BENCHMARKING IN VIRTUAL DESKTOPS FOR END-TO-END PERFORMANCE TRACEABILITY

A Thesis

Presented to

The Faculty of the Graduate School

At the University of Missouri

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Trung Quang Nguyen

Dr. Prasad Calyam, Advisor

DECEMBER 2014

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

BENCHMARKING IN VIRTUAL DESKTOPS FOR END-TO-END PERFORMANCE TRACEABILITY

Presented by Trung Quang Nguyen

A candidate for the degree of

Master of Science

And hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Prasad Calyam

Dr. Trupti Joshi

Dr. Rohit Chadha

Dr. Zhihai He

DEDICATION

Dedicated with love to my Wife Huyen T. Nguyen and my Daughter Sophia Nguyen

ACKNOWLEDGEMENTS

First and foremost, I thank my wife, my daughter for all of the love, support and motivation they have provided through the time I have been here in Mizzou for the pursuit of my Master, and forever after. And thanks to my parents, sister and brother for all of their encouragement; I have made my career path by far here and kept it going on.

I am deeply grateful to Prof. Prasad Calyam, my advisor who has been giving me with great supports on work and other aspects of life since I became a member of VIMAN lab. I always feel that I am a lucky student who has received supervision from him. I have been continually inspired with his ability and dedication in all the extraordinary researches on virtualization and networking. I thank him for all his wise advice and my discussions with him have significantly influenced my research.

I am grateful to the other three professors on my committee, Prof. Trupti Joshi, Prof. Rohit Chadha and Prof. Zhihai He, for their interest in my research. Without any of them, I cannot complete my Master program in Mizzou.

I wish to thank my VIMAN lab colleagues – Ronny, Sam, Duma and Giang for helping me in numerous ways during my research. I specially thank Ronny who supported and helped me in setting the testbed for my research. And anytime I had a trouble with virtual desktop settings I always called him for support.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

There are proven benefits in terms of cost and convenience in delivering thin-client based virtual desktops, versus the use of traditional physical computers for end-user computing purposes. New cloud offerings such as "desktop-as-service" are rapidly being adopted among various communities, ranging from government agencies to research institutes and business entities. However, several challenges remain in evaluating hardware resources and thin-client protocol configurations for delivering virtual desktops with adequate user Quality of Experience (QoE). Suitable performance benchmarking methods and tools are necessary for service providers to optimize resources to provision large-scale virtual desktop requests, and achieve cost efficiency as well as user satisfaction. Moreover, performance benchmarking can serve as a troubleshooting capability to identify bottlenecks or to qualify whether a particular configuration of virtual desktop infrastructure is valid for the supported user application profiles.

In this thesis, we develop a novel benchmarking methodology and toolkit for virtual desktop environments by enhancing earlier works on slow-motion benchmarking and the VDBench toolkit. We focus on automation aspects of benchmarking, and extend the end-to-end performance traceability for common desktop applications such as

Internet Explorer, Media Player and Excel Spreadsheets. Our approach prevents any invasive modification of thin-client systems, and allows emulation of user behavior with realistic workloads. We also address the user interface design issues of managing workflows between the client and server, and to easily instrument and generate comprehensive performance reports for complex environment setups. In a validation study, we deploy the enhanced VDBench toolkit in a real-world virtual desktop testbed that hosts applications that render 3D visualizations of disaster scenarios for scene understanding and situational awareness. Through the benchmarking results, we show how the toolkit provides user QoE assessments involving reliable video events display under different network health conditions and computation resource configurations.

1. INTRODUCTION AND PROBLEM MOTIVATION

1.1. Virtual Desktop Cloud

Cloud computing has become an inevitable trend of computing due to the development of computer hardware including computation and storage capacity and network connectivity.

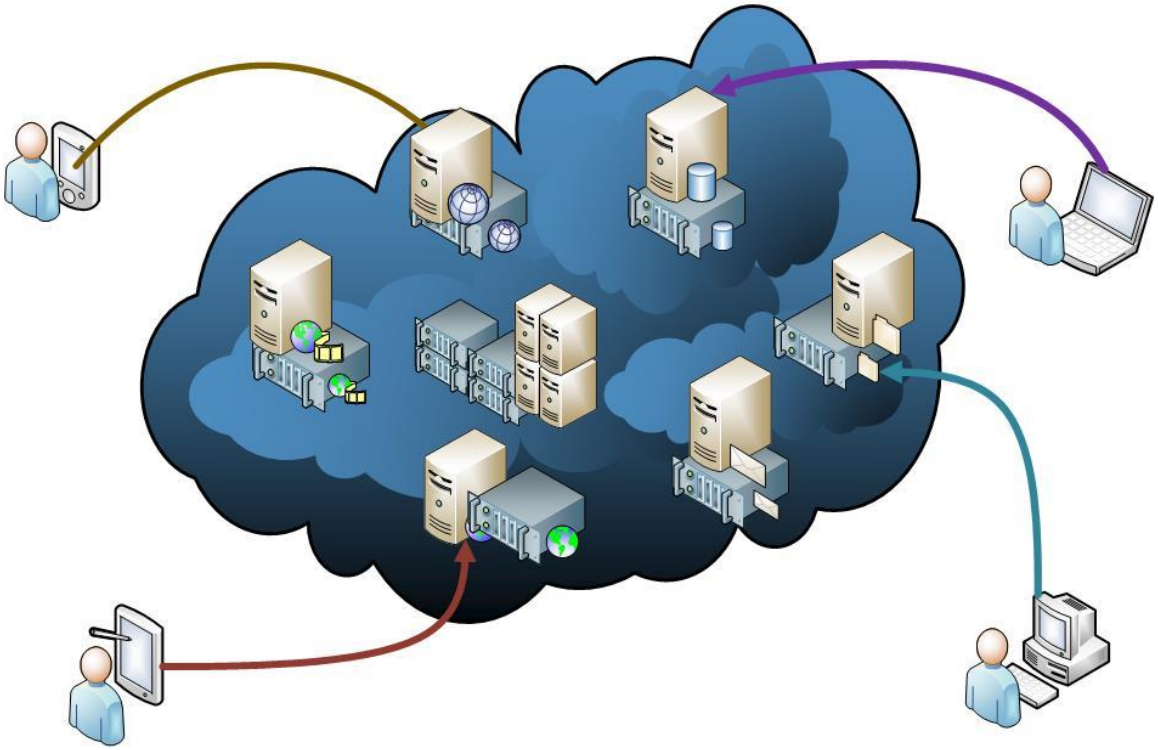


Figure 1: Cloud Computing Platform

The primary goal of cloud computing is to deliver dedicated, custom compute environments to users. The compute environments can range from something as simple

as a virtual machine running productivity software to a cluster of powerful physical servers running complex HPC simulations. In other words, virtual machines or virtual desktops are delivered as a service to customers or we would call it virtual desktop cloud.

The users then remotely connect to the remote compute environment using remote desktop, SSH, or any of the other supported protocols.

1.1.1. Virtual Desktop Cloud System

In order to understand how virtual desktop cloud works, take a closer look into its system and its components.

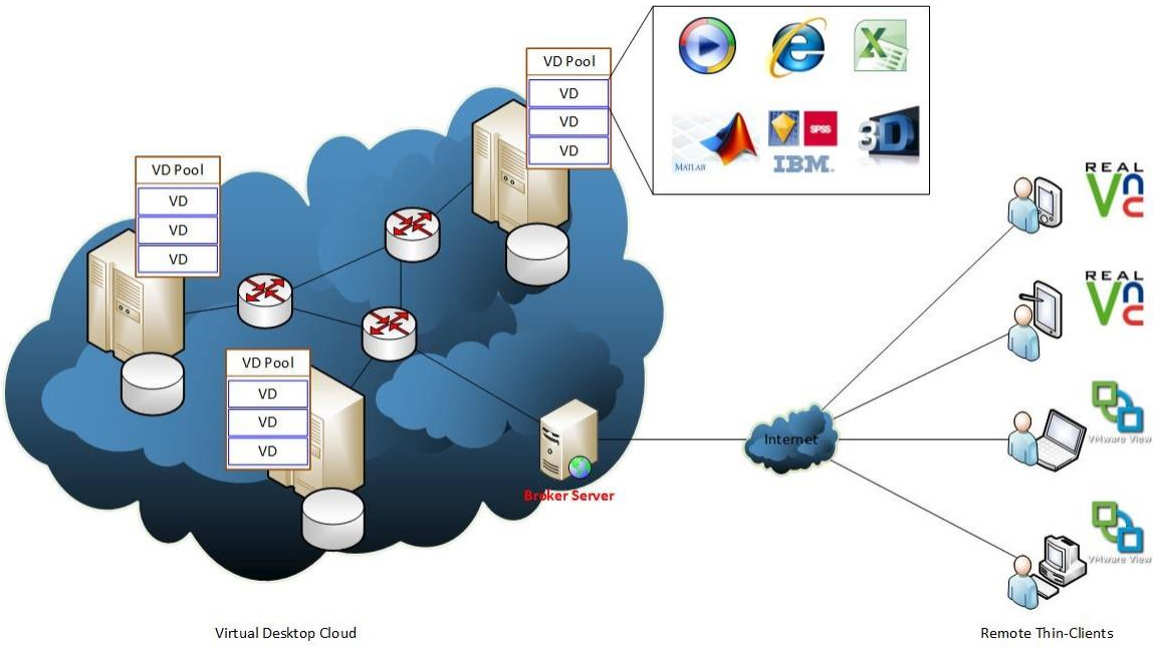


Figure 2: Virtual Desktop Cloud System

In the virtual desktop cloud, there is a number of data centers equipped with rack mounted servers or mainframes. They are located at different places which could be in a city, a country or even in other continents and connected by high-speed network cables.

Each server will run several virtual machines managed by a hypervisor program. In order to use a virtual desktop on the cloud, a remote user first has to connect to a web portal for authentication purpose and through the web service the user is able to allocate a virtual machine or more on the cloud. Subsequently, the user has to use a virtual desktop view client application such as VMware View Client or RealVNC installed on the client to connect to the pre-allocated virtual machine. Once he or she has logged in the virtual machine, a virtual desktop will be displayed and he or she can use popular applications such as Spreadsheet Calculator, Internet Browser, Media Player and Interactive Visualization installed in it.

1.1.2. Characteristics of Virtual Desktop Cloud

Virtual desktop cloud has brought many benefits compared to the traditional desktops, amongst which are mobility, cost efficiency and security. For the first benefit, since virtual desktop cloud is centralized data storage system, users can access to their own virtual desktops on the cloud from everywhere as long as the Internet connection is available. In addition, users can view the virtual desktops from different thin-clients or devices. Using virtual desktops can save costs. For example, with traditional desktops,

users have to buy good configuration hardware, install related software and licensing fees. However, with virtual desktops users can even use a thin-client with acceptable hardware to connect to their own virtual desktops and all needed software and licensing fees have been taken care of by the service providers. Moreover, unlike traditional desktops, maintenance work in virtual desktops does not require a lot of effects. For example, to reinstall or update software, administrators do not need to go one machine to other to do that many times; instead they just have to do it once on the cloud system.

Also, users, especially companies can scale up or down easily based on their needs at a current time. That, in turn, will result in cost efficiency. Finally, virtual desktop cloud is more secure. There is a strong-held belief in traditional IT security that cloud is less secure than traditional systems since users do not have a full control on the data stored on the cloud. However, it turns out that control matters less than accessibility. That means, with the same accessibility, both systems have the same risk of being attacked or encountering secure problems.

Moreover, secure problems with a system do not only come from outside of the organization which owns the system but also from inside, a bad employee, for example. Therefore, with strong security layers around datacenters, the cloud can be protected better from attacks and avoid risks of leaking data from inside since data is stored on the cloud far away.

Contrary to huge benefits which cloud computing possesses as mentioned above, cloud is facing two main challenges including service delivery in a scale manner and Quality of Experience (QoE) assurance. How to scale up or down system resources consisting of CPU and memory resource based on user needs, and how to retrieve idle resources to serve better for others and reallocate again when needed in order to optimize resource allocation and save costs.

Regarding the second challenge, how to ensure that QoE or user-perceived performance is good or at least acceptable for each customer with optimized resources to keep them interested in using services provided by the cloud. On the other hand, the service providers can achieve cost efficiency.

For the first challenge, every cloud platform is able to overcome that challenge as in its nature. With an aim to address the later challenge, benchmarking in virtual desktops is needed.

1.2. Needs for Benchmarking in Virtual Desktops

The main purpose of doing benchmarking is to measure user-perceived performance or QoE in that environment. Based on measurement results, service providers are able to validate and ensure that good QoE will be delivered to their customers. In addition, they also can void overprovisioning issue, allocating more compute resources than the amount customers need without QoE improvement, in order to optimize costs such as compute resources and electricity.

Another benefit of benchmarking is, by using benchmark results, administrators are able to select an appropriate system and connection configuration for a certain user. Last but not least, benchmarking persons can validate whether the current infrastructure is working or not.

1.3. Benchmarking Methodologies and Toolkit

Benchmarking in virtual desktops is essential as discussed in the previous section. However, how to do it or what approach or methodology we will use to make sure that benchmarking results are correct, and validate exactly user-perceive performance in virtual desktops should be considered. After determining what methodology to be used for benchmarking, in order to automate that work which usually requires a lot of steps, efforts and it is time consuming job, a benchmarking toolkit should be taken into account with purpose of saving time and efforts.

1.3.1. Benchmarking Methodologies

Since benchmarking in virtual desktops is very important, some methodologies have been developed in order to measure user-perceived performance. The first methodology would be considered for benchmarking in virtual desktops is conventional benchmarks. The methodology has been used to benchmark for traditional desktop systems. For example, benchmark will report the frame rate as running video playback application in the virtual desktops to measure the performance. However, this approach

can only validate the system performance or server performance in the cloud. It cannot measure or validate the user-perceive performance at clients since frame rate would be changed during display updates delivered to the clients. Based on the network conditions, virtual desktop protocols or mechanisms may discard some display updates or video frames. In other scenario, video frames would be lost during being transferred from server to client. As a result, conventional benchmark does not provide correct measurement.

Another approach is client instrumentation. This methodology will directly instrument the client. By logging input and output display events on the client, very detailed measurements of thin-client performance could be performed. Unfortunately, this approach is not easy to realize since many thin-client systems are quite complex and proprietary. Even if logging is possible, the benchmark results still do not reflect exactly user-perceived performance due to some reasons. The first reason is system modification. Inserting logging mechanism for benchmarking client could alter the client systems and result in incorrect measurement. The second one is inability in determining correlation between frame rates of video in video playback application case, for example, on the cloud system and in the client systems because differences in cloud platforms and remote display or connection protocols. Each connection protocol will have its own mechanism in merging display updates on network degradation.

Network monitoring is another promising approach for benchmarking in virtual desktops. Unlike client instrumentation, this approach does not need to modify client

systems, which would result in more accurate measurement. By using network traces and measuring the latency of operations on a benchmark application, taking place on the cloud system with those operations happened on the clients, and bandwidth consumed during the operations, network monitoring is able to measure more accurately user-perceived performance. However, network monitoring still has some downsides. Since bandwidth consumed during benchmarking is to determine quality of display updates delivered to the clients by comparing it in network degradation and in perfect network condition. Nonetheless, even in perfect network condition, the client system would operate display update merging to keep up with display updates rates generated on the cloud server. As a result, network monitoring cannot quantify the amount of display updates being discarded at the highest available bandwidth or perfect network condition.

The second disadvantage is that network monitoring cannot determine whether the cloud platform all transmit the same overall visual display data. Fortunately, a novel methodology has been concocted to resolve downsides of network monitoring method in order to obtain accurate measurement of user-perceived performance in virtual desktops. That is slow-motion benchmarking methodology which will be described in details in the next section.

1.3.2. Benchmarking toolkit

When an accurate benchmarking methodology is pointed out and it allows measuring exactly user-perceived performance or QoE in virtual desktops, a benchmarking toolkit is necessary. The toolkit enables benchmarking persons to automate their work. That, in turn, helps saving time and effort in doing benchmarking.

Another benefit of having a benchmarking toolkit is that benchmarking can be done in on-the-fly manner. It is flexible to run the toolkit for benchmarking.

