs) are entering into this niche sector with their virtualization solutions.

A provider can implement the virtualization

Not only application testing but also setting up the test labs for each virtual infrastructure is a great challenge.

Applications read from and write to various file directories and registries and if a virtualization solution imposes some access restrictions to the requisite folders and directories, application will not work correctly. Also, some application solutions allow access to host machine’s hard drive while

others restrict access to only the Virtual Machine's file system.

3.7 Hypervisors

Implementing virtualization requires a layer that decouples the hardware from the operating system. This virtualization layer is managed by what is called a hypervisor. Hypervisor manages resource allocation of all VMs. Hypervisors can be bare metal (which means that they sit on top of the hardware) or hosted (which means that they are installed on the OS that sits on the hardware).

Applications have to be tested on not just both types of hypervisors but also their models/brands available in the market. Some bare metal hypervisors work only on Linux while others work on Win or Mac or both.

3.8 Choice of Hypervisor also determines the virtualization implementation methodology and an enterprise can have any available hypervisor deployed in its virtualization setup so it's critical that the desktop application in question should work on that hypervisor.

3.9 Latency

Latency here has become more of an operational parameter as opposed to an observation. The reason is that when application is available as streamed, the user expects the application's performance to be consistent with what he gets if the application was installed on the local machine.

As infrastructure, application and storage move to the cloud, we do not know at which location our server will be that are hosting the application. A creative professional based out of South East Asia could be accessing a graphic intensive application, say AutoCad that is deployed and published in a N. West US location. This user could simply reject the application if there was too much latency while applying some graphics.

Every application virtualization applies a cap on the number of concurrent users accessing the application, sometimes the number spawns to hundreds. In such scenarios there is an added challenge of simulating load and volume that needs to be tested between the real and virtual components.

Another aspect of latency is benchmarking and measuring workflow turnaround time as these may vary depending on the host and guest hardware, network bandwidth, number of concurrent users and number of redundant servers that host the application. LoginVSI and SPECvirt are standard tools for benchmarking virtual desktop environments [3].

3.10 Application Provisioning

A desktop application is usually not tuned to support multiple users accessing the same machine and same installation of a product at the same time leading to issues in multiple user access scenarios. Users access applications in virtual environment by establishing a terminal session and user specific information are stored as session properties. Tests need to be conducted to ensure that one user does not get another users session if their sessions closes unexpectedly because of network glitches.

Also testing and updating of EULA becomes equally important as some Desktop Application's EULAs may prohibit from deploying the software to cloud server.

4. TESTING MATRIX FOR A DESKTOP APPLICATION

Following table, Table 1, lists the major values possible for each parameter discussed above. An exhaustive list of testable scenarios will include each and every combination from this parameter galaxy.For simplicity let's consider a product/desktop application with 5 major binaries and libraries

Product Binary Version	Middleware Version	Host OS	Guest OS	Host Device	Virtualization Solution	Hypervisor
1. Lib 1	1. Java runtime 1.6	1. Mac 10.8	1. Win 7	1. PCs	1. Citrix XenApp	1. XenServer
2. Lib 2	2. Java runtime 1.7	2. Mac 10.9	2. Win 8	2. Laptops	2. Citrix XenDesktop	2. Hyper V
3. Lib 3	3. Microsoft .NET framework 4.0	3. Linux RHEL	3. Win 8.1	3. Netbooks	3. VMWare ThinApp	3. ESX
4. Lib 4	4. Microsoft .NET framework 3.5 SP2	4. Linux Ubuntu	4. Win R2 2008 Server	4. Tablets	4. VMWare View	4. ESXi
5. Lib 5	5. Python 2.7	5. Linux Fedora		5. Smart Phones	5. AWS Workspace	5. Oracle VM Server
	6. Python 3.3	6. Win 7			6. Google Compute Engine	
	7. Device Drivers	7. Win 8			7. NVidia Grid VCA	
		8. Win 8.1			8. Microsoft App V	

Table 1

With the above parameters list which is non exhaustive, the number of test combinations approximates to two hundred and twenty four thousand (~224000). It is just humanly impossible to test two hundred thousand virtual lab combinations and then execute the identified test cases on each combination.

5. SELECTING COMBINATIONS FROM THE PARAMETER GALAXY

Before we proceed with selecting what all combinations to test, it is important that we evaluate and eliminate a parameter values that do not apply in case of the product under test. For example, a java based application will likely require a java runtime

version rather than a Microsoft .Net framework. Hence we can eliminate all the .Net framework versions from testing. For an application that does not support a particular operating system, we can remove that OS from the guess operating system list.

