Performance Observations Hosting Graphically Intensive Simulations in a 'Cloud' Environment

James Benslay, Rick Osborne, Joe Clapis
The MITRE Corporation
Orlando, FL
jbenslay@mitre.org, rosborne@mitre.org
jclapis@mitre.org

Colleen Matthews, Robert Parrish
PM ITE
Orlando, FL
colleen.d.matthews.civ@mail.mil
robert.s.parrish6.civ@mail.mil

ABSTRACT

Cloud computing technologies have matured and expanded dramatically in recent years. Along with this expansion, DoD organizations are being driven to take advantage of this technology and host as many computing systems as possible from Core Data Centers. This makes sense in many regards, except when it comes to graphically intensive applications. Virtual Machines (VMs) perform very poorly when having to virtualize the more advanced abilities of hardware Graphical Processor Units (GPUs) into software. Other technologies have been introduced to attempt to counter this deficiency, namely, graphics cards that have GPUs that can be divided into separate GPU segments, called "vGPUs" that are then directly assigned to VMs. These technologies offer the promise of being able to host even graphically intensive applications in a cloud environment. However, how well do these combined technologies perform when hosting demanding simulations? Can existing simulation systems be migrated to such a cloud environment? Are the technologies sufficiently mature enough to meet current performance demands? How well do they perform under realistic network conditions? To answer these questions, the PEO STRI Project Manager for Integrated Training Environment (PM ITE) teamed with the MITRE Corporation to conduct a series of experiments to evaluate the feasibility of hosting graphically intensive simulations within a hypervisor/cloud environment using advanced vGPU technologies. This paper reports on how the experiments were configured and executed, and presents a variety of performance observations including draw rates, CPU and GPU utilization, network bandwidths, etc. The paper also discusses what trade-offs were evaluated in order to host the highest number of VMs from a single server with the most acceptable performance, as well as the impact of operating within realistic network conditions. Finally, implications for potential future use of the technology are discussed.

ABOUT THE AUTHORS

James Benslay is a Lead Information Systems Engineer at The MITRE Corporation. He is a retired USAF Communications Officer and has over 21 years' experience leading and participating in a wide variety of major information systems engineering projects. He is currently assigned to the team supporting the Project Manager Integrated Training Environment (PM ITE) at U.S. Army PEO STRI. His current interests include model based systems engineering, cloud technologies, and gamification. He earned his B.S. in Computer Science from National University and his M.S. in Computer Systems from the Air Force Institute of Technology.

Colleen Matthews is a Project Director/Lead Engineer for the Project Manager Integrated Training Environment (PM ITE) at U.S. Army PEO STRI. She has 20 years of experience working in DoD industry on live weapon systems and virtual/constructive simulations in addition to 12 years working for PEO STRI in DoD Acquisition. Her current interests include gaming technologies, augmented reality and mobile applications. She earned a B.S. in Electrical Engineering from the University of Central Florida.

Rick Osborne is a Lead Simulation Engineer at The MITRE Corporation. He is currently assigned to the team supporting the Project Manager Warrior Training Integration (PM WTI) at U.S. Army PEO STRI. His current interest includes cloud computing, agile development, and game development. He earned his B.S. in Computer Engineering from Christopher Newport University, M.E. in Computer Engineering from Old Dominion University, and M.S. in Modeling and Simulation from University of Central Florida.

Joe Clapis has been a Software Systems Engineer at MITRE for five years. He specializes in reverse engineering and integrating disjoint systems into new capabilities. Joe has developed solutions for the systems administration, computer vision, video compression, network administration, virtualization management and aerospace domains. He is an acting SME of GPU virtualization technology and technology-driven experimentation and benchmarking evaluations. In his spare time, Joe is an amateur astronomer and model rocket enthusiast.

Robert Parrish is a Chief Engineer for the Project Manager Integrated Training Environment (PM ITE) at U.S. Army PEO STRI. He has 33 years of experience working in DoD acquisition community. He worked 10 years with the Navy and 23 years with the Army as a chief/systems/project engineer for both virtual and live training systems. His current interests include systems engineering, synthetic training environment, and immersive technologies. He earned his B.S. in Electrical Engineering from the Georgia Institute of Technology in December 1982.

Performance Observations Hosting Graphically Intensive Simulations in a 'Cloud' Environment

James Benslay, Rick Osborne, Joe Clapis
The MITRE Corporation
Orlando, FL
jbenslay@mitre.org, rosborne@mitre.org
jclapis@mitre.org

Colleen Matthews, Robert Parrish
PM ITE
Orlando, FL
colleen.d.matthews.civ@mail.mil
robert.s.parrish6.civ@mail.mil

INTRODUCTION: THE CATALYST, THE PROBLEM, AND THE CHALLENGE

The Catalyst

In recent years, there has been a concerted push within DoD organizations to relocate computing servers (and applications) to Consolidated Data Centers (CDCs). The reasons for doing so are obvious:

- Protection: moving to CDCs helps to ensure the continuous operation of critical information resources in facilities specifically designed to protect said resources;
- Share Resources: moving to CDCs allows the DoD to be more resource efficient by sharing computing servers where possible through the use of Virtual Machine (VM) technologies;
- Easier Hardware Allocation: focusing on using computer server resources at a CDC allows an organization to scale up or down their operations in relatively short order without having to invest in more hardware or let excess hardware go underutilized.