In addition, by using benchmarking toolkit, benchmarking persons are able to benchmark performance of popular user applications such as spreadsheet calculator, internet browser, media player and interactive graphic application in virtual desktops.

Also, benchmarking toolkit can benchmark TCP/UDP-based Thin-client Protocols such as RDP, RDS and PCoIP in order to determine performance of each protocol in different network conditions.

Last but not least, using benchmarking toolkit benchmarking persons able to validate remote user experience including interactive response times and perceived video quality under a variety of system load and network health conditions.

2. RELATED WORK

With an aim to do benchmarking in virtual desktops, a good benchmarking methodology which is capable of measuring exactly QoE including interactive response time and perceived video quality plays a vital role. As having been discussed in the previous section, slow-motion benchmarking methodology is the best choice since it provides a novel approach based on network monitoring to accurately validate QoE in virtual desktops. This thesis also investigates VDBench Toolkit – the first attempt to build a benchmarking tool using slow-motion benchmark's philosophy for benchmarking in virtual desktops.

2.1. Slow-motion Benchmarking

Inherited from advantages of network monitoring approach which uses network packet traces to monitor latency and data transferred between client and server, slow-motion benchmarking alters benchmark applications running in virtual desktops in order to resolve downsides or limitations which affect accuracy in QoE measurement. In details, slow-motion benchmarking introduces delays between the separate visual components of that benchmark such as web pages and video frames, so that display updates for each components is full completed on the client before the server begins processing next one.

By introducing delays between the separate visual components, slow-motion benchmarking ensures that no merging or discarding frames or visual components happens. That, in turn, helps qualify the amount of display updates discarded at the highest available bandwidths. Because in slow-motion benchmarking, percentage of display updates being discarded due to merging technique in cloud platforms is relatively low, by comparing data transferred in slow-motion and conventional cases, the amount of display updates discarded can be measured fairly correctly.

Additionally, with long enough delays inserted into visual components, every single visual component will be transferred totally to the client without merging or discarding, that will ensure a better measurement for QoE since the same overall visual display data on the server received by the client.

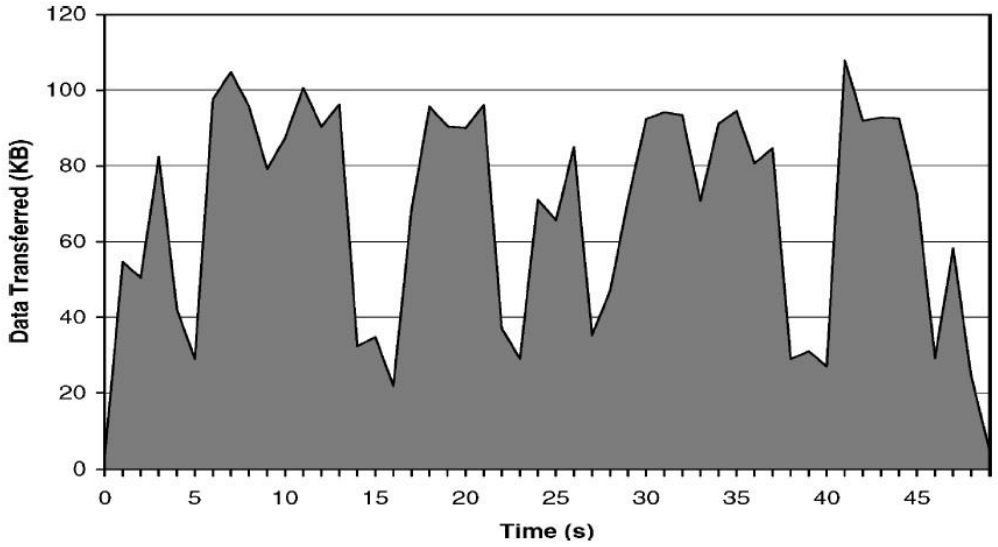


Figure 3: Data transferred during a sequence of Web page downloads with no delays at 100 Mbps.

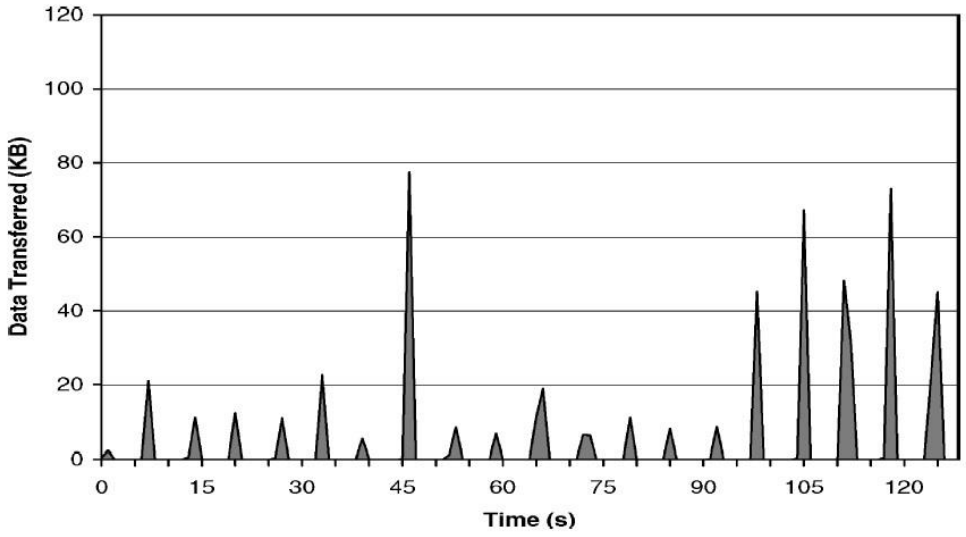


Figure 4: Data transferred during a slow-motion version of the same Web

2.2. VDBench Toolkit

VDBench Toolkit is the first ever attempt to build an automate tool for benchmarking in virtual desktop environments. Based on philosophy of slow-motion benchmarking, the tool has set up independent or discrete visual components such as web text page, image page for the Internet browser, and also for other benchmarking applications in order to achieve accurate measurement of QoE.

2.2.1. VDBench's Benchmarking Methodology

To automate benchmarking work in virtual desktops, VDBench has to simulate user operations or mimic what users do in virtual desktops. In addition, based on Slow-motion benchmarking's philosophy which is insertion of delays between separate visual components in order to ensure that those visual components are fully displayed on the

thin-client without any data lost. Therefore, the methodology used by VDBench involves following steps:

1. Creating realistic workflows in order to generate synthetic system loads.
2. Running those synthetic system loads in network health impairments that effect user-perceived 'interactive response times' (e.g. application launch time, web-page download time)
3. Allowing correlation of thin-client user events with server-side resource performance events by virtue of 'marker packets'.
4. The 'marker packets' particularly help in the analysis of network traces to measure and compare thin-client protocols in terms of transmission times, bandwidth utilization and video quality.
5. Generating resource (CPU, memory, network bandwidth) utilization profiles of different user applications and user groups.

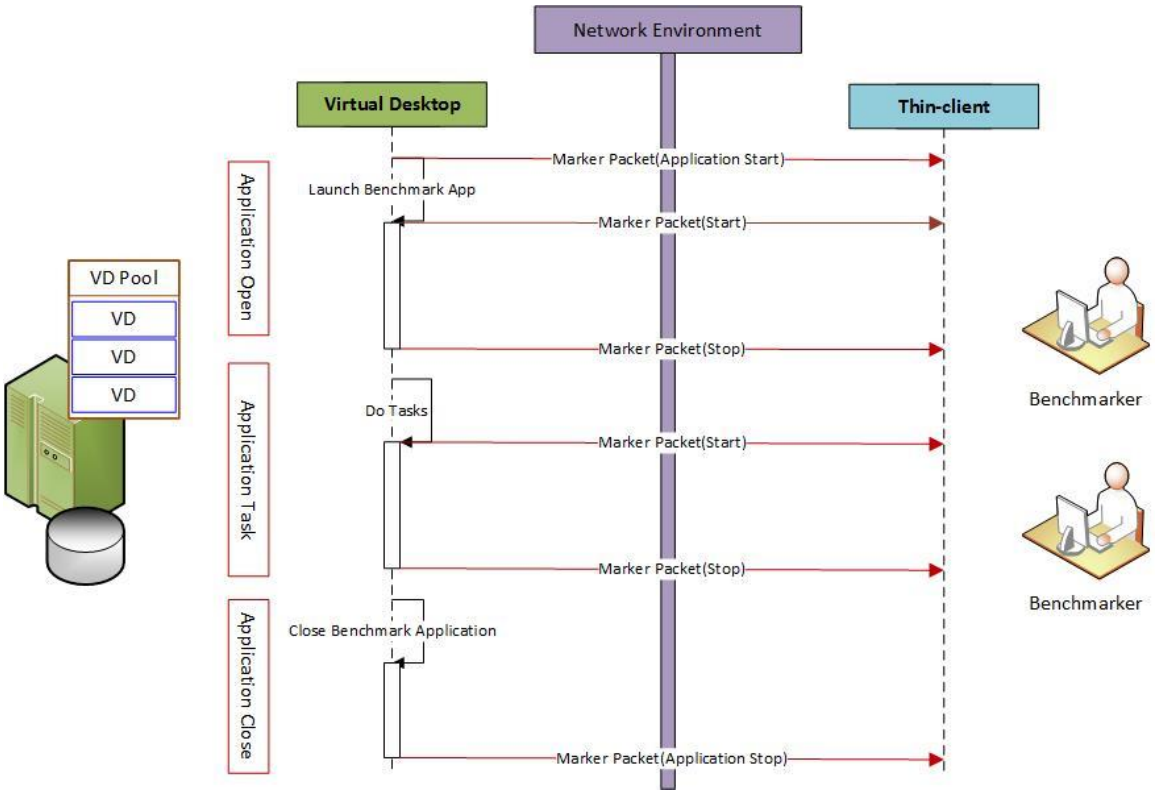


Figure 5: Maker packets inserted into each task or visual component during benchmarking

Based on that methodology, the ultimate purpose of VDBench Toolkit is to be benchmarking the performance of:

- (a) Popular user applications (Spreadsheet Calculator, Internet Browser, Media Player, Interactive Visualization),
- (b) TCP/UDP based thin client protocols (RDP, RGS, PCoIP),
- (c) Remote user experience (interactive response times, perceived video quality), under a variety of system load and network health conditions.

2.2.2. VDBench System Components

VDBench Toolkit is built with Java technology with an aim to run cross-platforms. For example, it is able to run in Windows or Linux-alike systems.

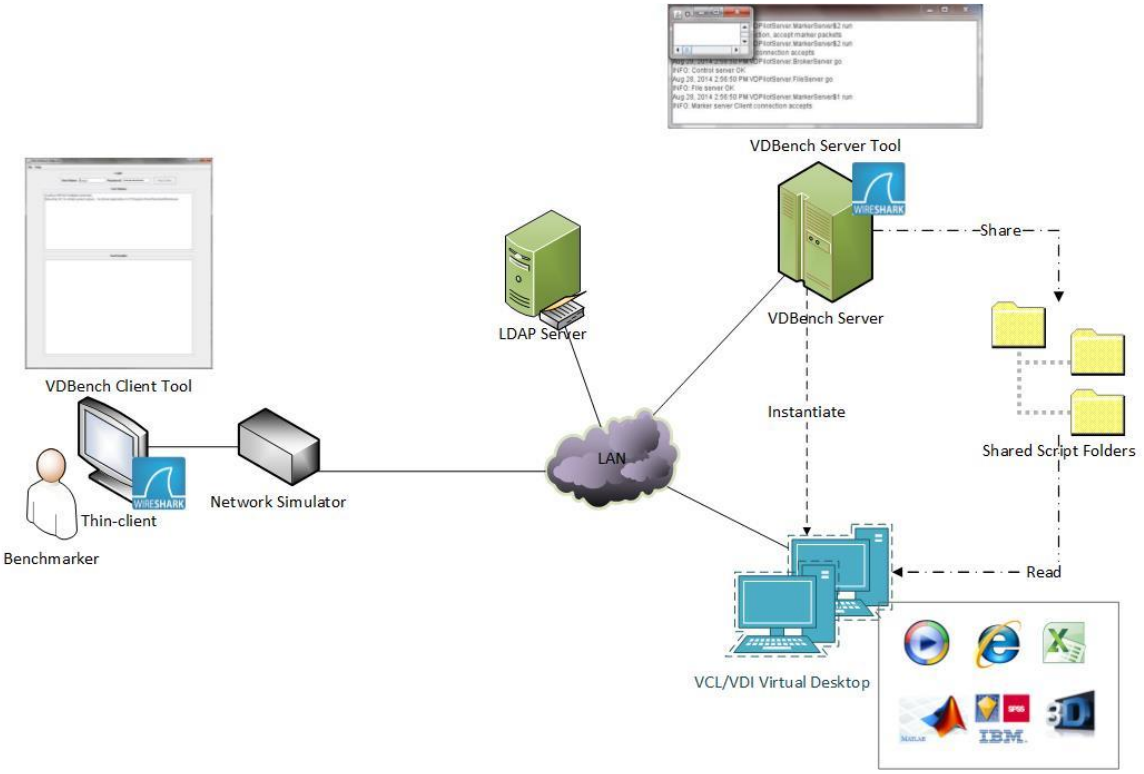


Figure 6: VDBench System Components

Before running VDBench Toolkit for benchmarking, first a benchmarking testbed has been set up or designed. For research purpose, a testbed is composed of a server running a number of virtual machine instances, a LDAP server for authenticating users, a thin-client for connecting to the benchmark server and a network simulator for

modifying network heath conditions. All components are connected to each other via a local area network as showed in the above diagram.

VDBench Toolkit consists of two main components including VDBench Client Tool and VDBench Server Tool. VDBench Client Tool will run on a Thin-client to connect to the VDBench Server for receiving benchmark results. On the other hand, VDBench Server Tool will be running on the VDBench Server to wait for requests from clients, and then start benchmarking work on a pre-configured virtual desktop. Since VDBench Toolkit is designed to do benchmarking for virtual desktops connected from Thin-clients which are with limitations in compute resources such as CPU and memory, VDBench Server Tool running on the server will be in charge of processing data or generating benchmark results before sending them back to the client. On the other hand, the client only has to run a network monitoring tool to track network packets.

Moreover, every time a request from the client is received by the server tool, it will run a remote command on the pre-configured virtual machine. Subsequently, the command will launch benchmark applications based on a pre-configured profile. Those applications will be running as scripted by script files shared over the local network.

2.2.3. VDBench's Implementation

In this section, many aspects of VDBench implementation will be discussed and analyzed. They are user interface (UI), benchmark application scripting, output design,

error tracking, supporting applications used accompany with the toolkit, and last but not least, workflow or work steps to generate benchmark results out of testbed settings.

User Interface

For benchmark persons, VDBench toolkit provides them a GUI built by Java Swing and AWT components.

