Reducing Image Generator Footprint with Virtualization

Matthew Moy, Devin Fowler
RAVE Computer
Sterling Heights, MI
mmoy@rave.com, dfowler@rave.com

ABSTRACT

Today's high-fidelity training systems are unable to be efficiently deployed at the point of need due to the size, weight, and power of the equipment that is required. There are limitations with how much the form factor of a system can be reduced, particularly with the slow but steady rise in power requirements on many of the central processing unit (CPU) and graphics processing unit (GPU) architectures over the last several years. A new approach is necessary to reduce the footprint of these compute resources. In this paper we will demonstrate how we were able to reduce the footprint of our image generator (IG) systems by 50% using virtualization. This approach brings with it a unique set of challenges. If virtualized using the same approach as a typical server, challenges relating to rendering performance, hardware support, user perception, and network latency are inevitable. These challenges must be addressed to ensure the same quality of delivery with virtualization. This paper will outline an approach to optimize virtual machines for real-time rendering and provide a testing methodology to verify that they can provide the same experience as their physical counterparts.

This approach includes:
1. Identification of the components for the complete virtualized solution
2. A comparison of a physical versus virtualized IG
3. Measuring average framerate, latency, and utilization, as well as a user's performance completing a repeatable task within the simulation, to verify that virtualization does not have a negative impact on delivery
4. Example use cases that both support and exclude virtualization

The methodology put forth in this paper and supported by the aggregated data points will provide the audience with an option for delivering high quality immersive training without the prohibitive footprint required by current solutions.

ABOUT THE AUTHORS

Matt Moy has been a Systems Engineer with RAVE Computer for over 12 years. With his passion for technology and focus on real-time rendering systems he has supported multiple players in the training and simulation community via the development of innovative and purpose-built rendering systems. In addition, Matt's R&D work led to multiple successful demonstrations of virtualized image generators, development of advanced system testing methodologies, and methods for customer-specific system optimizations.

Devin Fowler is a Systems Engineer at RAVE Computer where he is part of the Product Development team and is responsible for the development and enablement of new solutions within the RAVE portfolio. Devin has created and implemented multiple different testing methodologies that are used to test the performance and reliability of new hardware.

Reducing Image Generator Footprint with Virtualization

Matthew Moy, Devin Fowler
RAVE Computer
Sterling Heights, MI
mmoy@rave.com, dfowler@rave.com

PURPOSE OF THE RESEARCH

The purpose of this research is to study the impact that virtualizing IGs has on the ability to deliver an immersive experience and support effective training. If we can demonstrate that virtual machines (VMs) are capable of delivering an immersive experience with a similar level of quality as a traditional IG system, the footprint of the compute resources can be reduced. In our example, the overall footprint of our four-user configuration was reduced by 50% as a result of consolidating and virtualizing hardware. The GPU server we are using in our example use cases replaces four physical IG servers, which means we have reduced the total number of systems required by 75%.

Our test bed includes a single 4U GPU server running four VMs that are intended to replace four physical 2U IG systems. The test bed also includes one of our widely used 2U IG system configurations with the same RTX A5000 GPU that we are using in each of our VMs. The 2U IG will be used as our baseline configuration. The Varjo Aero and the HTC VIVE Focus 3 virtual reality (VR) headsets were used for the tests that were performed. The hardware configurations and testing methodologies being discussed are also applicable to use cases with standard displays or projectors, but we chose to use VR headsets in our research since they tend to be the most demanding in terms of system performance, latency, and input/output (I/O).

EXPERIMENT METHODOLOGY & CONTEXT

In this section we will provide an overview of the methods and considerations used to determine the appropriate hardware and software configuration for a GPU server intended for real-time rendering, followed by the testing methodologies used in each of our example use cases.

Overview of the Virtualized IG Building Blocks

- GPU Server
- Hypervisor
- Remote Access Software
- Client Device (Optional)

GPU Server Configuration Methodology

CPU Selection

One of the biggest challenges when selecting CPUs for a server that is intended for virtualizing high fidelity real-time rendering workloads is determining the appropriate balance between clock-speed and the number of CPU cores. In general, real-time rendering applications tend to require relatively high clock-speeds. Supporting multiple VMs will require a large number of CPU cores. The challenge is that these two specifications are typically inversely proportional. Server CPUs generally have lower clock-speeds than desktop and workstation CPUs, and the clock-speed on CPUs typically drop as the core counts increase.