The Problem

However, not all software applications or systems are well suited for operating in a VM environment. Specifically, those applications that require advanced computer graphics are not well suited to a VM environment because of the limitations of the hypervisor to emulate via software the advanced capabilities of graphics hardware. With machine virtualization, it is possible to run multiple, independent instances of a number of different operating systems (i.e., Windows, Linux) from within a single physical computer/server. The software that makes this possible, the hypervisor, emulates the computer's physical resources (such as CPU, memory, network interfaces, USB interfaces, disk drives) to the guest operating systems such that the guest operating systems perform as if they are on their own separate hardware.

The principal benefit of machine virtualization is the potential cost savings as virtualization encourages the consolidation of physical hardware and allows for more effective use of existing hardware by taking advantage of underutilized processing power. Virtualization also potentially saves money by not having to house, power, cool, and maintain excess hardware. Additional benefits to virtualization include dynamic instantiation and/or termination of virtual machines in a short time-frame in order to accommodate rapidly changing computational demands.

There are some difficult complications with machine virtualization, however. Whereas hypervisors are able to efficiently handle most physical computer resources such as those mentioned above, one physical resource that is difficult to virtualize is the graphics card. Graphics cards are separate computers all to themselves with their own dedicated Graphical Processing Unit (GPU), memory, input/output bus, and timing clocks that are independent from the computer system's clock. To make a virtual GPU, all of that hardware with the individual behaviors has to be emulated by the hypervisor's CPU; this is technically infeasible for advanced graphics functions due to fundamental differences between the CPU and GPU architectures. Consequently, most applications that have a need for advanced computer graphics typically perform very poorly, if at all, in a VM environment.

One approach for dealing with this problem is to allow "pass-through" control from the VM to the graphics card. In this scenario, the hypervisor is configured to allow one VM to use and control the graphics card resources, hence passing the VM's graphics requests through to the graphics card. The major draw-back of this approach is that only

one VM can have pass-through access to the graphics card. It is not possible to allow two VMs simultaneous control over the same graphics card. This would cause unpredictable and undesirable results.

In order to address this problem, graphics cards are now available that are essentially multiple graphics cards in one. Utilizing these types of cards allows the hypervisor to assign a flexible portion of the overall graphics card to a specific VM so that the VM is able to have direct control of that portion of the graphics card assigned to it. This type of graphics card is generally referred to as having "virtual GPUs" or "vGPUs". However, it is important to note that in order to utilize a graphics card with vGPUs, both the host hypervisor as well as the guest VM operating systems must utilize hardware drivers specifically designed to allow for GPU virtualization.

The Challenge

In April 2015, the Program Manager (PM) for Integrated Training Environment (ITE), under the Program Executive Office (PEO) for Simulation, Training, and Instrumentation (STRI), tasked MITRE to evaluate the feasibility of running a graphically intensive, interactive simulation program from within a simulated "cloud"/ VM environment and gather performance data. Specifically, PM ITE requested MITRE:

- Evaluate the feasibility of using vGPU technologies to run graphically demanding simulation software in a "cloud"/VM environment,
- Report on the hardware and software components used to accomplish this, and
- Explore the range of server and network conditions and performance requirements that would allow for "acceptable" usability of the simulation software.

TEST OBJECTIVES

In order to accomplish the overall task, the team devised a series of high level objectives to guide the activities during the course of the overall task.

- **Objective 1**: Research and assemble the hardware and software components necessary for the experiment.
- **Objective 2**: Validate that the simulation software will work in the cloud/VM environment using vGPU technologies, and that the simulation software can be controlled using only the point-of-need client machines.
- **Objective 3**: Determine the Upper Performance Bound (UPB) for the maximum number of VMs that can be run concurrently from the VM server and still maintain an acceptable level of performance.
- **Objective 4**: Determine the limiting factors encountered when running the simulation in a cloud/VM environment.
- **Objective 5**: Determine the amount of network bandwidth required to support running the simulation in a cloud/VM environment.
- **Objective 6**: Determine how the simulation performs when network bandwidth and latency restrictions are applied.

DEFINING "ACCEPTABLE"

The team defined "acceptable level of performance" as having both objective and subjective measurement components. The objective measurement is primarily gauged by graphical frames per second generated within the VMs and at the point of need client machines. The subjective measurement is the user's opinion of how the graphics appear as well as how responsive the simulation "feels".