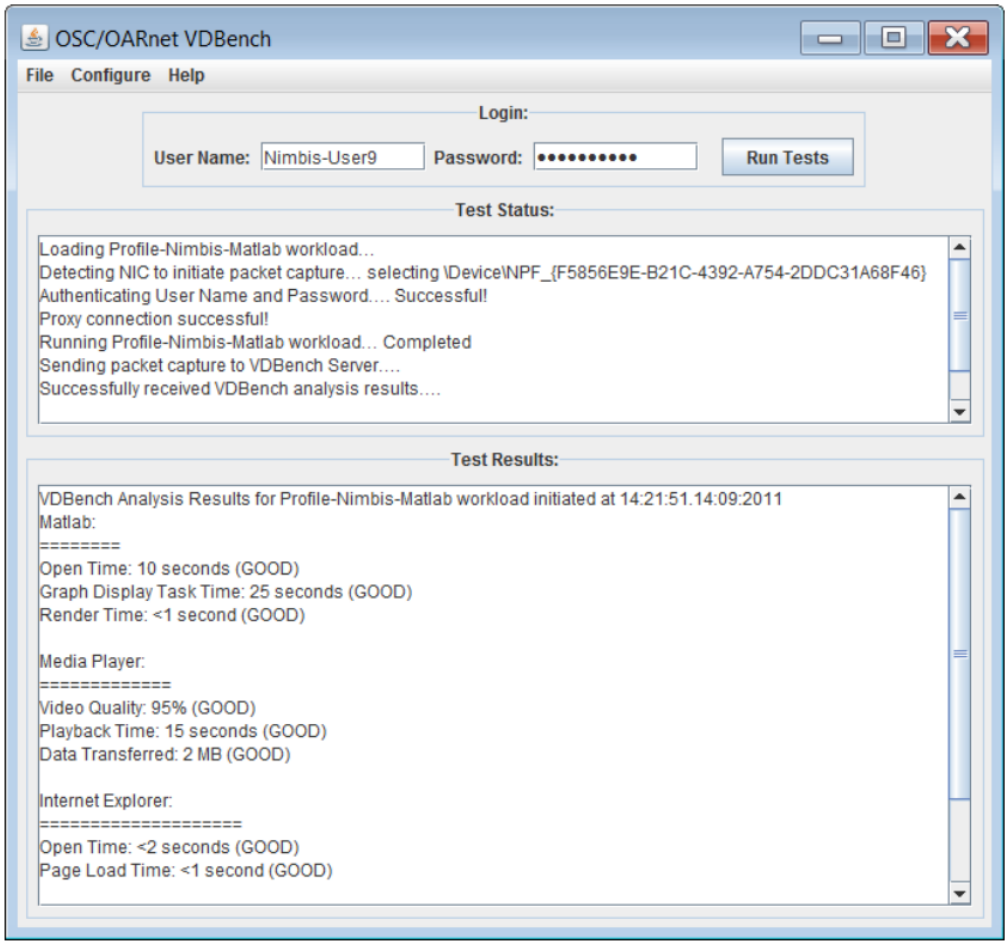


Figure 7: GUI of VDBench Client Tool

For administrators who run VDBench Server tool, a simple log window has been displaying during the tool is running.

Figure 8: VDBench Server Tool

Scripting in VDBench

In order to generate system workloads and simulate user operations or user behaviors in virtual desktops, the toolkit uses AutoIt – a Basic-like scripting language for automating Windows GUI events. By using AutoIt and data files for specific benchmark applications such as Matlab, Excel and SPSS IBM, created system loads are very close to real user behaviors. That allows achieving a better benchmark results or better measurement of user-perceived performance.

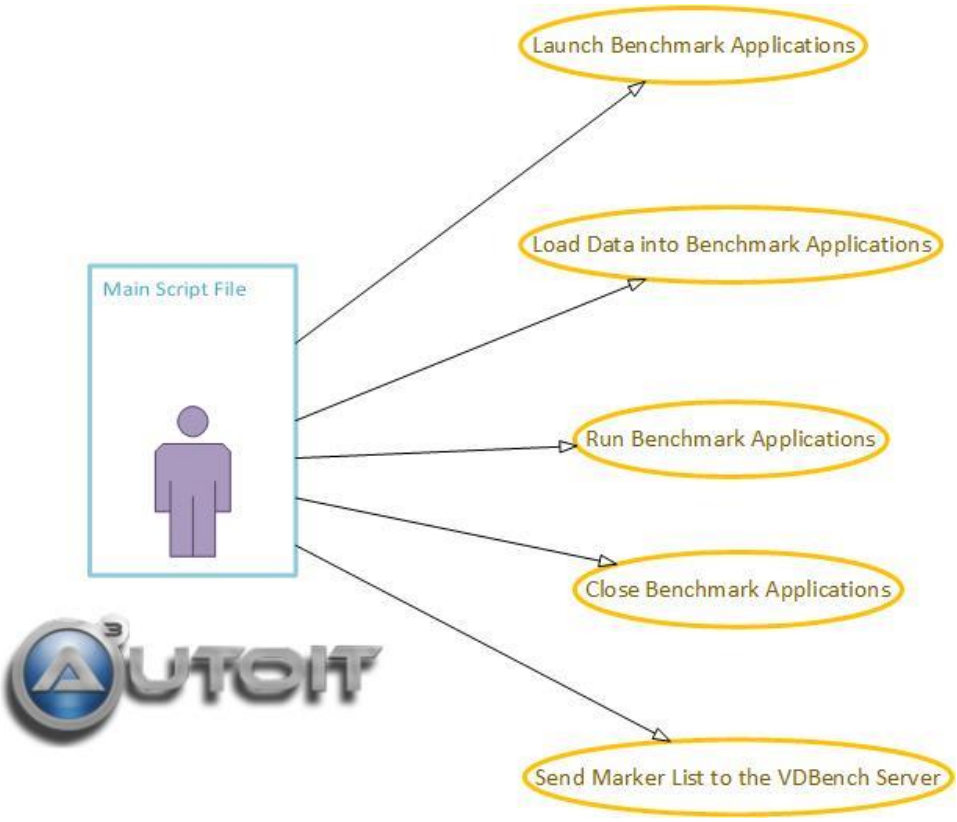


Figure 9: Tasks done by the main script file built with AutoIt3

VDBench Server tool will revoke a remote command on the pre-configured virtual desktop to run the main script file which is response for launching, loading data into, using, and closing benchmark applications, and sending marker list to the VDBench Server.

Output of VDBench

Benchmark results will be only displayed on GUI of VDBench Client tool. And some log information will be showed in the log window of VDBench Server.

Error Tracking Design

Since VDBench Toolkit is a client/server application, it has a high potential to get crashed or encounter unexpected errors during benchmarking. However, the tool just provides log on screen with inadequate information and no history tracking. That will take users long time to fix bugs or to understand what has happened.

Supporting applications

VDBench is a complex system, thus, going along with it other supporting applications including Wireshark – a network monitoring tool, PSTool – a tool for executing remote command on other machines, AutoIt – a scripting language for automating Windows GUI events, VMware View Client to connect to a virtual desktop and MySQL database for storing log information.

VDBench's Workflow

In order to understand how VDBench toolkit generates benchmark results out of the testbed settings, take a look at VDBench's workflow described as below.

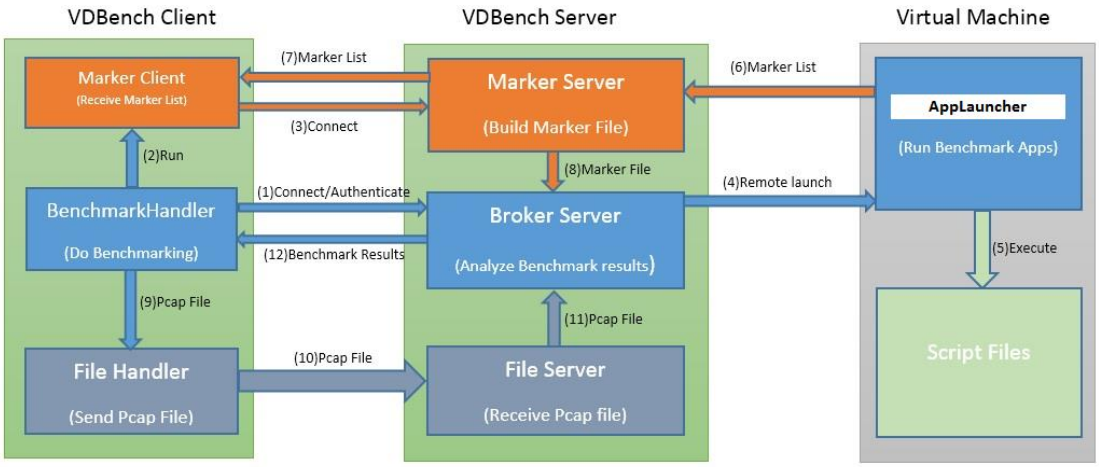


Figure 10: Workflow for generating benchmark results

There are three main modules running on separate threads in VDBench Client tool, including BenchmarkHandler, MarkerClient and FileHandler. BenchmarkHandler module is responsible for doing benchmarking such as connecting to the VDBench server for authentication purpose and requesting to start benchmarking. Also, it spawns an instance of MarkerClient module on a different thread. MarkerClient module will do a loop to wait for marker packets transferred from the server. FileHandler is in charge of sending pcap files, obtained from network monitoring tool – Wireshark, from the client to the server.

For VDBench Server tool, three main modules including BrokerServer, MarkerServer and FileServer will be taken into account. Those modules will run on separate threads. The primary functions of BrokerServer consist of waiting requests from a client, then executing remote commands for doing benchmarking on the pre-configured virtual desktop, processing benchmark results based on marker list and pcap file and sending those results back to the client. MarkerServer module is responsible for receiving Marker list from the virtual desktop and sending it back to the client in order to correlate thin-client user events with server-side resource performance events for benchmarking aim. The last module FileServer is accountable to receiving pcap files transferred from clients, and those pcap files will be used with Marker list to calculate the final benchmark results.

The last object comes to play in the workflow is Virtual Desktop. In Virtual Desktop, the main script file, AppLaunch.exe as mentioned in the previous sections will be executed remotely from the VDBench server. Once the main script file has run, benchmark applications will be launched and run as scripted. At each task of a benchmark application, a marker will be sent to VDBench server for building Marker list file which is used for benchmarking purpose.

2.2.4. Limitations

Since VDBench is the first attempt to do benchmarking in virtual desktops, it cannot avoid limitations which prevent users from deploying and using it easily as

named a few. This section will take a deep look into VDBench toolkit with an aim to draw out all its limitations. Based on limitations, refining and improving is essential to help users do benchmarking at least effort and benchmark results could show more meaningful or comprehensive.

The first and biggest limitation is its adaptivity. It is difficult for users to deploy the tool on other network systems such as VIMAN lab's network. That is caused by system or environment configurations or settings being fixed or embedded into source code. Every time users want to change them due to new system, they have to modify it in the source code and recompile it for running it again. From my own experience, it took me a couple of months to deploy the toolkit in our own testbed in VIMAN lab.

Another limitation or reason that caused users so long time to deploy the toolkit on other network is documentation. In other words, due to lacking of related documents such as a detail manual to guide users in deploying and setting the tool.

Next limitation is about maintainability or scalability. Since there is a limitation in design of the system such as not following module-orientation approach in java and scripting code, it is difficult for adding new functions or modules for scripting a new benchmark application for an example, to the tool or system.

The fourth limitation is traceability. In this scope, the output of the toolkit is so simple and it does not help much in fixing bugs during benchmarking and in collecting data after benchmarking. Collecting data plays a very important role in benchmarking, thus, the tool should have provided better output methods to leverage that work.

Last but not least, the tool has a limitation in term of comprehensibility. That means the benchmark results should be more meaningful or comprehensive to benchmark persons in order to have a better validation on a certain system.

3. CONTRIBUTIONS

In this section, I will describe my contributions for making benchmark work easier, more reliable and more comprehensive. In details, I will refine the VDBench system including adding a new function set for VDBench Toolkit and redesigning scripting in order to achieve adaptivity, scalability, traceability and comprehensivity.

3.1. Deployment Issues and Solutions

In order to do benchmarking in our lab network for research purposes, I was trying to set up a testbed and using VDBench toolkit. However, during deploying the toolkit on our testbed, I encountered many issues as mentioned in previous section, which resulted in a couple of months to figure out. That is serious problem for a toolkit to use in reality.

3.1.1. Deployment Issues

There are a number of reasons that took me so long time to get it run on our testbed. The first reason is that many environment configurations are fixed in the source code. Therefore, I had to update the source code, rebuilt it when I deployed on our new network. Another reason is that I did not have a manual to walk me through steps for configuring and running the toolkit. Additionally, the toolkit does not provide adequate log information to help users like me fix bugs or errors as I encountered.

There are also objective reasons such as complexity of the VDBench system and nature of Client/Server application. VDBench system includes many supporting tools as mentioned in the previous sections. Any of them fails or work improperly, it will cause the entire system fail. Also, VDBench system has a workflow with multiple steps to generate benchmark results. If any of those steps fail, it will result in failure or interruption in the system. In addition, VDBench system is consisting of several environments or components including client, server and virtual desktops. It requires more effort to handle all of them.

Last but not least, VDBench is a client/server application which parses data in the marker list transferred via the network. Consequently, it has a higher potential in causing bugs or errors.

3.1.2. Solutions

I have proposed solutions addressing issues or limitations I had during deploying and running VDBench toolkit, and realized those solutions. As a result, deployment and execution of VDBench system for benchmarking and collecting results much easier.

For the first solution, User Interface (UI) of VDBench Toolkit will be redesigned with an aim to allow users setting system configurations right on GUIs. They do not need to get their hands dirty on modifying and recompiling source code of the toolkit. Also, with new GUIs for both client and server side, benchmark persons and administrators are able to track down output and errors happening right on GUIs.

Second solution is redesigning output to allow taking less effort in collecting or reporting data. In details, log information will be displayed right on GUIs. Also, benchmark results will be saved to a csv file for reporting purpose.

Third solution is to deal with log information and error tracking. In details, log information will be saved to log files for later usage or reference.

Fourth solution is adding a new function set in which allows users to set number of times to loop running benchmark applications, then calculate average results for more accurate benchmarking.

The last solution is documenting the toolkit by creating a detail manual for guiding users in setting, running and maintaining the toolkit.

3.2. Software Enhancement Models

Scripting is a very important part in the VDBench system. The main goal of scripting is to simulate user behaviors during using virtual desktops. It helps automate benchmarking in those virtual desktops. In the current system, there is only one main script file which will handle all of benchmark applications. In case one or more benchmark applications are added to the system, a lot of efforts is needed to modify and add new code in the big main script file. In order to make VDBench Toolkit scalable, in other words, to make it maintainable, I have proposed, redesigned and implemented two enhancement models with module-oriented approach.

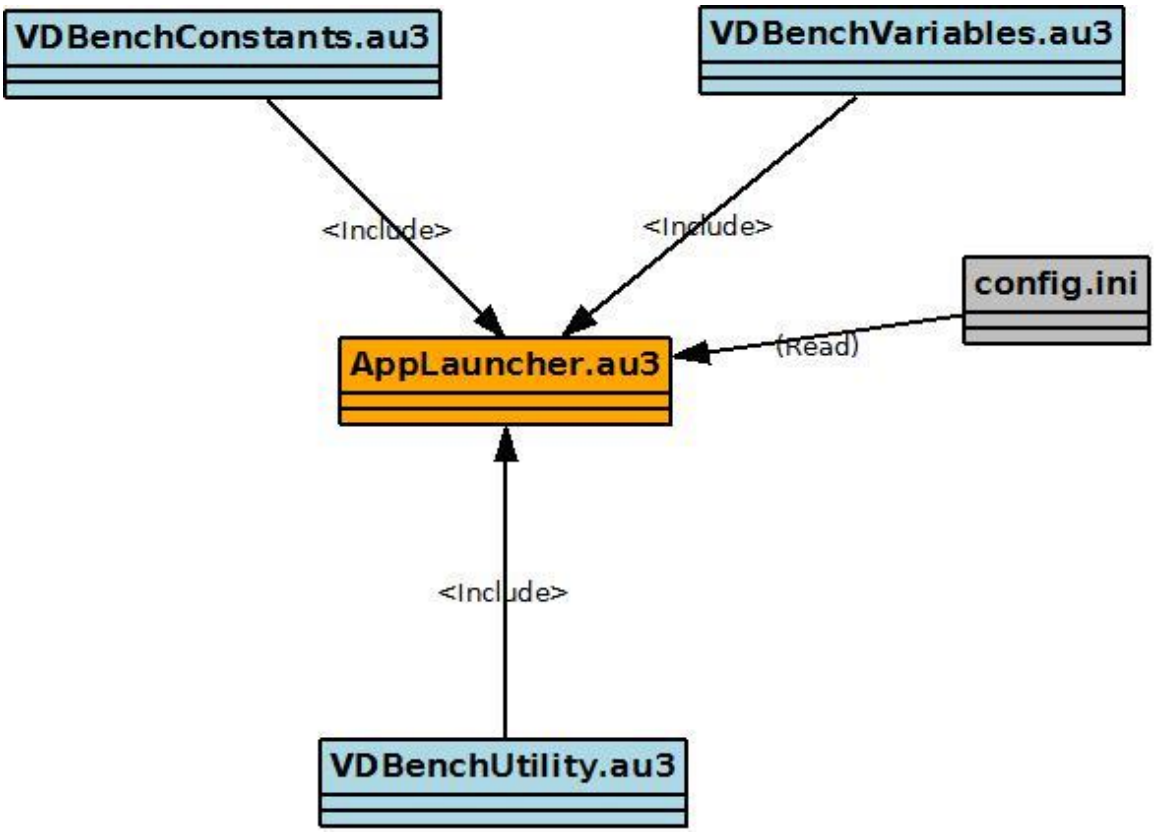


Figure 11: Scripting Model 1

In this model, I have separated constants and variables used in the main script file into different script files, and created a utility file which contains all commonly-used functions. That will help maintenance work easier. Moreover, with this model configuration information will be read from config.ini file rather than be fixed in the main script file. Every time administrators want to update environment settings or configurations related to benchmarking, they can do it with much less effort now. This model seems promising.