The first step is to determine the approximate clock-speed and number of cores that will be needed for each VM. In our testing we are using a baseline system configuration in our 2U physical IG that has been proven to work well with a wide range of applications used by many of our IG customers in the training and simulation industry. The CPU in this configuration is the Intel i7-10700k which has eight physical cores and a clock-speed range of 3.8 to 5.1GHz. This means we will need a minimum of 32 CPU cores and a base clock that is as close to 3.8GHz as possible to support four virtualized IGs. We anticipate that whichever CPU we select will run above the base clock

by some amount, but since the clock-speed increase can be difficult to predict in mixed workloads the base clock should be weighted more heavily than the boost clock when making this selection. The turbo or boost clocks are determined by a number of variables such as workload, thermals, utilization, power consumption, and voltage stability (AMD, 2022) (Intel, 2022). Since there are no CPUs that support a 32-core (or higher) configuration with a base clock as high as 3.8GHz available on the market today, we will have to accept a CPU model that is as close to 3.8GHz as possible. To simplify this selection, we have compiled a table of CPUs with at least 16-cores (for CPUs supporting dual socket configurations) and 32-cores (for CPUs limited to single socket configurations) with a base clock of 3.0GHz or higher. (see Table 1)

Table 1. Comparison of CPU Specifications

CPU Models	Cores	Base Clock	Boost Clock	Aggregate Clock	CPU Mark Score
Intel Core i7-10700k (Baseline)	**8**	**3.8**	**5.1**	**30.4**	**19,277**
Intel Xeon Scalable (Dual CPU)					
Gold 6346	16	3.1	3.6	49.6	37,606
Platinum 8354H	18	3.1	4.3	55.8	n/a
Gold 6354	18	3	3.6	54	41,434
Platinum 8360H	24	3	4.2	72	n/a
AMD Threadripper Pro (Single CPU)					
3975WX	32	3.5	4.2	112	63,056
AMD EPYC (Single or Dual CPU)					
73F3	16	3.5	4	56	46,085
7343	16	3.2	3.9	51.2	45,882
7373X	16	3.05	3.8	48.8	n/a
7313	16	3	3.7	48	40,575
74F3	24	3.2	4	76.8	60,948

For the applications we will be running we are targeting a clock-speed of 3.5GHz or higher. Even though many of the CPUs in the chart above have turbo or boost clocks that exceed 3.5GHz, it is difficult to determine the level of turbo or boost performance we can expect within the virtual machines.
The 3975WX was determined to be the best fit due to the high base clock. There are other CPUs in the chart that may perform at a similar level if the turbo or boost clocks are capable of maintaining a clock-speed above the base clock, but the result would be less predictable, and these alternative options are substantially higher in cost.

Since the Threadripper Pro platform is currently limited to single socket configurations, our host system will be limited to 32-cores / 64-threads. This means we can dedicate six to eight physical cores to each virtual machine and the clock-speeds will run at 3.5GHz or higher. Currently there are not any proprietary or large form factor motherboards available on the Threadripper Pro platform which means we will be limited to a 7-slot ATX or SSI-EEB motherboard. This means we can support up to four high-end GPUs per server.

GPU Selection
For high-end configurations, the GPU model that is selected should be similar in performance to whatever is being used in the corresponding physical IG, but it is important to ensure the model selected is compatible with whichever hypervisor that will be used. If the physical IG is using a GPU that does not support virtualization, you can compare specifications to find a supported card that offers similar performance. Some of the key specifications to compare include core configuration, clock-speed, floating-point performance, as well as scores from common benchmark applications such as 3DMark. For example, if you have a physical IG that is configured with a NVIDIA RTX 3090

GPU, this is an example of a GPU model that does not have official support for virtualization. However, NVIDIA does support virtualization on the RTX A6000 which has very similar specifications to the RTX 3090. (see Table 2) Table 2 also includes a similar comparison with AMD GPUs. Since the only current model AMD GPUs on the VMware compatibility list are data center GPUs, these cards do not have video outputs so they would only be applicable to configurations using remotely connected client devices. The Instinct MI250 GPU in the table below has more than double the compute power of the RX 6900XT, so this model should be able to support at least two VMs per GPU when trying to replicate the performance of the RX 6900XT. Keep in mind that these examples are based on the VMware hardware compatibility list; supported GPU models will vary depending on the hypervisor being used.