Objective Measure

For the objective element, the key measurement is the draw rate that occurs within the VM on the server and on the point of need client machines. The draw rate is a common metric for measuring computer graphics performance and is expressed in "frames per second" (fps). Essentially, the higher the draw rate the better the graphics performance and the higher degree of acceptability. Based on input from PEO STRI simulation subject matter experts, the minimum usable draw rate for personal computer based simulation/gaming applications is approximately 18 fps.

Subjective Elements

For the subjective element, the team members focused on two principal areas: graphics quality and overall responsiveness.

1. **Graphics Quality**. Regarding graphics quality, it is generally true to say that if there is a high draw rate, then the graphics quality should also be good. There are exceptions however, when other factors can negatively affect the graphics quality independent of the draw rate. These exceptions occur specifically in the case of more advanced setups like when streaming VM desktops. Examples of these factors include downstream network data rate bottlenecks as well as higher amounts of transmission latencies. Examples of poor graphics quality include pixilation (a loss of detail), stuttering or jerkiness, and increased incidences of tearing. Accordingly, it is possible to have an acceptable VM draw rate but unacceptable graphics quality at the end client.

2. **Overall Responsiveness**. Regarding overall responsiveness, it is generally true to say that the lower the amount of network transmission latency, the better the simulation responsiveness should be. This is analogous to saying that when a person is playing a computer game on a standalone machine with no network involved, and when the person provides inputs to the game with a mouse or controller, the person will expect to see the results on the screen immediately with no discernable delay. (Note that this topic of responsiveness applies specifically to advanced setups when streaming simulations over networks and generally does not apply otherwise.)

   It is also generally true to say that the higher the amount of network transmission latency, the worse the simulation responsiveness should be. Unfortunately, there is no absolute scale that specifies at what degree of latency the responsiveness of a simulation will become unacceptable. Some simulations are more sensitive than others with regards to how much a certain amount of network latency will affect the simulation and thus acceptability. For example, a slower paced simulation with a lesser amount of user input would generally tolerate a higher amount of network latency than a faster paced game with near constant user input. That is to say, the faster the gameplay the more sensitive the players will be to game responsiveness and the less tolerant of network latency.

Evaluating Subjective Areas

When evaluating the subjective areas during the exploratory tests, the team members simply used their own judgement and experience with other similar simulations and computer games to make a determination if the performance was acceptable or not. There is simply no other effective way to make these determinations. It is important to note however, that determining acceptability is a highly subjective area dependent on the evaluator's tolerance.

For instance, a person who plays racing simulation games only seldom and just for fun would in general be much more tolerant of poorer system performance than another person who plays frequently on a high-end gaming machine. The latter would be much more discriminating in system performance, the former not much so. Accordingly, it is quite possible to have a level of system performance that is acceptable to one group of people but not another. The team members sought to eliminate as much of the single-person bias as possible by discussing the performance of each test and coming to agreement on the subjective analysis.

OBJECTIVE 1: RESEARCH AND ASSEMBLE

The first objective was to research and assemble the hardware and software components necessary for the experiment. The following subsections describe the pertinent elements for this objective.

Overall Network Configuration

Figure 1 below depicts the hardware components and the network topology for the experiment. The network was a closed network and these were the only elements in the network. Note that although we did not get there, an original goal of the experiment was to attempt to run 16 VM/client machines simultaneously, hence the 16 thin-client laptops in the diagram.

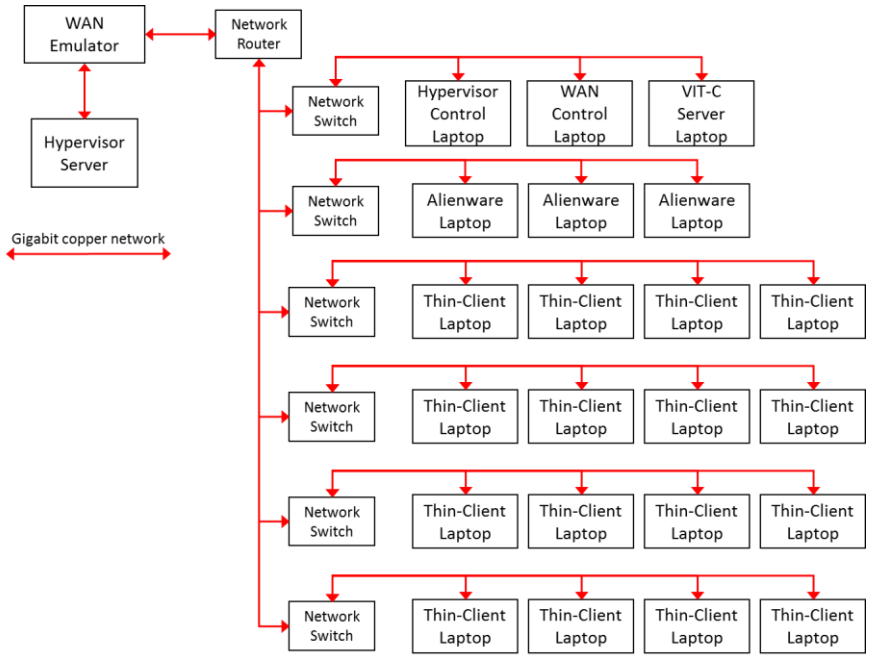

Figure 1. Experiment Network Configuration

Servers and Point-of-Need Clients

The following were the hypervisor server hardware specs used in the tests:

- Model: Cisco Systems UCSC-C240-M3S
- CPU: 2x Intel Xeon E5-2665, 16 cores (32 threads) @ 2.4 GHz
- Memory: 128 GB DDR3
- Storage: 1.8 TB hard disk
- Graphics: 2x vGPU capable graphics cards
- Network: Intel X540-AT2 (10 Gigabit)

The thin-clients were standard business class Dell Latitude E6510 laptops with Intel Core i7 CPUs and basic integrated graphics capability.

Wide Area Network Emulator

The team used a hardware based Wide Area Network (WAN) emulator that allowed the team to explicitly control the available bandwidth and latency on the experiment network. Unless the team was evaluating the effect of network impairments, the WAN emulator was set to pass network traffic through at the max transmission rate of the network router and switches. The router and switches in network were rated for a max transmission rate of 1 gigabit per second, so this is what the test team used for the WAN emulator. No additional/artificial latency was added to the network until the tests in Objective 6.

Performance Measurement and Data Gathering Software

To control the tests and capture all of the relevant metrics and performance data from each component, the test team developed a custom tool called Virtual IT Composition (VIT-C). VIT-C aggregated the APIs from the various software and hardware systems into a central endpoint, where data from each one could be visualized and recorded. During the tests, VIT-C recorded six metrics from the virtual machines (CPU usage, memory usage, network usage, GPU usage, desktop draw rate and desktop refresh rate). The same six metrics were also recorded from the laptop thin clients. CPU, memory and network measurements came from the Windows Management Instrumentation (WMI) API. GPU

utilization came from an undocumented function called D3DKMTQueryStatistics in the GDI32 API. Draw rate and refresh rate came from the Desktop Window Manager (DWM) API.

From the hypervisor, VIT-C recorded overall CPU, memory and network consumption using the hypervisor's API; GPU temperature, usage and power consumption using the graphics card API; finally, server temperatures, overall power consumption and fan speeds from the Cisco Integrated Management Controller (CIMC) via the IPMI interface. VIT-C recorded each of these metrics once per second and displayed the latest value on a central dashboard where they could be analyzed in real time during the test runs. At the end of each test, VIT-C saved the data into a CSV file, which was then imported into a spreadsheet for postmortem analysis.

In addition to gathering statistics, VIT-C also served as the master command and control console for each of the machines involved in the test. The team developed a simple user simulation bot that would send commands to the simulation software in lieu of actual users. This was accomplished by sending low-level DirectInput messages to the software – to the simulation, these messages are indistinguishable from an actual human user pressing keys on the keyboard and moving the mouse.

The user simulator/bot ran as a background process on each VM and listened for commands from the central VIT-C console, where it could be enabled and disabled at will. This capability was necessary to stimulate the simulation into constantly redrawing a new perspective and subjecting the hardware to high-load conditions; without user input, the simulation's screen would rarely change and consequently the hardware would be largely idle. As this test was focused on scalability at maximum load conditions, the bot provided a simple way to force each VM into high-load conditions without needing a large number of actual users present.