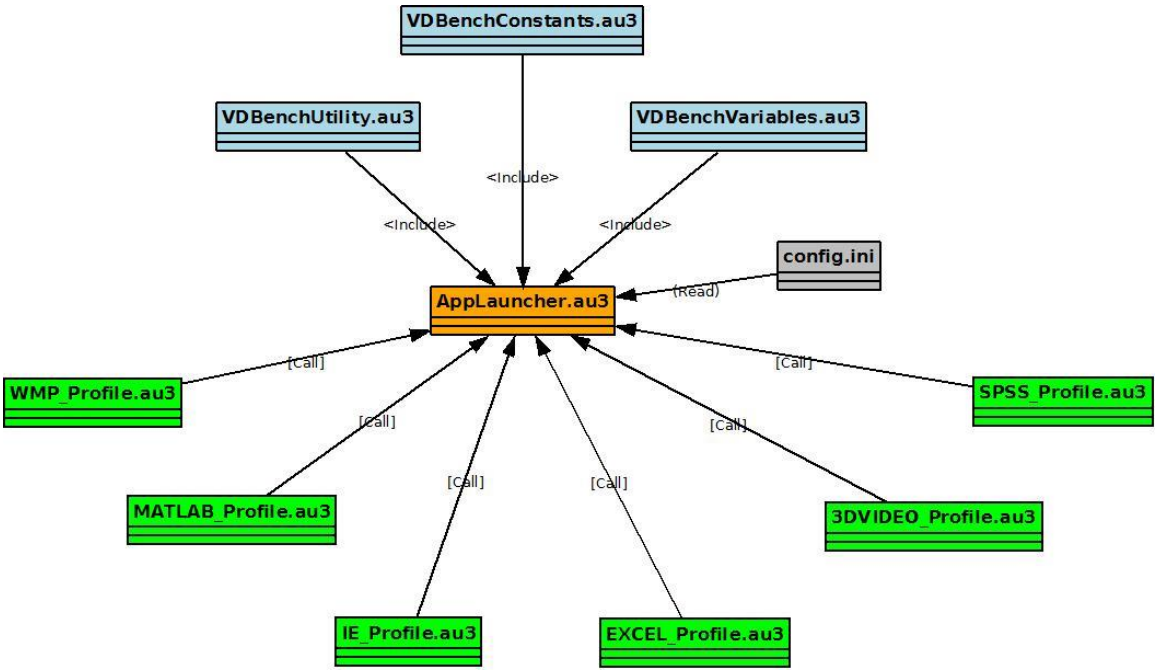


Figure 12: Scripting Model 2

Unlike the first model all benchmark applications will be executed in DVBenchUtility script file, each benchmark application in the model 2 will be taken care of by a script file. That will make maintenance work clearer and easier. In details, if a new application is added to the VDBench system for benchmarking, then a new script file x_Profile.au3 will be created to handle that application. However, compared to model 1, model 2 requires more files to manage while it still has to maintain VDBenchUtility module to contain common functions used in script profile files.

Both models have advantages compared to the old version. Since the main script file is divided into small files with particular functions such as constants, variables and

common functions container. That will help maintenance work easier as in object-oriented programming languages. More importantly, both models use config.ini file to store configuration information. By doing that, system administrators can update system settings on the fly. They do not need to modify source code and recompile it.

3.3. GUI Redesigning and Implementation

The biggest downside or deadlock of VDBench system is that all system environment settings are fix into the source code. That causes the toolkit unadaptable to new network systems or testbeds. In order to resolve the problem, I have redesigned and implemented GUIs for both client and server side of the toolkit.

3.3.1. VDBench Client Tool

VDBench Client Tool will be run on a Java-supporting thin-client. The thin-client would have Windows or Linux installed on.

For testing purpose, VDBench Client Tool can be run on any kinds of computers that support Java such as Laptops, MacBook, and Desktops. In addition, VDBench Client Tool must run along with Remote Desktop Connections-alike software such as VMware Horizon Client View. By using that software, testers can view the Virtual Desktop which runs benchmark applications and follow instructions popped up on it.

“Connection” tab

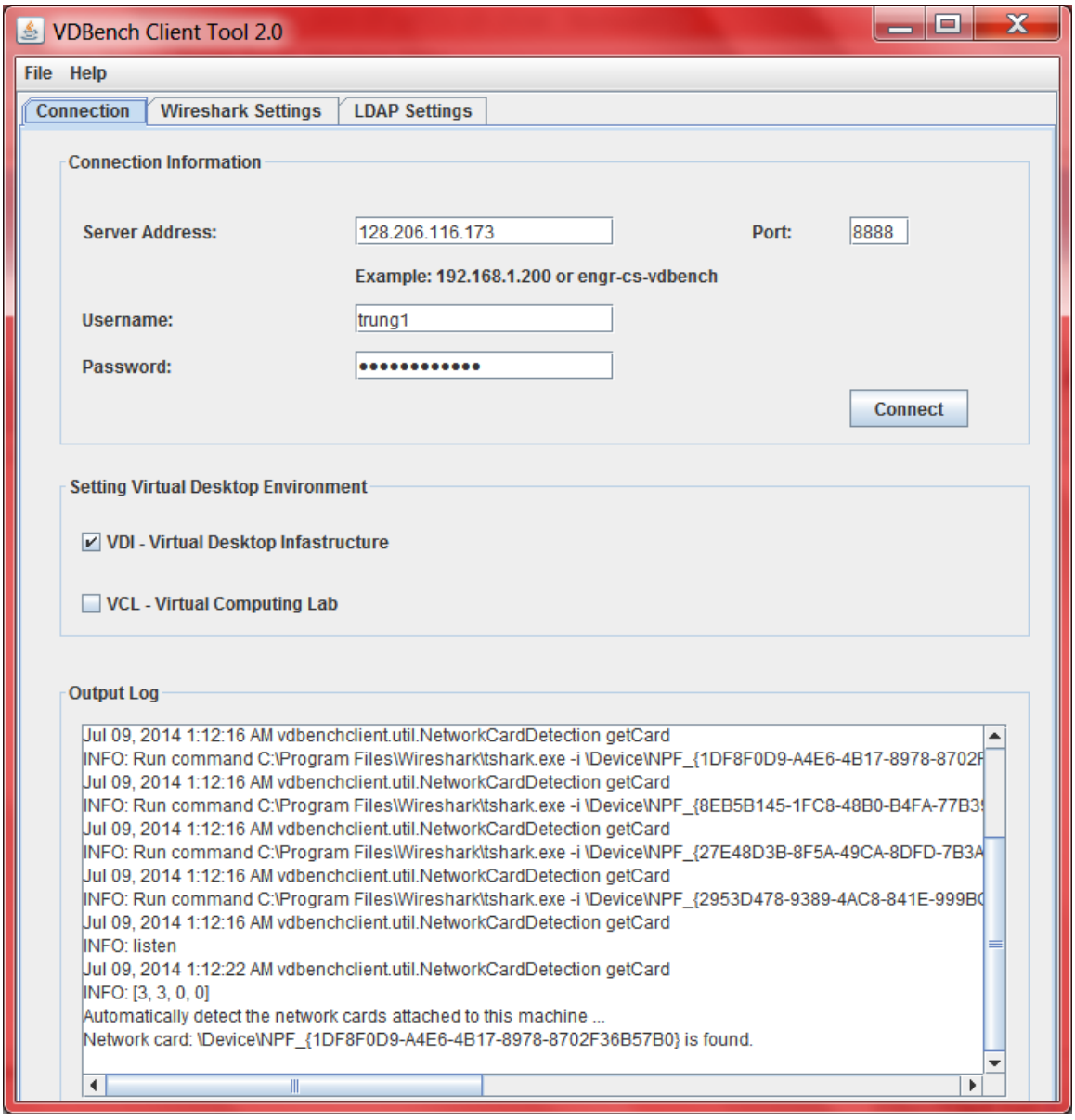


Figure 13: “Connection” tab of VDBench Client Tool Version 2.0

On the GUI of VDBench Client, there are three setting tabs. The first tab named “Connection” will be in charge of setting connection information including Server IP

address or domain name and Port number by which the client will connect to the Server.

Besides, users also need to input Username and Password to authenticate on the Connection Broker or Active Directory, which is described on the below figure. User authentication will be discussed further on “LDAP Settings” tab.

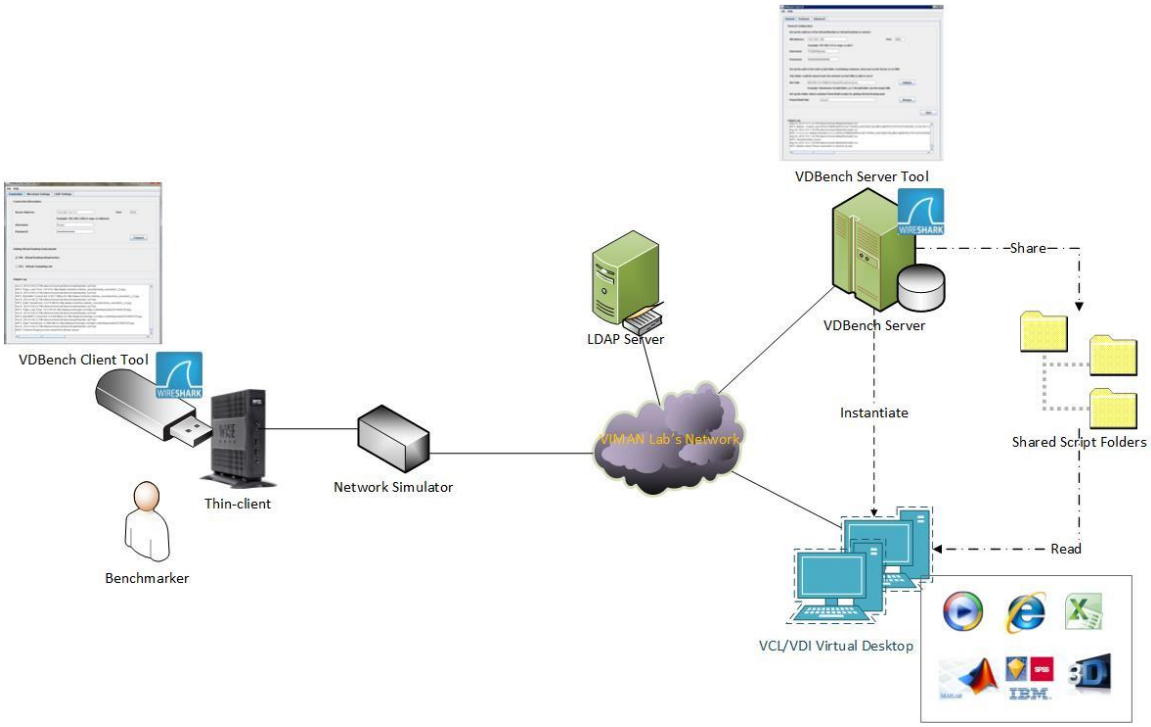


Figure 14: Virtual Desktop Cloud System Components

For benchmarking a Virtual Desktop Cloud, there are two main types of it including Virtual Desktop Infrastructure introduced by Windows and supports Windows Systems, and Virtual Computing Lab introduced by Apache Software Foundation, supports Windows and Linux Systems. By choosing the Virtual Desktop environment,

the Client will let the Server know how to deal with benchmarking requests from the Client.

On the “Connection” tab, a Log window will print out all log information while doing benchmark to let users know what’s going on.

After configuring users’ own environment or using the default configuration, press “Connect” button to start connecting to the Server.

Interestingly, after launching VDBench Client Tool, it will start looking for network card attached to the current Thin-client or machine by using Wireshark. Thus, make sure that Wireshark is already installed on the Thin-client or setting file path to “tshark.exe” program is right before starting connection to the Server.

“Wireshark Settings” tab

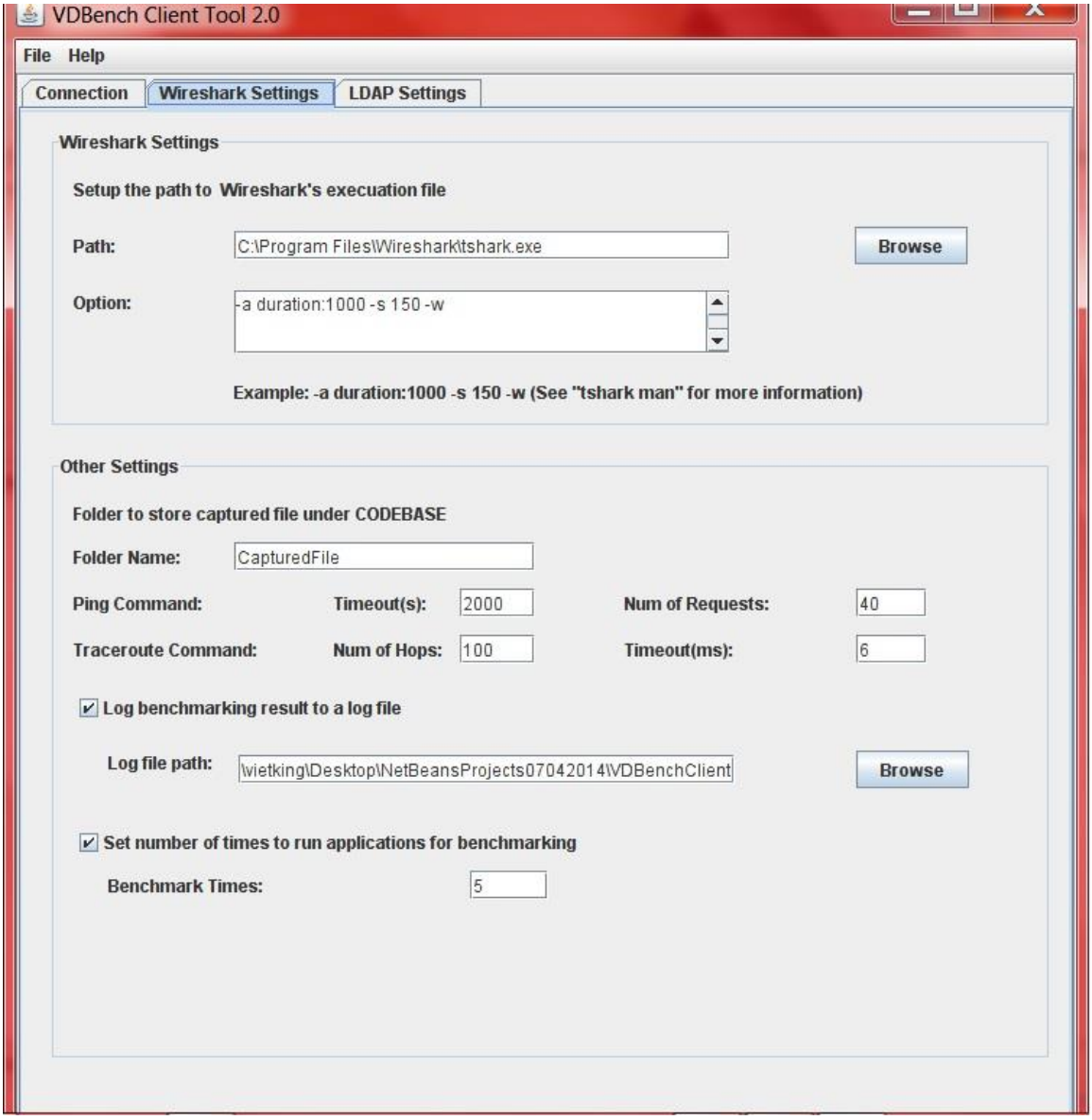


Figure 15: "Wireshark Settings" tab of VDBench Client Tool

On the “Wireshark Settings” tab, users are able to set file path to “tshark.exe” program that will be responsible for tracking network packs sent from the Server. By default, the file path to “tshark.exe” program is under “Program Files” folder, however if

VDBench Client Tool cannot detect the network card attached to the machine, that means file path to Wireshark is incorrect or Wireshark has not been installed yet.