Table 2. Comparison of GPU Specifications

GPU Model	Cores	Base Clock (MHz)	Boost Clock (MHz)	Memory	FP32 (TFLOPS)
RTX 3090	10496	1395	1695	24GB	35.58
RTX A6000	10752	1410	1800	48GB	38.7
GPU Model	**Cores**	**Base Clock (MHz)**	**Boost Clock (MHz)**	**Memory**	**FP32 (TFLOPS)**
Radeon RX 6900XT	5120	1825	2250	16GB	23.04
Instinct MI210	6656	1000	1700	64GB	22.63
Instinct MI250	13312	1000	1700	128GB	45.3

Use cases that can accept a low to mid-range GPU have more flexibility in GPU selection. If the application only demands a fraction of the resources from a high-end GPU, the GPU itself can be virtualized and distributed across multiple VMs using a technique known as 'time slicing'. This is where some of the largest gains in efficiency are possible since the GPU resources become more elastic and reconfigurable which results in a substantial reduction in the total number of GPUs that are required. The GPU resources are allocated by selecting from various 'GPU Profiles'. If the profile that was selected cannot provide sufficient GPU performance, the VM can be reconfigured in a matter of seconds and additional resources can be allocated. However, this approach does come with some limitations. When the GPU is time sliced you cannot connect displays or headsets directly to the outputs on the GPU. Instead, the VM must be accessed over a network with a client device and 3d accelerated remote access software. There are also additional licensing costs associated with time-slicing GPUs as well as additional requirements relating to GPU and hypervisor support. As a result, leveraging this time-slicing to improve efficiency and lower cost is partially dependent on the scale of the solution.

Hypervisor

There are several hypervisor and operating system (OS) options available with GPU support. The example use cases in this paper are running on the VMware ESXi hypervisor. Citrix Xen Server, Red Hat Enterprise Linux KVM, and Microsoft Hyper-V are some of the other popular choices for virtualizing 3D accelerated workloads. Support for PCI-Express passthrough is required for dedicating an entire GPU to a virtual machine, as done in our example use cases. For configurations where each GPU is shared across multiple VMs, the hypervisor must support that specific feature set on whichever GPU is being used. For NVIDIA GPUs this feature is referred to as vGPU (NVIDIA® Virtual GPU (vGPU) Software Documentation, 2022). For AMD GPUs this feature is referred to as Virtual Functions or VF (AMD, 2017).

Remote Access Software

Remote access software is used to provide access to the VM over a network. Some important considerations when selecting remote access software include support for the required display types and resolutions, number of displays, support for peripherals, and the types of client devices that are supported. If you are using a client device to access the VMs, the remote access software will need to support hardware-based video encoding/decoding. However, if you are using a VR headset most of the common options for remote access software and client devices will not work properly. This is being addressed with the development of many new applications which are currently emerging to

support remotely rendered virtual reality (VR) and mixed reality (XR) such as NVIDIA's CloudXR, Varjo Reality Cloud, and many others. For configurations where the displays and peripherals are directly connected to the GPU server, 3d accelerated remote access software is not needed. Instead, basic remote access software such as VNC can be used for system administration.

Client Devices (Optional)
Client devices can either be dedicated appliances, existing PCs running remote access client software, or a VR headset capable of running remote access software such as the Meta Quest 2 or the HTC Focus 3.
In order to maintain low latency and a high level of responsiveness it is important that the client device supports hardware-based video decoding, as well as support for the required connections for peripherals and displays.

Connectivity Methods

There are two different approaches to connectivity that can be taken when leveraging virtualized IGs in a simulator.