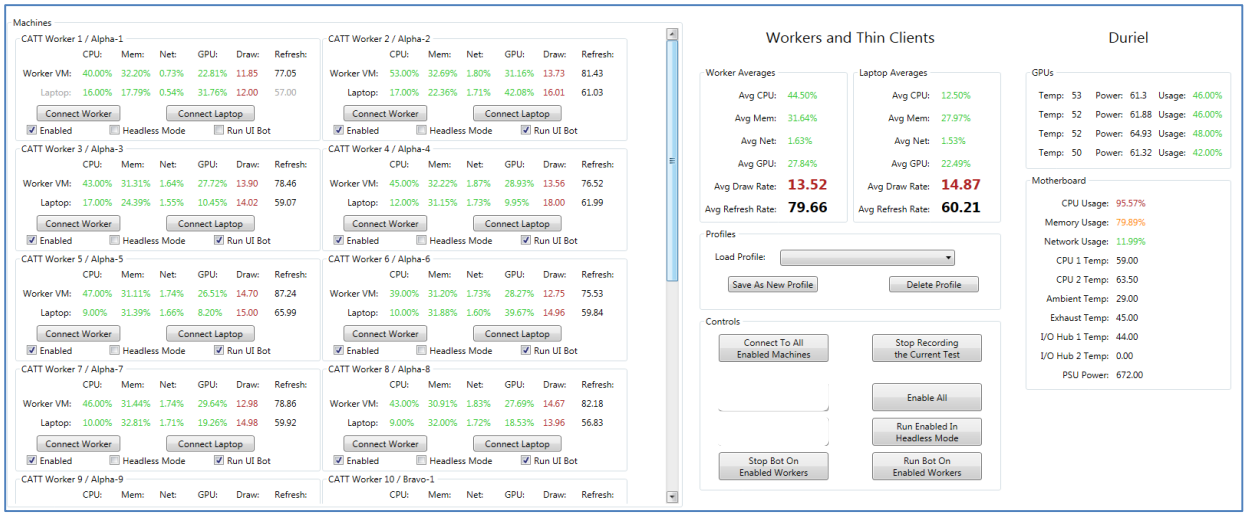

Figure 2. VIT-C Dashboard used during the experiment

Hypervisor Software

The team used a commonly available hypervisor software package for the experiment. The hypervisor utilized a client software program on the user machines to display the VM desktops from the hypervisor and allowed the users to interact directly the VMs. The actual name of the hypervisor is not used in this report as it is not the point of the experiment to specifically evaluate this individual software package, but the concept and capability as a whole.

VM and Point-of-Need Client Operating System

The team installed the Army Gold Master (AGM) edition of Microsoft Windows 7 for both the VMs and the client machines. PM ITE requested this version specifically to determine if there would be a problem with actual Army desktop installations. The test team used the AGM edition without making any configuration changes, and did not observe any access difficulties during the tests.

Simulation Software

The test team used a specific simulation software package as stipulated by PM ITE. The actual package name is not used in this report as it is not the point of the experiment to specifically evaluate this individual software package, but the concept and capability as a whole. Suffice it to say that the simulation software was definitely graphics intensive and required active user input to control.

OBJECTIVE 2: VALIDATE THE CONFIGURATION WILL WORK

The second objective was to validate that the simulation software will work in the simulated cloud/VM environment using vGPU technologies, and that the simulation software can be controlled using the point-of-need clients. The team conclusively verified and demonstrated that it is possible to run the graphically demanding simulation software from a cloud/VM environment using the vGPU technologies, and to remotely interact with the program via a client machine at the user point-of-need. There are caveats however, with regards to the system's performance under some conditions, and the configuration settings needed for the system to run efficiently. By the end of the allowed time to conduct the experiment, the **test** team conducted 44 formally measured test cases (TCs) and many more less formal and not measured tests to explore the performance of the overall capability. Continue reading in the following section for specifics regarding configuration settings.

OBJECTIVE 3: DETERMINE THE UPPER PERFORMANCE BOUND (UPB)

The third objective was to determine the UPB for the maximum number of VMs that can be run concurrently from the VM server and still maintain an acceptable level of performance. It is important to note that this objective was conducted without applying any network impairments (restrictions on maximum allowed bandwidth or forced latencies). Network impairments were applied and performance observed as part of Objective 6.

In order to determine the maximum number of concurrent VMs with acceptable performance, the team examined a number of different elements within the hypervisor settings for the VMs, including: number of assigned CPU cores and CPU topologies, RAM allocation, and vGPU configuration. Additionally, various configuration settings were changed and evaluated in the simulation software running in the VMs, most notably the displayed resolution and the distance viewable in the simulation. The team used both the objective draw rate measurement (in frames per second) within the VM and the subjective measure of how responsive the simulation "felt" at the client machine. To compare the draw rate performance from test to test, the testers evaluated the draw rate measurements via histogram charts and looked for those tests that had the greater number of incidents of higher draw rates.

The team conducted TCs that ranged from using 1 to 12 VMs. The team had originally estimated that it might be possible to run as many as 16 concurrent VMs, but the performance of 12 VMs was already very poor, so the team didn't test any more than 12 concurrent VMs. After all the testing was complete, the team determined that there were two viable choices for a UPB:

- Choice 1: Six VMs in high resolution (1920 x 1080) seemed to represent the best UPB for the higher resolution setting (as demonstrated in TC-28)
- Choice 2: Eight VMs in lower resolution (1440 x 900) represents a viable trade-off in resolution for additional VMs (as demonstrated in TC-30)

Figure 3 below shows the histogram for the VM draw rates for UPB Choice 1. Likewise, Figure 4 below shows the histogram for the VM draw rates for UPB Choice 2. The graphs depict the number of times a draw rate was measured from within the VMs according to the designated measurement brackets. Disregard the fact that the two tests have a dissimilar total number of measurements as the tests were different lengths in time and the total number of measurements is irrelevant. What is important to note is the relative distribution of measurements within the different measurement brackets. The test team characterized UPB Choice 1 as mostly acceptable, and UPB Choice 2 as somewhat acceptable. TCs with additional VMs did not perform as well and were not acceptable.