Users would set parameters for the “tshark” command such as duration of capturing network packets. However, by default, setting parameters will be taken care of by the network administrator who runs VDBench Server Tool on the Server. That setting information will be sent to the Client and then set for “tshark” command. If for any reason, the Client receives no setting information, it will use the parameters set by the users or default values.

Besides, other information such as name of folder storing pcap files generated by “tshark” command, parameters for Ping command consisting of timeout in seconds and number of requests made by the command, and parameters, including number of hops and timeout in seconds for “tracerout” command.

In addition to displaying log information right on the GUI, the log information would be written to a log file for later use. Log folder, containing log files, can be chosen by users.

Last but not least, users are also able to set number of times to run benchmark applications for generating the final result or average result based on benchmark results generated by each time. This will help make the final result more accurate.

This number of times for running benchmark applications also can be set on the server side with VDBench Server Tool. However, if this number is set already from the

client side and the tool on the server side will use the value set from the client side. Otherwise, VDBench Server Tool will use the value set on the server side.

“LDAP Settings” tab

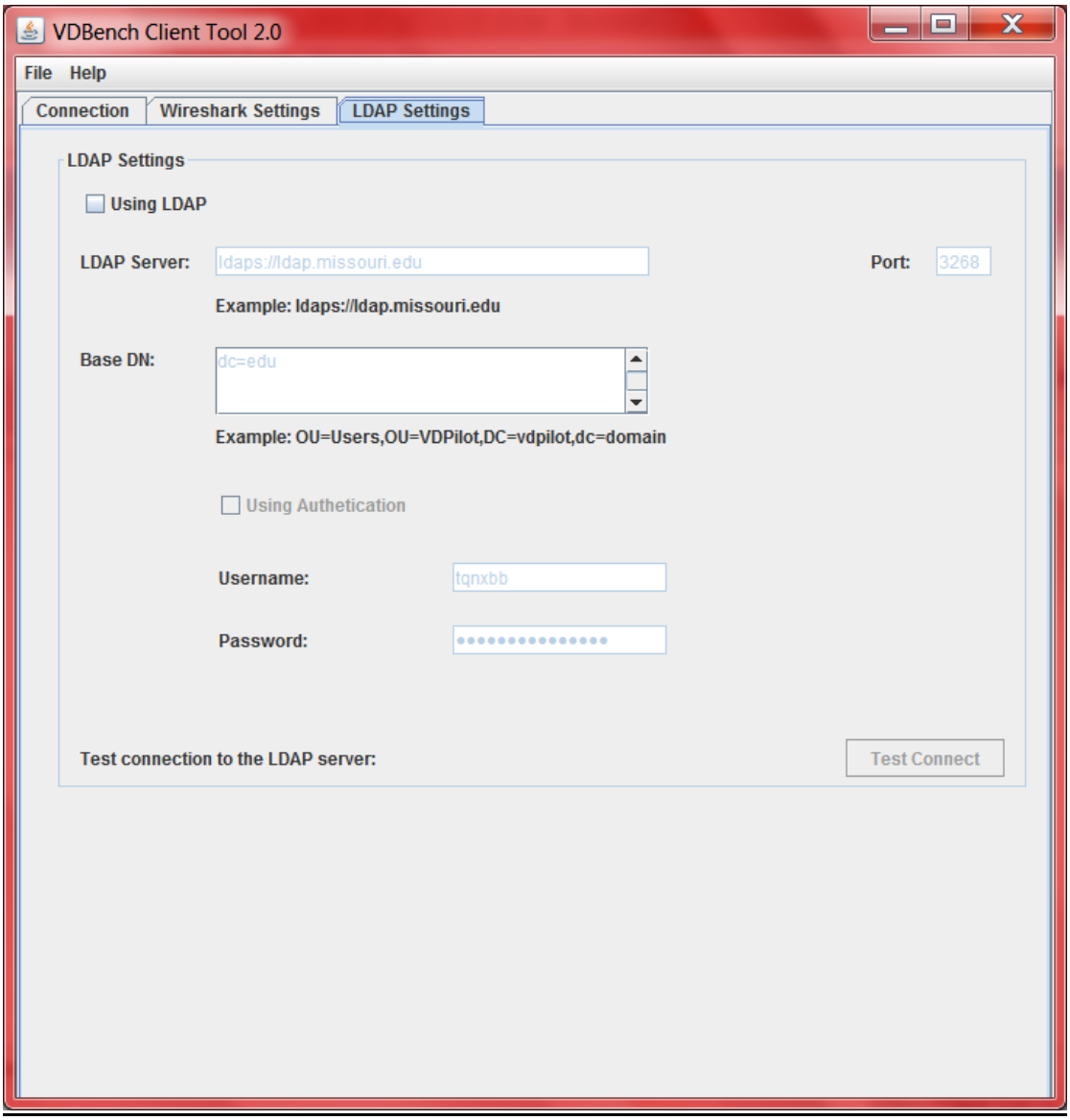


Figure 16: "LDAP Settings" tab of VDBench Client Tool

As a natural characteristic of Virtual Clouds for safety, a Thin-client connecting to the Clouds needs to be authenticated. As described in Fig.1 of the paper mentioned in previous section, a Thin-client will connect to the Broker Server, which uses LDAP server to authenticate connections. For that reason, users could set up LDAP information including LDAP server's IP address, Port number and Domain name. However, if the VDBench Toolkit is deployed in a network that does not require authenticating, for example in a lab network, so users could bypass LDAP authentication by disabling LDAP option.

More conveniently, users can test connection to and authentication on LDAP server to ensure that configuration information is correct before hitting "Connect" button on "Connection" tab.

3.3.1. VDBench Server Tool

For this version, VDBench Server Tool works only in Windows systems since it has to rely on PSTool. An alternative for PSTool, winexe tool is being considered. "Winexe" program can call a remote function or command on a remote Windows machine.

“General” tab

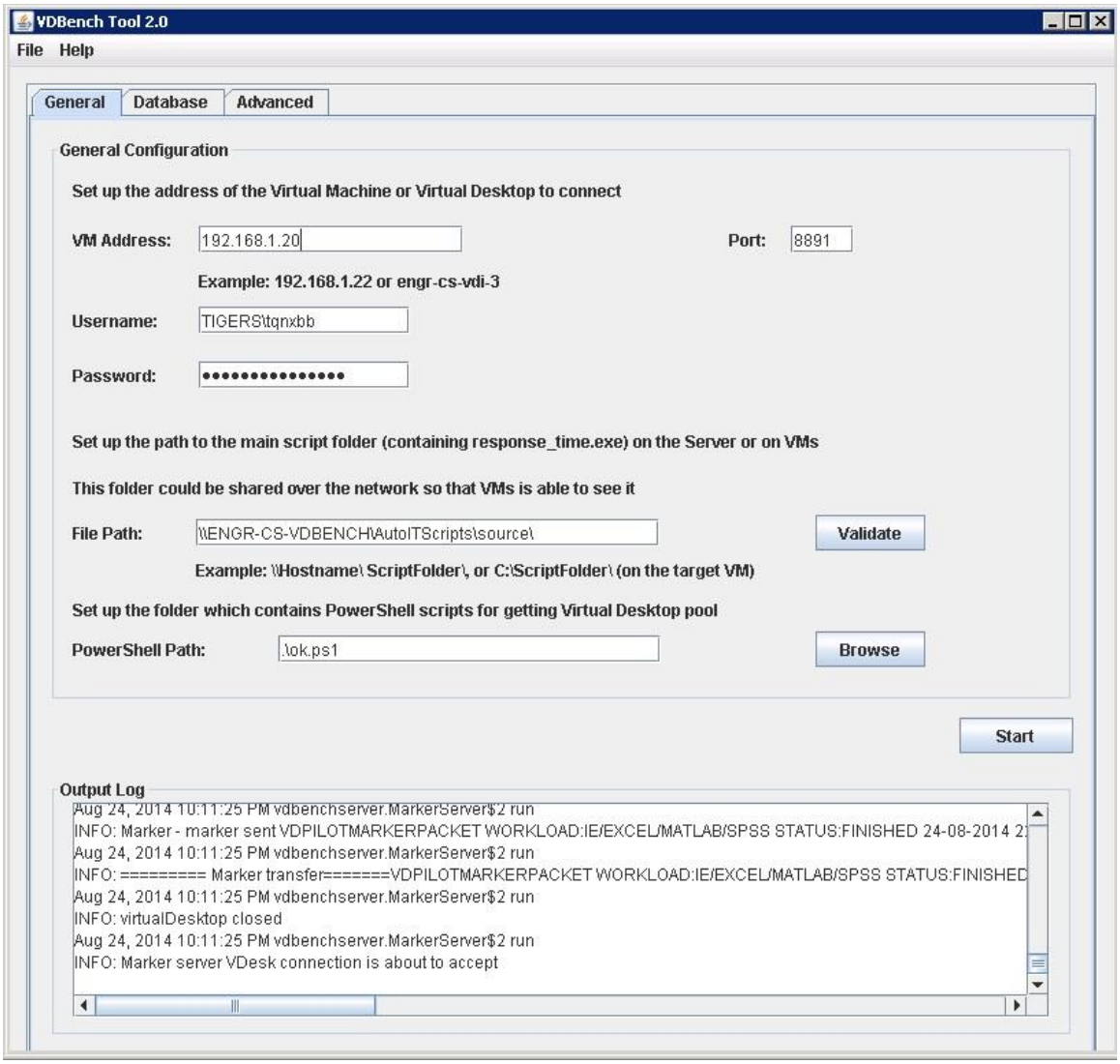


Figure 17: "General" tab of VDBench Server Tool

For the “General” tab or main tab, users are able to configure information of Virtual Machine (VM) on which benchmark applications are run, including VM’s IP address and connection port number based on which connection from the VDBench

Server to VM is established. For authentication on the VM, or administration account must be provided as well.

In addition, in order to execute applications on the VM automatically, the administrator who launches VDBench Server Tool needs to specify the path to main script folder which contains script files for running benchmark applications.

The main script folder should be shared over the local network in order the VM is able to see it or access it. Otherwise, the folder should be located in the VM, thus, it can access script files in the folder.

On the “General” tab, administrator is also able to configure the path to a PowerShell file which pulls out the list of available VMs. However, this function currently is not used since a specific VM is set as mentioned above.

Besides, like VDBench Client Tool, VDBench Server Tool also has an Output Log window which helps administrator know what is happening during running the tool.

“Database” tab

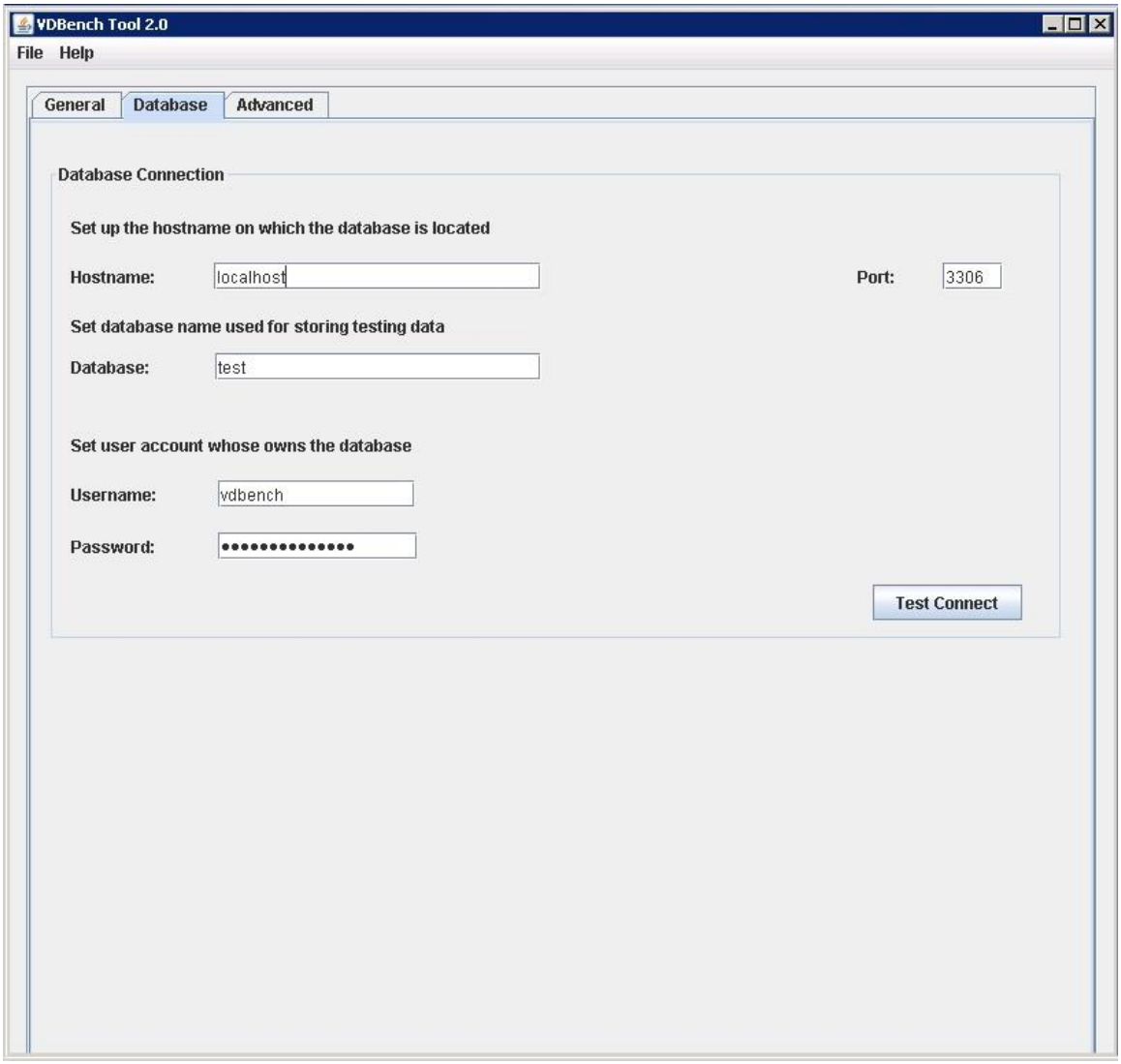


Figure 18: "Database" tab of VDBench Server Tool

Since VDBench Server Tool uses Database to store benchmark results, administrator who is in charge of configuring system must set up connection to an existing database. First, host name and port number are specified. Hostname is a

machine on which Database is being located. Hostname can be localhost or a network database server.

Then the administrator needs to specify a database, with existing tables used for storing log information, and authentication information to the database, including username and password.