Product Binary Version	Middleware Version	Host OS	Guest OS	Host Device	Virtualization Solution	Hypervisor
1. Lib 1	1. Java runtime 1.6	1. Mac 10.8	1. Win 7	1. PCs	1. Citrix XenApp	1. XenServer
2. Lib 2	2. Java runtime 1.7	2. Mac 10.9	2. Win 8	2. Laptops	~~2. Citrix XenDesktop~~	2. Hyper V
3. Lib 3	~~3. Microsoft .NET framework 4.0~~	3. Linux RHEL	3. Win 8.1	3. Netbooks	3. VMWare ThinApp	3. ESX
4. Lib 4	~~4. Microsoft .NET framework 3.5 SP2~~	4. Linux Ubuntu	~~4. Win R2 2008 Server~~	4. Tablets	4. VMWare View	4. ESXi
5. Lib 5	5. Python 2.7	5. Linux Fedora		5. Smart Phones	5. AWS Workspace	5. Oracle VM Server
	6. Python 3.3	6. Win 7			~~6. Google Compute Engine~~	
	7. Device Drivers	7. Win 8			7. NVidia Grid VCA	
		8. Win 8.1			8. Microsoft App V	

Elimination should be followed by prioritization. Prioritization is the key to narrowing down parameter values that should and must be tested. For any given parameter, we need to prioritize the most extensively used values in the target space. This can be done through extensive research and surveys at customer's end.

Priority of a parameter will differ from one desktop application to another. So an evaluation for one desktop application cannot be blindly applied to another desktop application which would have to be supported in virtualized space.

For a specific product A, here is the prioritized parameter chart.

Product Binary Version	Middleware Version	Host OS	Guest OS	Host Device	Virtualization Solution
1. Lib 1	~~1. Java runtime 1.6~~	~~1. Mac 10.8~~	1. Win 7	1. PCs	1. Citrix XenApp
~~2. Lib 2~~	2. Java runtime 1.7	2. Mac 10.9	2. Win 8	2. Laptops	~~2. Citrix XenDesktop~~
~~3. Lib 3~~	~~3. Microsoft .NET framework 4.0~~	~~3. Linux RHEL~~	~~3. Win 8.1~~	~~3. Netbooks~~	3. VMWare ThinApp
4. Lib 4	~~4. Microsoft .NET framework 3.5 SP2~~	~~4. Linux Ubuntu~~	~~4. Win R2 2008 Server~~	4. Tablets	~~4. VMWare View~~
~~5. Lib 5~~	~~5. Python 2.7~~	5. Linux Fedora		~~5. Smart Phones~~	~~5. AWS Workspace~~
	6. Python 3.3	6. Win 7			~~6. Google Compute En~~
	~~7. Device Drivers~~	7. Win 8			~~7. NVidia Grid VCA~~
		~~8. Win 8.1~~			8. Microsoft App V

By prioritizing alone, testers can bring down the number of combinations from a few hundred thousand to a few hundreds.

Once we have a prioritized list, we must identify redundancy points. For example, for a workflow that we know has been implemented to behave in a similar fashion on all windows platforms, we can split the test cases equally among all the windows flavors rather than duplicating tests on all flavors. In the above table, there are two windows platforms in Guess OS, so we can assign Win 7 and Win 8 50 redundancy points each. Or if a particular workflow is less frequently used, we could assign less redundancy points to it so that it tested only on few combinations.

6. CONCLUSION AND KEY TAKEAWAYS

In this age of green computing and cost optimization where virtualization is becoming a matter of survival, testing desktop application in virtualization environment is a big challenge.

1. Focus on the above nine key parameters that define the scope of testing in virtualized setups strengthens our testing exercise and reduces chances of failure and rejection.
2. Test tools like LoginVSI and SPECvirt effectively test for performance and scalability of the virtualization solution.
3. Prioritizing parameter values brings down the combination matrix to 0.l%
4. Apart from the key factors, another best practice that engineers can apply to maximize testing rigor and maturity is to start using virtualization in test and pre-production environments during the product life cycle.

REFERENCES

[1]http://www.dabcc.com/documentlibrary/file/application%20virtualization%20smackdown.pdf

[2]http://en.wikipedia.org/wiki/Windows_Virtual_PC

[3]http://www.virtualizationpractice.com/testing-within-the-virtual-environment-22515/

[4] http://www.spec.org/virt_sc2013/

[5] http://www.loginvsi.com/

[6]http://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/delivering-applications-anywhere-anytime-with-maximum-security-and-control-over-data.pdf