- Direct Connect: Displays and peripherals can be directly connected to the server. This option is represented in both of our example use cases. (see Figure 1)
- Remote Connect: Displays and peripherals connected to a client device, with the server streaming the rendered scenes over a network. (see Figure 2)

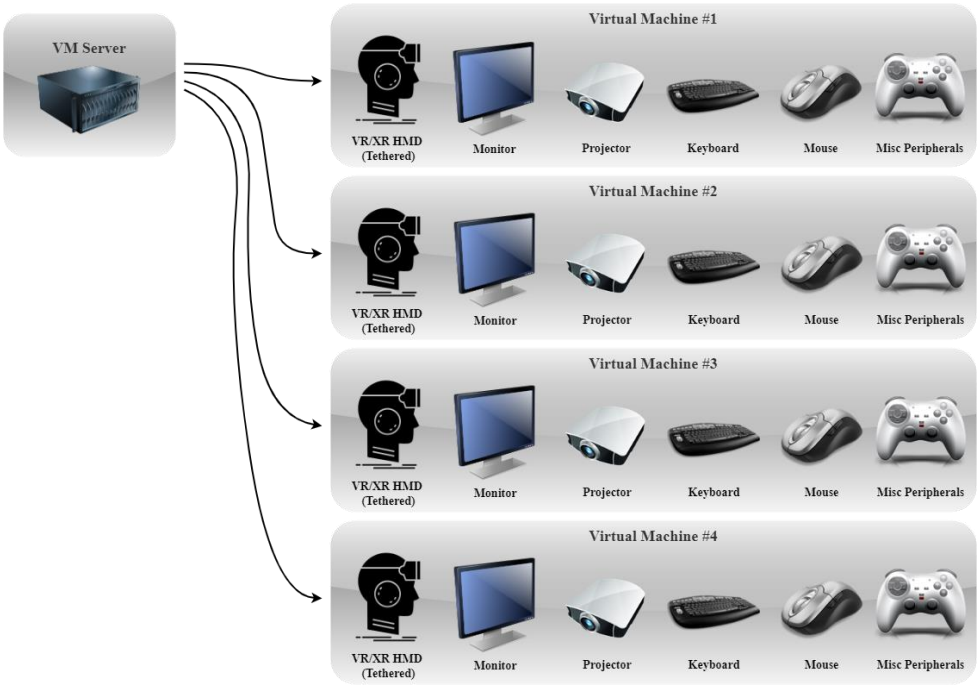

Figure 1. Direct Connect Diagram

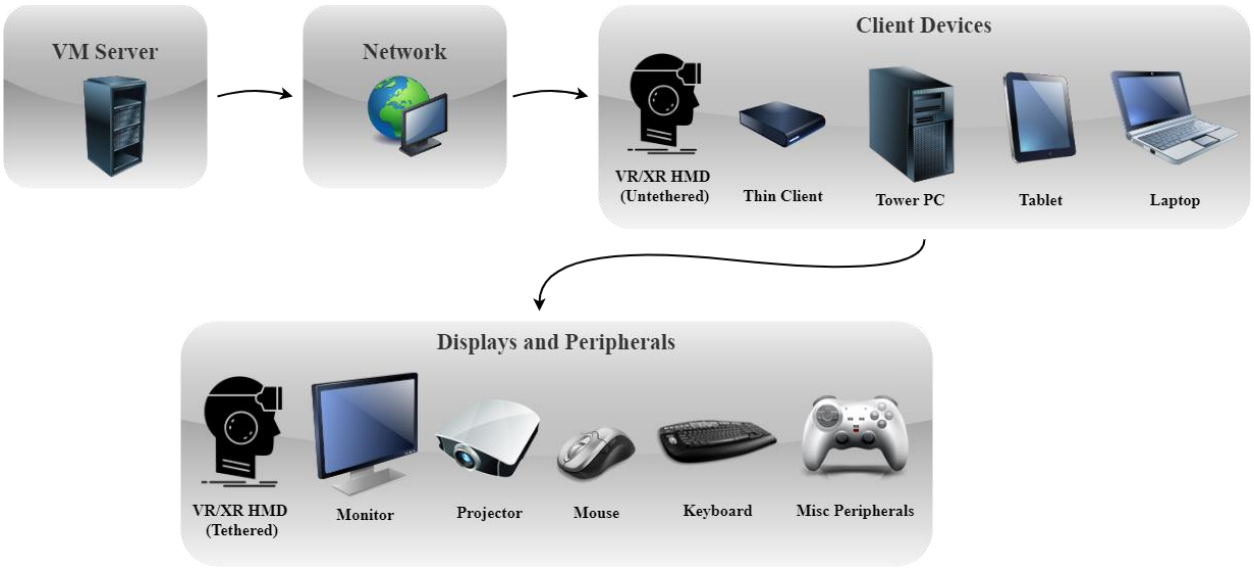

Figure 2. Remote Connect Diagram

Direct Connect

The first approach has the advantage of being less sensitive to network latency and it also requires less equipment, however the tradeoff is that it requires the GPU server to be in close proximity to the simulator. This approach also requires additional USB controllers beyond what the motherboard can provide in order to deliver sufficient bandwidth for HMDs, input devices and other peripherals. HMDs in particular can be very bandwidth sensitive and both HMDs and peripherals can be highly sensitive to latency. This means that a dedicated USB controller for each virtual machine will be needed. Fortunately, there are a small number of companies manufacturing specialized PCI-Express USB controllers that have multiple independent controller chips. To increase the density in our GPU server configuration, we can leverage a single PCI-Express USB card with multiple controllers and dedicate each USB chipset on the card directly to a VM using PCI-Express pass-through with a supported hypervisor. With the hardware configuration that was selected for our example use cases, the four GPUs will occupy all available PCI-Express slots, so as a workaround one of the SFF-8643 ports on the motherboard can be utilized to provide four PCI-Express lanes to a riser board that the USB card can be installed into. These SFF-8643 ports are typically used for NVMe storage devices, but since these ports use PCI-Express signaling we can use them in this unconventional way to provide the additional PCI-Express slot that is needed. However, this will require a mounting solution for the USB controller card since it cannot be mounted in the typical expansion slot location due to the GPUs occupying that space.

Remote Connect