Administrator is able to test connection to the database in order to determine whether configuration information is correct or not

“Advanced” tab

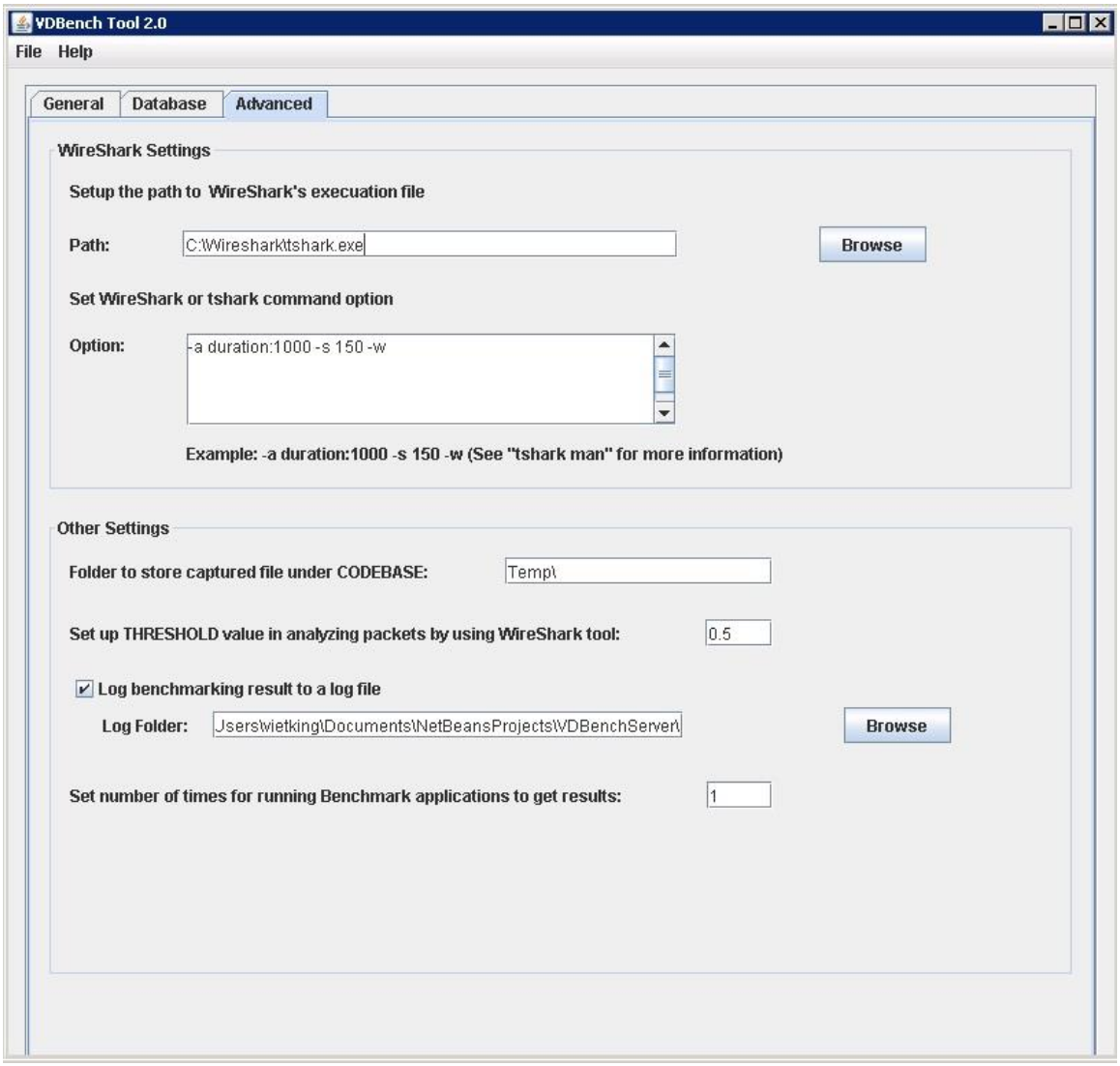
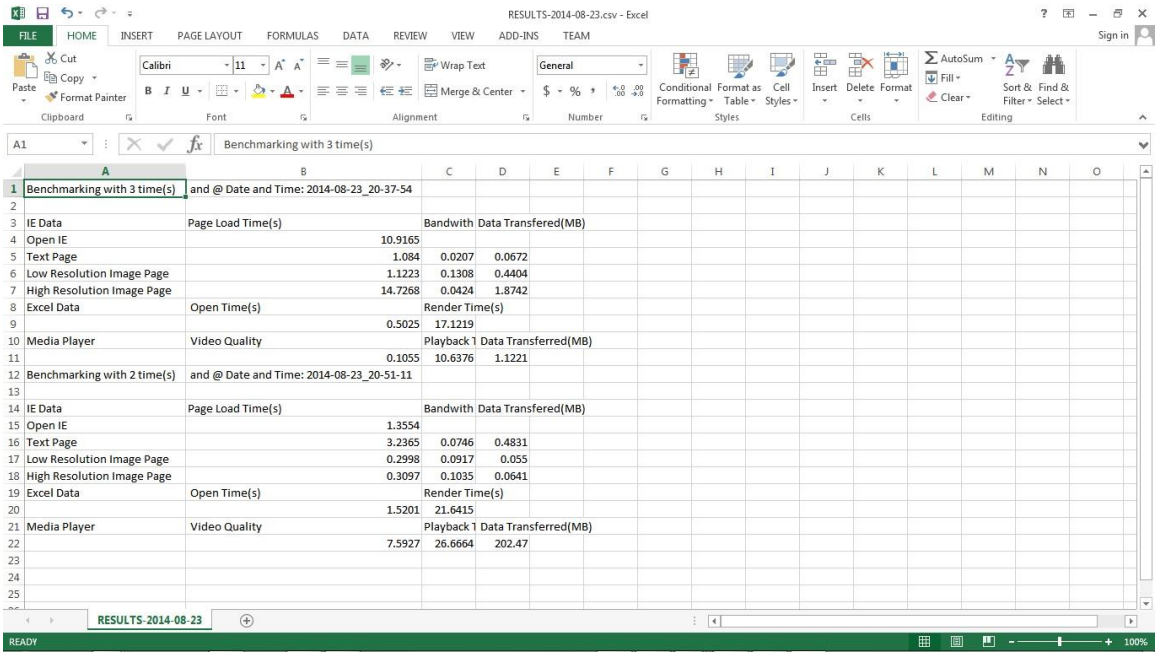


Figure 19: "Advanced" tab of VDBench Server Tool

3.3. New Output Design and Implementation

To support benchmark persons in reporting and analyzing benchmark results, I have saved those results into csv files, which can be opened and edited by Excel, and named them based on the current date. That will help users a lot in managing and referencing later. Moreover, since csv files can be opened by Excel, users are able to summarize benchmark results and insert diagrams based on that data very easily.



	A	B	C	D	E
1	Benchmarking with 3 time(s)	and @ Date and Time: 2014-08-23_20-37-54			
2					
3	IE Data	Page Load Time(s)	Bandwith	Data Transfered(MB)	
4	Open IE	10.9165			
5	Text Page	1.084	0.0207	0.0672	
6	Low Resolution Image Page	1.1223	0.1308	0.4404	
7	High Resolution Image Page	14.7268	0.0424	1.8742	
8	Excel Data	Open Time(s)	Render Time(s)		
9		0.5025	17.1219		
10	Media Player	Video Quality	Playback 1	Data Transferred(MB)	
11		0.1055	10.6376	1.1221	
12	Benchmarking with 2 time(s)	and @ Date and Time: 2014-08-23_20-51-11			
13					
14	IE Data	Page Load Time(s)	Bandwith	Data Transfered(MB)	
15	Open IE	1.3554			
16	Text Page	3.2365	0.0746	0.4831	
17	Low Resolution Image Page	0.2998	0.0917	0.055	
18	High Resolution Image Page	0.3097	0.1035	0.0641	
19	Excel Data	Open Time(s)	Render Time(s)		
20		1.5201	21.6415		
21	Media Player	Video Quality	Playback 1	Data Transferred(MB)	
22		7.5927	26.6664	202.47	
23					
24					
25					

Figure 20: Open benchmark results in csv files with Excel

3.4. New Configuration Method

As mentioned in previous sections, scripting plays a very important role in VDBench system. Thus, in order to leverage benchmarking work with different profiles, I have introduced a new configuration method. Instead of modifying settings or configurations in the script source files, users are able to modify them in on-the-fly manner without recompiling those script source files, as described in Software Enhancement Models.

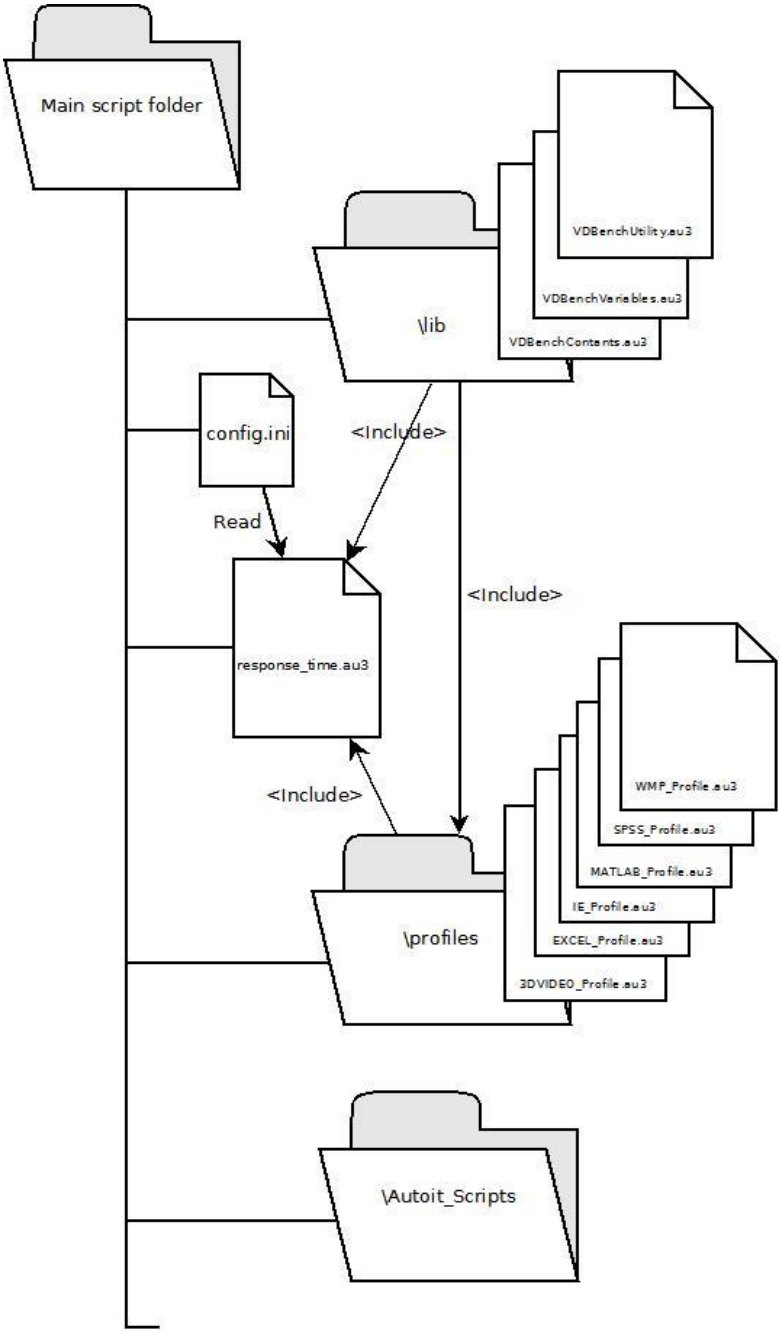
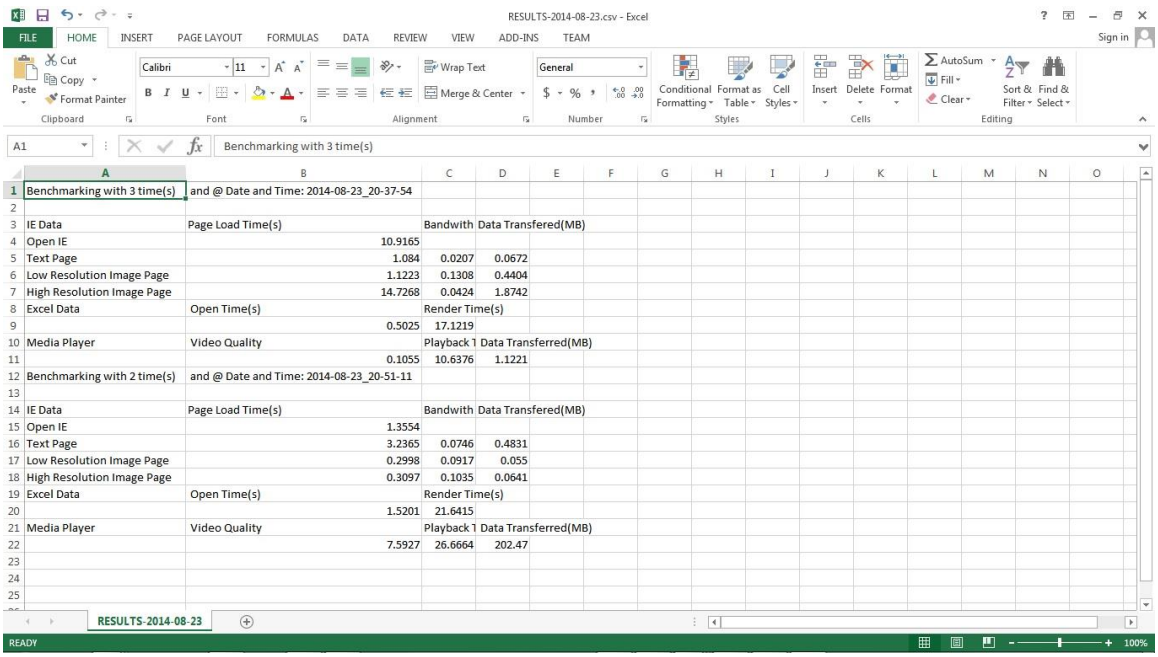


Figure 21: The main script file reads system configurations from config.ini

3.5. Data Collection

3.5.1. Collecting benchmark results

With an aim to leverage result collection for analyzing purpose, VDBench Toolkit introduces output in csv files which are used by Excel. Therefore, it is very convenient for users who do benchmarking to collect data and draw diagrams based on that data by using Excel.



	A	B	C	D	E
1	Benchmarking with 3 time(s)	and @ Date and Time: 2014-08-23_20-37-54			
2					
3	IE Data	Page Load Time(s)	Bandwith	Data Transfered(MB)	
4	Open IE	10.9165			
5	Text Page	1.084	0.0207	0.0672	
6	Low Resolution Image Page	1.1223	0.1308	0.4404	
7	High Resolution Image Page	14.7268	0.0424	1.8742	
8	Excel Data	Open Time(s)	Render Time(s)		
9		0.5025	17.1219		
10	Media Player	Video Quality	Playback T	Data Transferred(MB)	
11		0.1055	10.6376	1.1221	
12	Benchmarking with 2 time(s)	and @ Date and Time: 2014-08-23_20-51-11			
13					
14	IE Data	Page Load Time(s)	Bandwith	Data Transfered(MB)	
15	Open IE	1.3554			
16	Text Page	3.2365	0.0746	0.4831	
17	Low Resolution Image Page	0.2998	0.0917	0.055	
18	High Resolution Image Page	0.3097	0.1035	0.0641	
19	Excel Data	Open Time(s)	Render Time(s)		
20		1.5201	21.6415		
21	Media Player	Video Quality	Playback T	Data Transferred(MB)	
22		7.5927	26.6664	202.47	
23					
24					
25					

Figure 22: Open benchmark results in csv file with Excel

All benchmark results collected in the same day will be saved into a unique csv identified by the date the results are created. It is easy for users to organize their benchmark results.

Since csv files can be opened by Excel, users are able to summarize benchmark results and insert diagrams based on that data very easily as following.