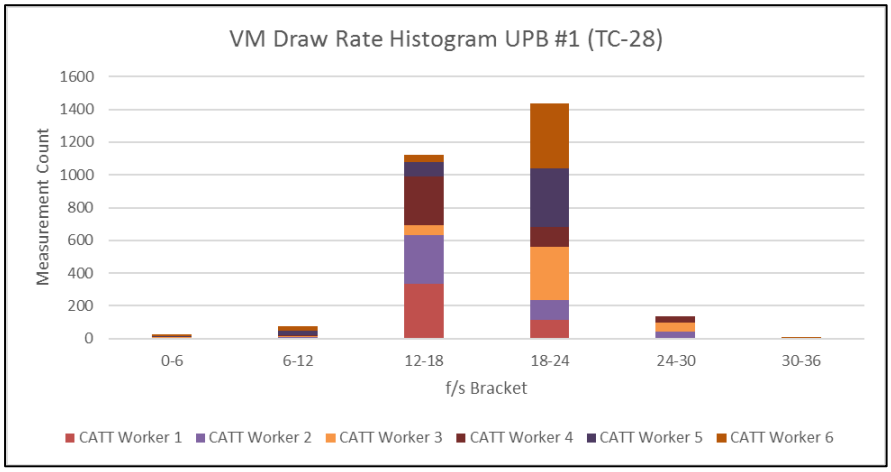

Figure 3. VM Draw Rate Histogram for UPB #1

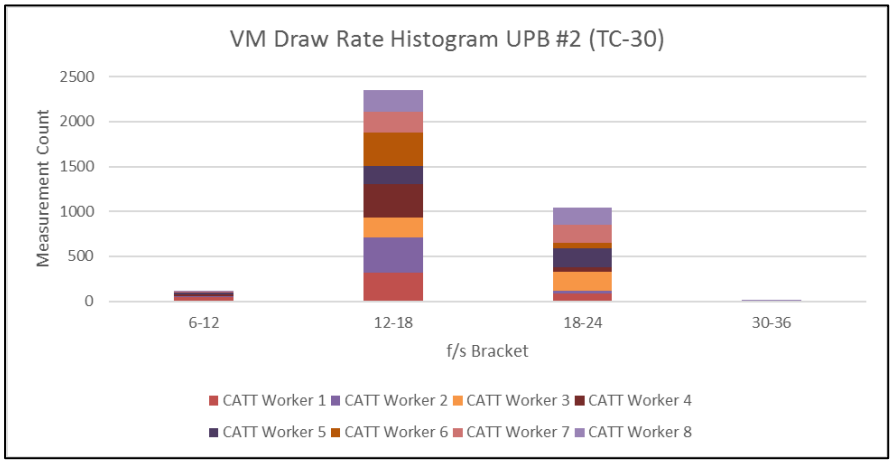

Figure 4. VM Draw Rate Histogram for UPB #2

Virtual CPUs Assigned

Hypervisors allocate the server's CPU cores as virtual CPUs (vCPUs) to the VMs. The particular server being used in the experiment had 2 physical CPU sockets with 16 individual cores each for an overall capacity of 32 individual cores and thus a total of 64 hyper-threaded cores. The test team determined that for this particular simulation software, it was sufficient to assign 6 vCPUs per VM. In the test cases, the software showed that it did not use multiple CPU cores very efficiently, and assigning more than 6 vCPUs did not demonstrate any clear performance increases.

System RAM Allocated

Hypervisors allocate a specified amount of the overall system RAM to each VM. Measurements of each VM's performance during the test cases indicated that each VM needed no more than approximately 4.2 GB RAM for both the guest operating system as well as the simulator software to function. Accordingly, the team allocated 6 GB RAM per VM and this was more than sufficient for all the tests. The server had a total of 128 GB RAM installed, and had plenty of extra capacity above what was required.

Graphics Card Configuration

Graphics cards that utilize vGPU technologies can be configured to emulate various levels of graphics card performance. Essentially, the system administrator can choose between options that divide the graphics card into as many as 16 vGPUs with sharing of the associated video RAM (VRAM). The obvious performance principle being

employed is that the more vGPUs that are created, the less processing power is assigned to each individual vGPU. Accordingly, the graphical rendering requirements of the simulation software have to be weighed against the number of VMs that need to run concurrently.

The simulation software required a minimum of 512MB of VRAM but recommended 1GB. The test team ran test cases with vGPU settings in both a 1GB and a 2GB VRAM configuration. The test data did not show a significant performance improvement when using the 2GB VRAM setting, hence the team settled on the 1GB VRAM setting as being the optimum balance between performance and number of vGPU/VM combinations supported. In the 1GB VRAM configuration, the graphics card would support 8 separate vGPUs. Given the server being used in the test contained 2 graphics cards; the server could then support a (potential) total of 16 vGPU/VM combinations.

Simulation Software Configuration for "Visibility"

The simulation software contains a configuration setting for "visibility distance" that is normally left in the default setting of 3000 meters. However, when the system was not performing as well as desired in the earlier TCs, the team explored what settings could be changed that would improve the simulation's performance but not significantly detract from the quality of the simulation. After some trial and error exploration, the team decided that the visibility distance could be reduced to 750 meters without significantly altering the usability of the simulation. Reducing this setting had a discernably positive effect on the simulation and system performance.