The second approach has advantages relating to accessibility and management. With the compute resources available as a network resource the simulators can be mobile, and the compute resources can be reconfigured in a more granular way using the time-slicing technique. Since the compute resources do not require close proximity to the simulators, they can be located in a server closet or datacenter which has potential to simplify management and allows the systems to be physically secured. The tradeoff in this case is that additional equipment is required as well as additional network bandwidth and lower network latency. This approach will also require additional software licensing for remote access software. Since simulators require high fidelity graphics with low latency, the remote access software that is selected must be optimized for 3d accelerated workflows and it must utilize hardware-based video encoding and decoding in order to maintain an acceptable level of latency. If the encoding or decoding is done at a software level, the latency and CPU overhead will be too high to achieve an immersive experience for effective training.

Testing Methodology

Due to the vast number of variables within hardware and software that can impact the quality of the training and level of immersion, we have developed multiple tests to cover a range of use cases with the goal of addressing as many of these variables as possible. Scripted benchmarks are used to measure potential differences in compute and rendering capability as well as the end-to-end latency of the rendered images outputted to the head mounted display (HMD) or monitor.
User experience testing is used to determine the potential impact virtualizing IGs has on the user's ability to perform a highly skilled task in a simulated environment. User experience testing is critical since there is potential for issues that cannot be easily discovered via metrics from benchmarking and latency testing. In this case the user experience testing represents the proof-of-concept testing that would normally be done in a production environment when rolling out a virtualized solution for 3d accelerated or real-time rendering workflows. This combination of benchmarking and user experience / proof of concept testing makes the inevitable tuning and reallocating of resources on the GPU server more effective and less disruptive since it can be done on a smaller scale before the solution is rolled out to the entire user base.