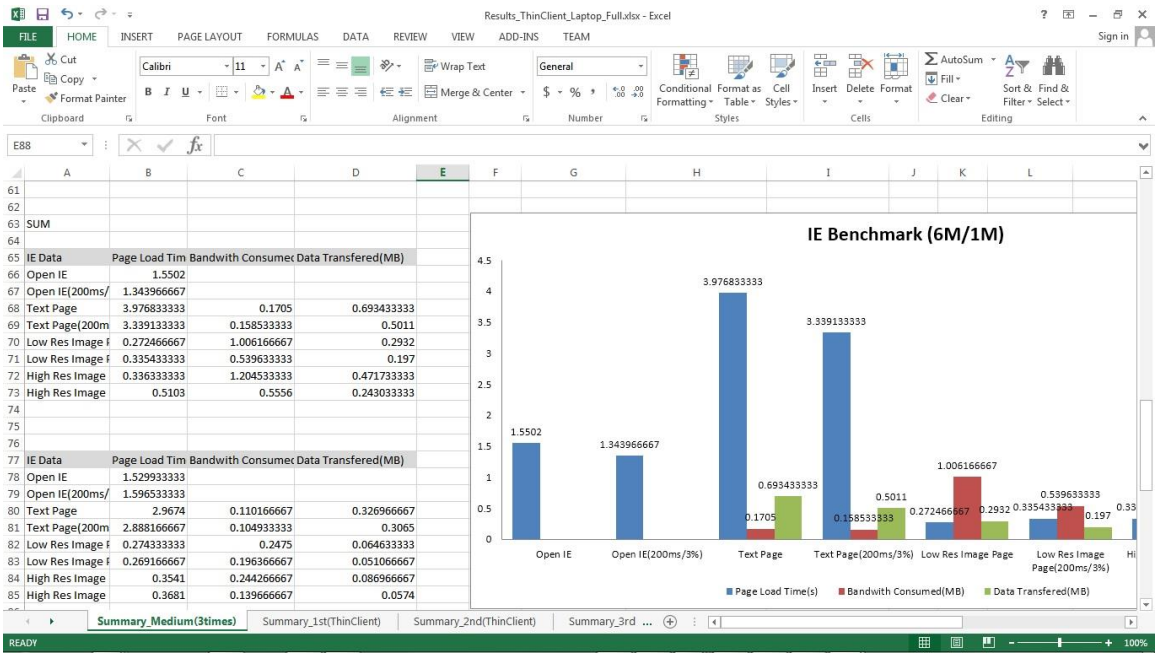


Figure 23: Collect benchmark results with different network conditions and draw diagrams based on that data by using Excel.

3.5.2. Viewing Benchmark Traces with Wireshark

Another tool to deal with benchmark results obtained by VDBench Client Tool is Wireshark. Since VDBench Client Tool uses Wireshark to monitor network traces during benchmarking, users are able to take advantage of IO Graph Tool of Wireshark to view benchmark traces.

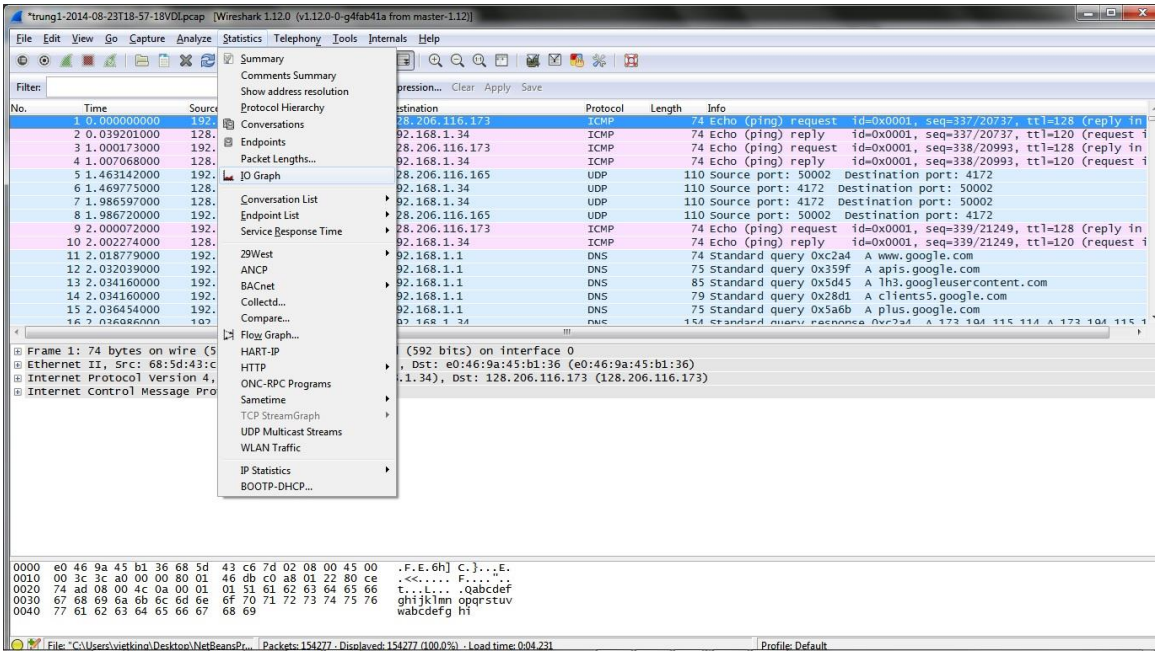


Figure 24: Open Pcap file obtained after benchmarking by using Wireshark

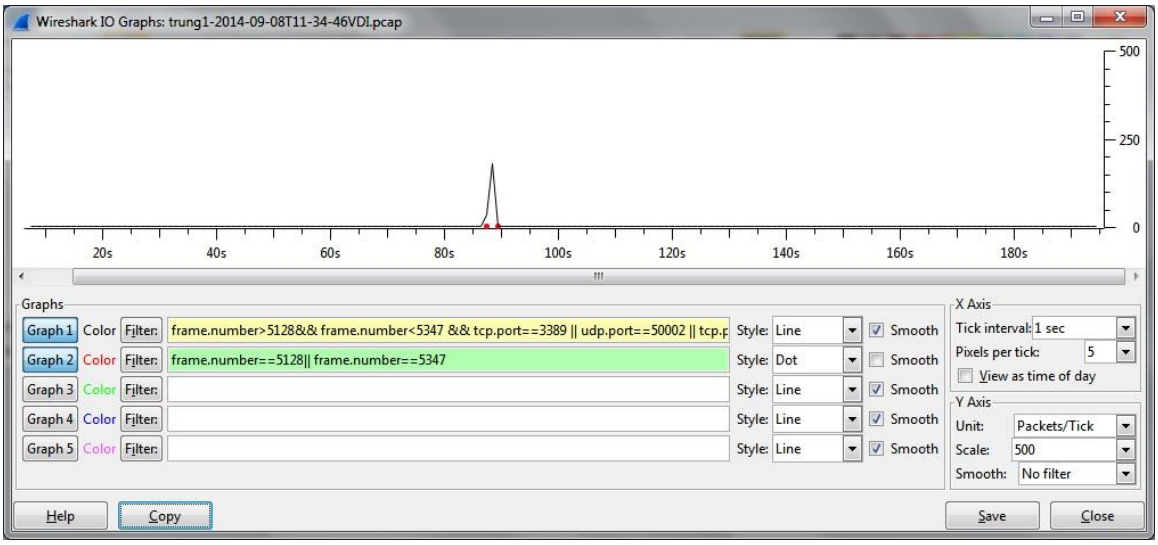


Figure 25: Use IO Graph of Wireshark to view benchmark traces

Above diagram is benchmark traces for Internet Explorer open load. In order to view traces for a particular task (such as open) of a benchmark application such as Internet Explorer, users need to access to the corresponding log file of VDBench Server Tool on the Benchmark Server for start and stop marker as shown in following figure.

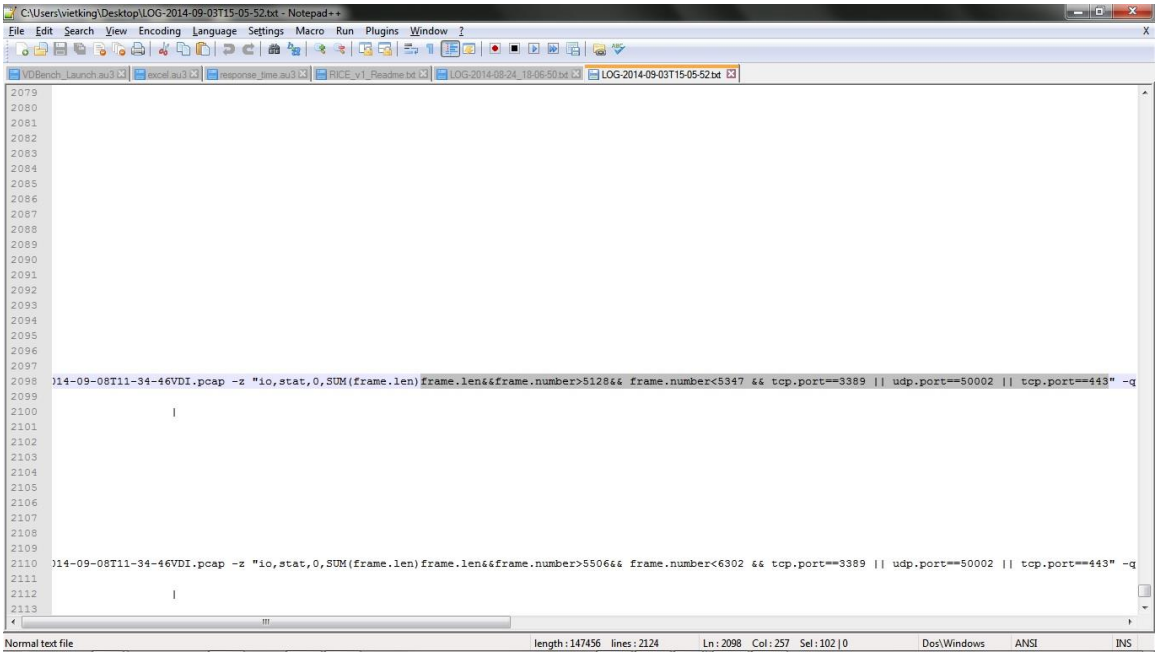


Figure 26: The filter of benchmark traces of Internet Explorer open load found in the log file on Benchmark Server

Since a task load is determined by start and stop marker (red dots), to view benchmark traces of that task users need to obtain start and stop marker (from the log file or marker file) which are used for filters in IO Graph of Wireshark.

3.5.3. Some examples of Benchmark Traces with Wireshark

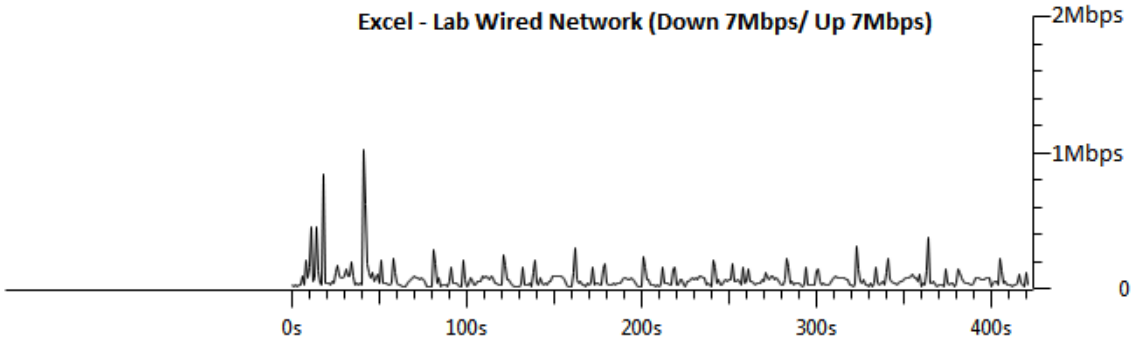


Figure 27: Run Excel 10 times benchmarking trace

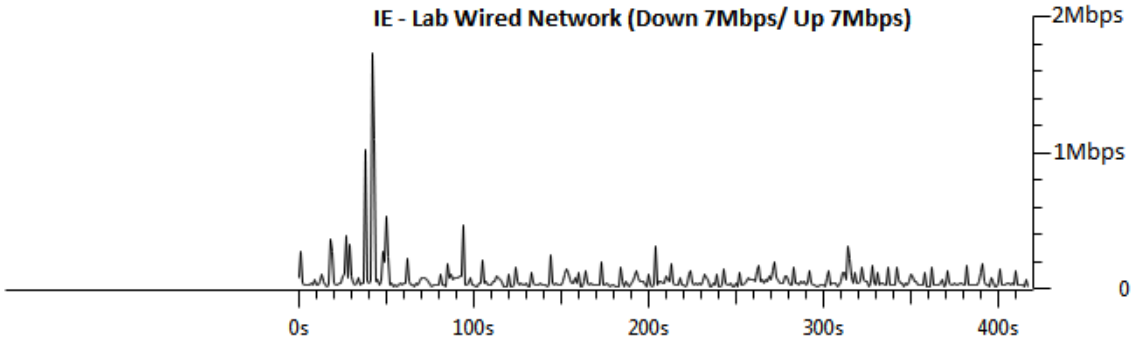


Figure 28: Run IE 10 times for benchmarking trace

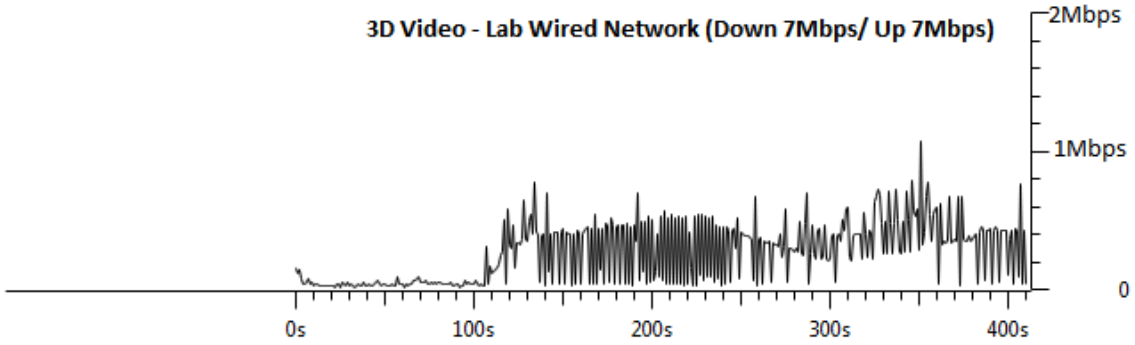


Figure 29: Play 3D Video for benchmarking trace

3.6. Use in Reality

As discussed at the first place, benchmarking is an essential work in cloud computing in order to validate QoE provided by the cloud. Therefore, benchmark toolkit as VDBench can be used for all cloud systems which need to validate QoE in those environments. As an example, I have used the toolkit for benchmarking in virtual environment cloud in the project named “LIDAR-based Virtual Environment Study for Disaster Response Scenarios”. With benchmark results obtained by the toolkit, we are able to validate QoE in that environment for disaster response scenarios.

More details, in disaster scenarios many videos from many perspectives would be collected by civilians and surveillance cameras at the disaster side. Those videos can be extremely useful for officials and emergency responders who need to ascertain the extent of the damage. However, handling such large collections of videos and performing this type of surveillance can be quite challenge. With viewing 3D model out of those 2D videos can provide better observation or understand on what is happening on that disaster site.

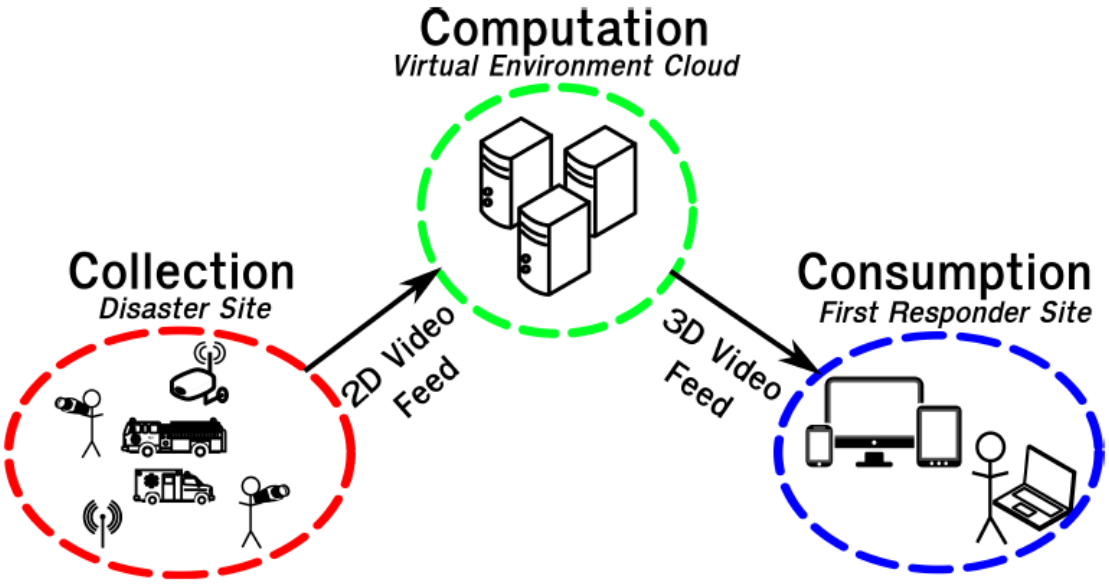


Figure 30: Dynamic interactive 3D scene visualization system for disaster scenarios

The above figure is the overview of the 3D model system for disaster scenarios. All 2D videos collected on a disaster site will be fed to or stored on one or more data centers of cloud. Subsequently, that kind of data will be processed to build up in 3D models or 3D videos. Eventually, those 3D videos will be delivered to responders as a service through the Internet. They can then use their own personal devices such as smart phones, tablets or laptops to view the models. That, in turn, will help them understand the situations happening on the disaster sites and then provide countermeasures quicker and better.