Simulation Software Configuration for Screen Resolution

PM ITE typically prefers to run simulation software in full high definition resolution of 1920x1080 pixels (or greater, if the software and hardware allow). However, as part of the exploratory tests, the team found that a reduction in resolution to 1440x900 pixels improved overall system performance such that additional (simultaneous) VMs could be added. Although the change was notable in the simulation, the consensus in the test team was that it did not significantly hamper the usability of the simulation.

OBJECTIVE 4: DETERMINE THE LIMITING FACTORS

The fourth objective was to determine the limiting factors in the experiment configuration, meaning, which system resource was exhausted first. Through all tests conducted, both formally measured and informal exploratory tests, the test team observed that the one system resource that ran out first was CPU capacity. This means that as CPU capacity neared 100%, the GPUs, system memory, and network bandwidth all had remaining capacity. Naturally, this observation is closely tied to the physical configuration of the system (see previous system hardware configuration detail) and the simulation software being used. Had we used another server chassis that allowed for additional CPUs, then perhaps the limiting factor might have been the vGPU capacity or perhaps memory. Likewise, using a different simulation program could have required differing amounts of system resources.

One notable observation from all the tests is that there appeared to be a correlation between the performance capacity of 32 CPU cores and that of a single graphics card. Meaning, in the server configuration with two graphics cards installed, as CPU capacity neared 100%, the overall GPU capacity was about 50% per card. In a reconfigured test when there was only one graphics card installed, tests consistently showed that as CPU capacity neared 100%, the overall GPU capacity also neared 100%. This data led us to the generalized conclusion that from a performance capacity planning perspective, 32 CPU cores and one vGPU capable card could roughly support six to eight graphically demanding VMs (depending on configuration settings as previously noted). Again, we must caveat this observation with the understanding that it is for this hardware and software configuration. Different configurations may have different results.

OBJECTIVE 5: DETERMINE THE REQUIRED NETWORK BANDWIDTH

The fifth objective was to determine the amount of network bandwidth required to run the system. In our test configurations, the bandwidth used was generally a function of:

(# of VMs) x (graphical density per VM) x (frequency of desktop update from server to client)

Naturally, this means that the observed network bandwidth can be positively or negatively affected by each of these factors. Indeed, the test team recorded the network bandwidth performance for all measured TCs, and noted a range of behaviors that make citing a specific bandwidth measurement somewhat problematic. It is certainly possible to cite a peak utilized bandwidth, but any such measurement must be clearly caveated that it is a peak, and not sustained, measurement. For example, consider Figure 5 below. This graph shows the network utilization over time, measured once per second, for each of the six VMs active in the TC for the UPB Choice #1.

As can be clearly seen, even though the VMs are all configured exactly the same for this TC, there is a significant variance in the network bandwidth utilization. The variance is a result of different levels of activity in each of the simulations occurring in the respective VMs. Basically, the more activity in a simulation that causes changes to what is displayed to the user, the more frequently those desktop changes need to be transmitted to the user's client machine, thus causing an increase in the amount of network bandwidth needed. Naturally, the converse is also true.

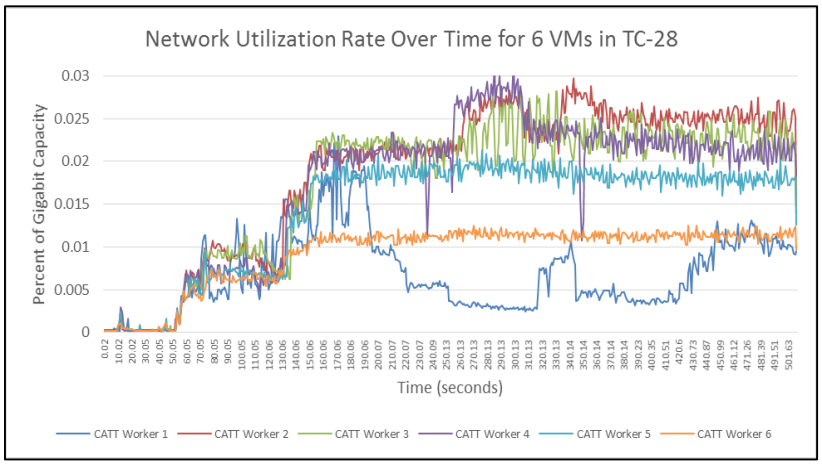

Figure 5. Network utilization over time for 6 VMs in UPB #1 (TC-28)

For the two UPB choices, the test team measured the peak utilized network bandwidth as follows:

- UPB Choice 1: Six VMs in high resolution (1920 x 1080): 105.8 megabits per second (Mbps)
- UPB Choice 2: Eight VMs in lower resolution (1440 x 900): 98 Mbps

Note that with UPB Choice 2, even though there were eight VMs in this TC, given that they were in a lower resolution made an overall difference in the peak network bandwidth. Note also that the peak bandwidth cited is for a "mixed case" of VMs performing differing levels of tasks, such as depicted in the graph in Figure 5. For a "max usage" measurement, we take the highest peak bandwidth encountered by any of the VMs and simply multiply that by the number of VMs used. In the case of UPB Choice #1 above, the max usage bandwidth would be approximately 30 Mbps x 6 = 180 Mbps. This measurement would approximate the needed bandwidth if all the VMs were performing at the same peak load simultaneously.