However, QoE obtained by responders or users through the system should be validated to determine whether the system works well or not, that is when

benchmarking comes into the play. By using VDBench toolkit, I can validate QoE of 3D model though the system without any problem. In details, comparison between popular applications such as Internet Explorer and Excel, and 3D video playback application should be made to give better understanding on how much more of bandwidth 3D video playback application consumes compared to other applications.

I have done benchmarking with three applications by using VDBench and Wireshark tool.

Bandwidth Measurement	Excel (Kbps)	IE High Resolution Image(Kbps)	3D Video (Kbps)
Campus Wireless Network(8-9Mbps/10Mbps)	75.944	93.2	238.716
Lab Wired Network(7Mbps/7Mbps)	41.663	73.6	232.969
Lab Wired Network(Down 5Mbps/ Up 3Mbps)	38.923	43.2	172.814
Lab Wired Network(Down 3Mbps/ Up 1Mbps)	37.194	38.514	168.322

Figure 31: Bandwidth consumed by Excel, IE and 3D Video on different network conditions

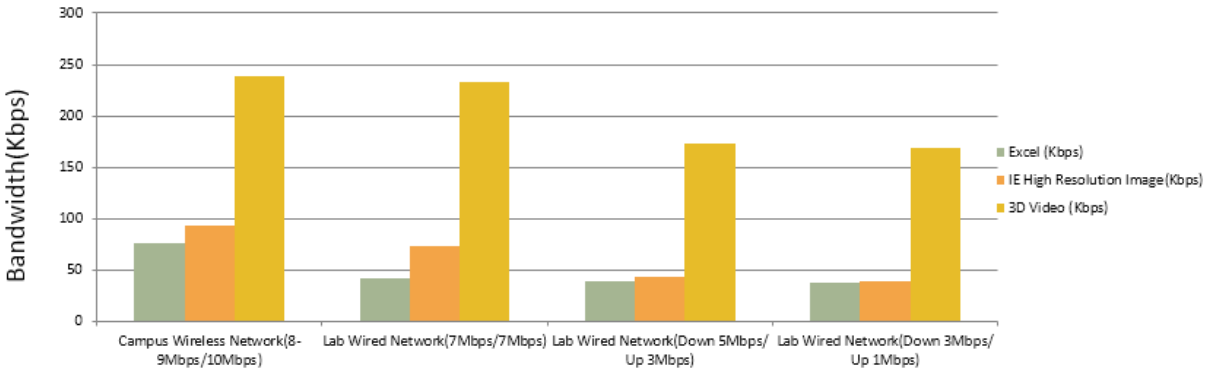
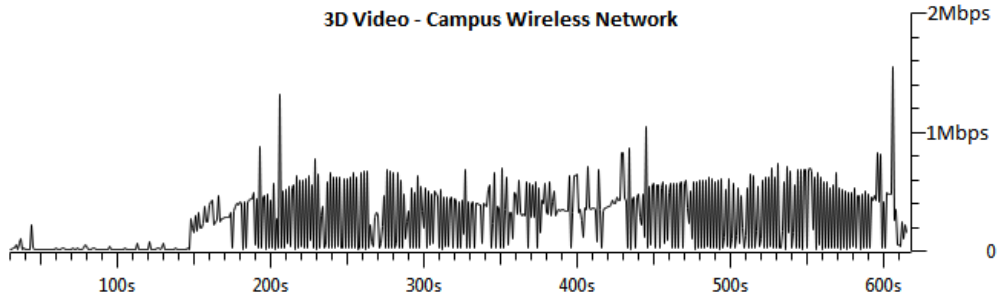
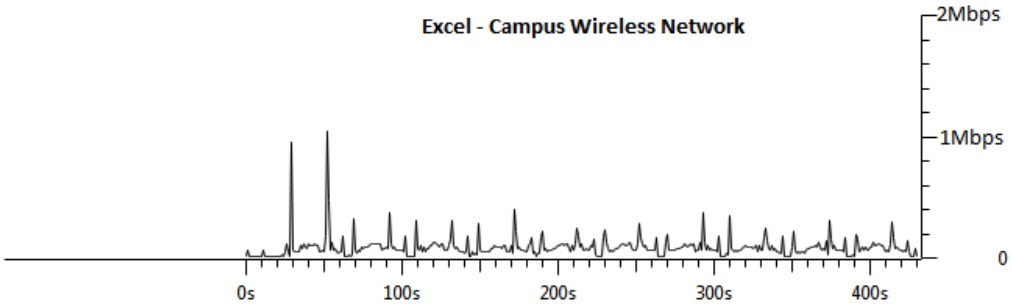


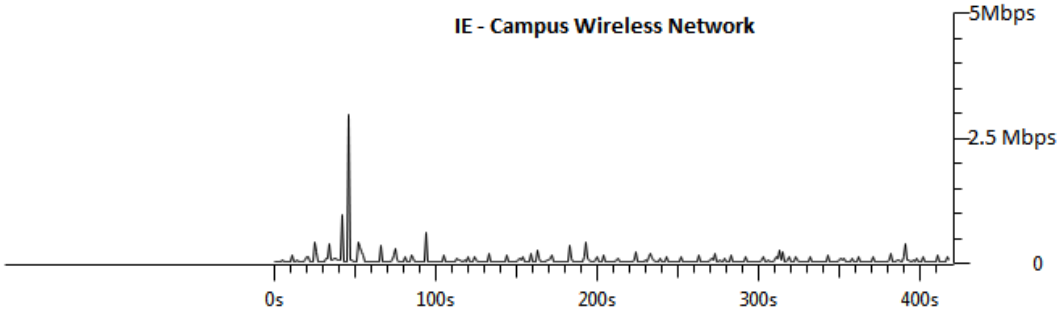
Figure 32: Bandwidth consumed by Excel, IE and 3D on different network conditions

In other option, by using Wireshark tool we are able to compare bandwidth consumed by those applications.

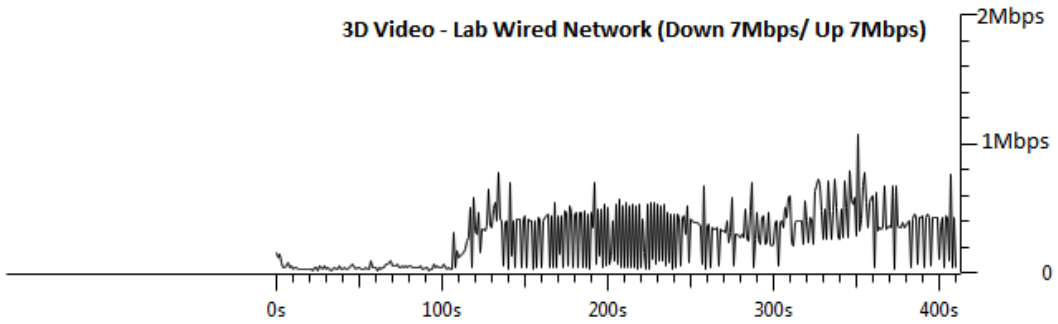
Bandwidth consumed by those applications on Campus Wireless network

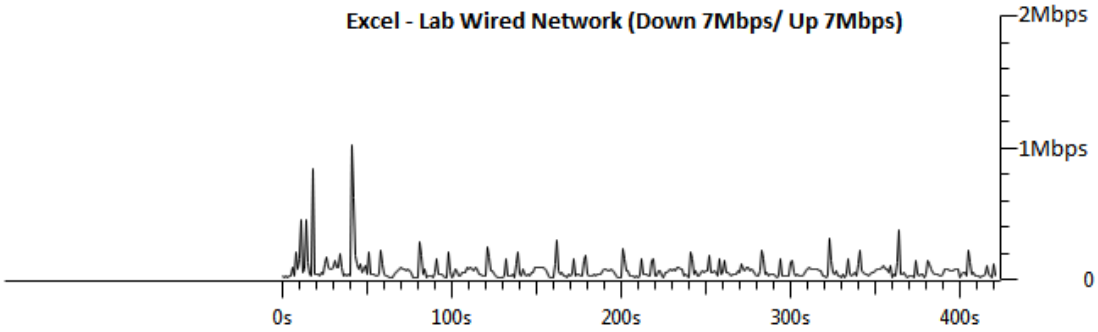


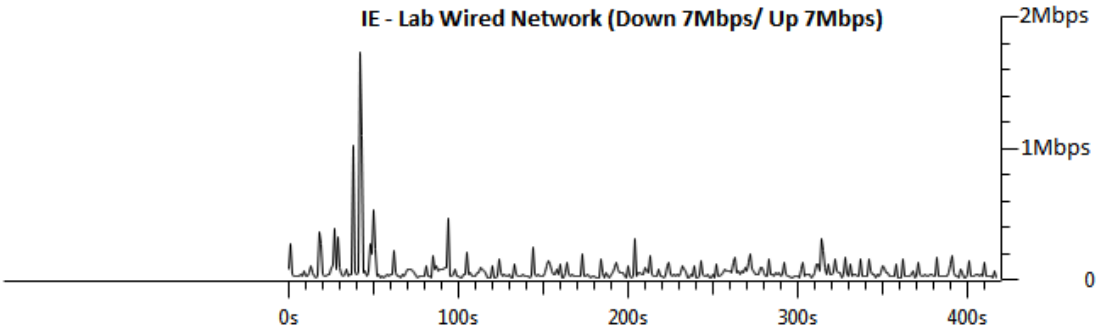




Bandwidth consumed by those applications on VIMAN lab network







4. CONCLUSIONS

Today, we are living in cloud computing era. We have been enjoying advantages that cloud computing has been bringing to our society. In contrary, we also dealing with challenges to deliver better computing services to users. Apart from the challenge of how to serve a huge number of customers up to thousands or millions, ensuring quality of experience for users at the least cost is another big challenge that cloud computing has to solve. And benchmarking is an important step to overcome that challenge.

By digging deep into benchmarking in virtual desktop environments, I understand concepts related to benchmarking and how important benchmarking is to validate QoE in virtual environments. Also, by doing benchmarking in virtual desktops on a variety of network health conditions, I can understand how application performance is effected by network degradations. In addition, by measuring open time, execution time of performing a task, data transferred and bandwidth consumed, I can know whether it is good or bad performance of an application in virtual desktops. Based on that, I can know how to select good system configuration and good protocol in order to obtain good QoE. Moreover, I can also understand how to validate whether the system is working well and how much resource is needed to ensure QoE.

Based on obtaining background knowledge of benchmarking in virtual desktop environments, I have come beyond it to revise the use of VDBench - a first attempt of building a toolkit to automate benchmarking. Revising work includes data collection, analysis strategy as well as documentation and feature set to make the tool easier to use. More importantly, all of this package I have is able to use or apply it in real application contact for 3D video model viewer application which helps to understand this source of disaster scenarios.

Last but not least, I can understand how machine learning can be used to really go behind the scene of benchmarking and come up with poor-acceptable-good assessment that we have in earlier work. However, I want to improve it by using machine learning approach.

5. FUTURE WORK

With VDBench toolkit, benchmark results including time, bandwidth consumed and data transferred for each virtual component in virtual desktops have been collected and viewed in Excel by benchmark persons. However, one limitation is still remained. That kind of benchmark data does not tell right away whether or not QoE is poor, acceptable or good in various network health conditions. In order to obtain that QoE information, users still need further steps or analysis such as drawing diagrams by Excel or graphs by using Wireshark and some knowledge on benchmarking.

With an aim to resolve the above limitation, machine learning should be considered as a good approach. By applying machine learning on benchmark results combined with subjective QoE collected from real users or testers, QoE with scale from poor, acceptable to good can be achieved right way by running the toolkit. That, in turn, will be very useful to benchmark persons or administrators of cloud systems.

Machine learning in this approach is supervised learning, or more detail, classification since benchmark results as input data and subjective QoE as output. From input and output data, we can build a model and revise it by data collected through time of using the toolkit. We would apply several classification algorithms including Naïve Bayes classifier, Decision trees, nearest neighbors (kNN) and neural networks. Subsequently, a comparison of results among those algorithms to find the best one for benchmarking.

In addition, because VDBench toolkit is built in Java, I will use WAKA library – a Java library for machine learning, and apply it right in the toolkit for more comprehensive benchmark results.

6. BIBLIOGRAPHY

[1] A. Berryman, P. Calyam, M. Honigford, A. M. Lai. VDBench: A Benchmarking Toolkit for Thin-client based Virtual Desktop Environments. *In Cloud Computing Technology and Science*, 2010.

[2] Y. Xu, P. Calyam, D. Welling, S. Mohan, A. Berryman, R. Ramnath. Human-centric Composite Quality Modeling and Assessment for Virtual Desktop Clouds.

[3] J. Nieh, S. Yang, N. Novik, "Measuring thin-client performance using Slow-motion benchmarking", *ACM Transactions on Computer Systems*, Vol. 21, No. 1, Pages 87-115, 2003.

[4] J. Nieh, S. Yang, N. Novik, "Measuring thin-client performance using Slow-motion benchmarking", ACM Transactions on Computer Systems, Vol. 21, No. 1, Pages 87-115, 2003.

[5] A. Lai, J. Nieh, "On The Performance Of Wide-Area Thin-Client Computing", ACM Transactions on Computer Systems, Vol. 24, No. 2, Pages 175-209, 2006.

[6] J. Rhee, A. Kochut, K. Beaty, "DeskBench: Flexible Virtual Desktop Benchmarking Toolkit", Proc. of Integrated Management (IM), 2009.

[7] N. Zeldovich, R. Chandra, "Interactive Performance Measurement with VNCplay", Proc. of USENIX Annual Technical Conference, 2005.

[8] Thinstation: Open-source Thin-client Operating System -http://www.thinstation.org

[9] Wyse Thin Client Products -http://www.wyse.com

[10] P. Hasselmeyer, N. dHeureuse, “Towards Holistic Multi-Tenant Monitoring for Virtual Data Centers”, Proc. of IEEE/IFIP NOMS Workshop, 2010.

[11] S. De Chaves, R. Uriarte, C. Westphall, “Toward an Architecture for Monitoring Private Clouds”, IEEE Communications Magazine, Vol. 49, No. 12, Pages 130 - 137, 2011.

[12] S. Clayman, A. Galis, C. Chapman, et. al., “Monitoring Service Clouds in the Future Internet”, Towards the Future Internet - Emerging Trends from European Research, IOS Press ISBN 978-1-60750-538-9, 2010.

[13] V. Emeakaroha, M. Netto, et. al., “Towards Autonomic Detection of SLA Violations in Cloud Infrastructures”, Future Generation Computer Systems, Vol. 28, No. 7, Pages 1017-1029, 2012.

[14] J. Shao, H. Wei, Q. Wang, H. Mei, “A Runtime Model Based Monitoring Approach for Cloud”, Proc. of IEEE Conference on Cloud Computing (CLOUD)”, 2010.

[15] Dusi, M. Napolitano, S. Niccolini, S. Longo. A closer look at thin-client connections: statistical application identification for QoE detection. *Communications Magazine*, Vol. 50, Pages 195-202, 2012.

[16] Giang Bui, Prasad Calyam, Brittany Morago, Ronny Bazan Antequera, Trung Nguyen, Ye Duan, “LIDAR-based Virtual Environment Study for Disaster Response Scenarios”, 2014.