OBJECTIVE 6: EVALUATE THE EFFECT OF NETWORK RESTRICTIONS

The sixth objective was to apply network restrictions to the experiment network and evaluate the results. By restrictions, we specifically mean limitations to available network bandwidth as well as increased network latency, or "ping times". The test team applied the restrictions via the WAN emulator.

In the course of planning for the experiment, the team decided to test network restrictions by introducing increasing amounts of degradation in a step-wise fashion and then evaluate the effect on the system performance. In evaluating the effect on the simulation performance, the team placed more emphasis on the subjective evaluation aspect. The team thought that the subjective aspect was more relevant given that the effects of network restrictions are perceived by the user as increased lag in responsiveness and a decrease in graphics quality.

Both the subjective observations and objective data in the measured TCs clearly indicate that network bandwidth limitations and data transmission latency have detrimental effects on system performance acceptability. Of the two, transmission latency appears to have the greater effect. Not only does the latency make the system less responsive to user input, but it also induces a cascading negative effect to the client draw rate because of the apparent handshake process that occurs between the hypervisor and client components.

With regards specifically to latency, the system responsiveness was borderline unacceptable at 75 milliseconds (ms) latency in either resolution tested. That is to say, as the latency approaches upwards of 75 ms, the system responsiveness becomes less and less acceptable. Any more latency and the system responsiveness would not be acceptable. For the team's purposes, responsiveness was defined as the elapsed time between a mouse or keyboard input and the discernable effect in the simulation on the screen.

With regards specifically to bandwidth limitations, the hypervisor components appear to actively compensate for decreased network bandwidth. They appear to do so principally by decreasing the quality at which they encode the VM desktops as well as the rate they transmit desktop changes to the clients. The effect of this decreased quality is a noticeable pixelation and some tearing in the simulation graphics. The point at which the quality becomes unacceptable is subjective according to the user's expectations and ability to tolerate graphical aberrations.

Overall, the team's consensus is that whereas both network bandwidth degradation and increased transmission latency have a discernable impact on system performance acceptability, the key factor is the latency. Even with small amounts of latency, the system responsiveness appears to start to degrade rapidly, and this was more of an issue than decreased video quality. The team's conclusion was that whereas they could tolerate some loss of video quality, as long as the simulation was sufficiently responsive, then they could overlook the video quality. However, if the simulation was not responsive enough to be used, then that was considered a critical failure.

POST EXPERIMENT FOLLOW-ON ANALYSIS

Now that the research team knows that simulation/game streaming is feasible, the next step is to identify the ideal system and network configuration for simulation/game streaming. Because the simulation is sensitive to network latencies and bandwidth restrictions, the research team believes all simulation/game streaming should happen from the local processing node. A local processing node is essentially a local data center that supports a base or training facility. All further experiments will make the assumption that simulation/game streaming will occur from the local datacenter. The research team will determine if hardware-based video encoding will improve the frame rate of the simulation software. Hardware-based video encoding uses the GPU or special purpose hardware to encode the video stream. The hypothesis is that offloading video encoding to special purpose hardware will give the simulation more CPU cycles and thus improve the frame rate of the simulation. In addition to hardware-based video encoding, the research team wants to determine the required bandwidth for a large multiplayer simulation session. The hypothesis is that for large multiplayer sessions a 10 Gigabit network will be required.

SUMMARY

This paper describes the results of an experiment to host a graphically demanding simulation program within a simulated cloud environment utilizing hypervisor, VM and vGPU technologies. The observations and data collected during the experiment demonstrated that it is definitely feasible to host graphically intensive simulations from a cloud environment with vGPU technologies. There are trade-offs to consider however, in order to obtain the best balance between simulation performance and the maximum sustainable simulation/VM instances within a single hypervisor server. These trade-offs include the proper vGPU configuration, vCPU allocation, as well as various simulation configuration settings. The experiment showed that there is a general correlation in that 32 CPU cores and one vGPU enabled graphics card can support 6-8 simulation VMs, depending on how the simulation is configured. Network performance does matter, in that there must be sufficient bandwidth available and the simulation can be particularly sensitive to network latency times. The results of the experiment imply that whereas it is possible to host the simulation in a cloud environment, some simulations may be too sensitive to the network latencies involved thus implying that the simulation must be hosted in a local/on-site cloud as opposed to a cloud across the WAN.