Test Bed Configuration
The 2U IG hardware configuration was based on common configurations we use to support many of our IG customers. The VM configuration was determined using the methodology outlined earlier in the 'GPU Server Configuration Methodology' section. The following image provides a side-by-side comparison of the two configurations we will be comparing in the following test examples. (See Table 3)

Table 3. Test Bed Configurations

	2U	VM
CPU	8-core 2.9-4.8GHz	8-core 3.5-4.2GHz
Memory	16GB DDR4 2933MHz	16GB DDR4 3200MHz
GPU	NVIDIA RTX A5000	
Storage	1TB NVMe	1TB NVMe (shared)
Network	1GbE (Intel i210)	1GbE (VMware vNIC)
OS	Windows 10 Pro	
HMD #1	Varjo Aero	
HMD #2	HTC VIVE Focus 3	

Example #1: Benchmark Testing with Digital Combat Simulator
The benchmarking methodology should resemble the real-world workload as closely as possible. In our first example we will use a flight simulator application called 'Digital Combat Simulator'. We have created a test mission with a series of waypoints and a flight pattern that includes many of the more detailed areas of the environment. Autopilot is used to ensure that the scenes being rendered are as consistent as possible with repeated runs. Since we are using a VR headset, we also have to incorporate head motion in our test. In order to do this in a way that is consistent with each run, we have set up a manikin head that has been mounted to a motorized pan and tilt head. The pan and tilt head we chose was designed to be used with cameras, but it also happens to work perfectly for this purpose. This allows us to mimic the head movements one would normally make while flying the aircraft but in a way that is highly consistent between test runs.

Now that a method for rendering a consistent scene has been established the next step is to measure the systems performance. The best metric for measuring system performance in a real-time rendering application like 'Digital Combat Simulator World' is frame time. A more common metric is frame rate; however, this is a less accurate way to measure performance because it assumes all frames were displayed for a similar duration. For example, if your target frame rate is 90 FPS or frames per second, that could potentially mean that a single frame was displayed for .9 seconds and the other 89 were displayed in that final .1 seconds. This would create a noticeable stutter or glitch in the rendered scene. Since we are doing our testing with VR headsets this dichotomy is even more significant due to

the fact that VR headsets can cause motion sickness if there is noticeable stuttering or glitches in the scene. To measure frame times, we are using an application called 'FCAT VR'. In effort to keep the test runs as consistent as possible, the logging is initiated as soon as the throttle is applied on the runway. The logging duration can be set in the FCAT VR application which in our example has been set to 3 minutes and 30 seconds for the scene we are testing. Since the compute resources in the GPU server are being distributed and shared between multiple VMs, it is important to have the same number of VMs under load on the GPU server as the production environment would demand. In our example, that would mean running workloads on all four VMs simultaneously. While it would be ideal to run the same exact workload on each VM, as long as we ensure that the additional VMs are under a similar level of utilization as the VM being measured the test will produce meaningful data. In our example, we ran a benchmark called Unigen Heaven in a loop on each of the remaining VMs during the testing.

We must run the test several times to ensure that it can produce consistent results. Once it has determined that the testing methodology is capable of producing consistent results with an acceptable margin of error, iterative testing can now be started. Ideally these repeated runs should produce results within 5% of each other, but the variance should be reduced as much as possible. This will provide a path for tuning and optimizing the VM configurations through a series of test cycles using the established methodology to measure and compare performance between changes.

Once the VM configurations have been fully optimized, the same tests can be performed on the 2U physical IG system. This will allow us to compare performance with the virtualized IG. While there is some level of tuning and optimization that can be done with the 2U physical IG, a virtual machine configuration has an order of magnitude more variables and options to configure. For this reason, the research and methodology in this paper is focused on configuring and optimizing the VMs. If the results show that the average scene render time is within 5% between the two configurations, we will consider the VM to have met our performance target.

Example #2: User Experience Testing with Asseto Corsa

In our second example we are using a racing simulator application called 'Asseto Corsa'. Since there is some potential for issues relating to performance and usability that may not be captured during the benchmark testing, it is best practice to include some form of user testing or proof of concept testing when configuring virtual machines to replace physical IGs. This is also the standard practice among organizations rolling out virtual desktop solutions (VDI) to users of 3d accelerated applications. Even if performance and latency targets have been met, it is still critical that the solution can provide the same look and feel that users are accustomed to.

In effort to expand upon the common methods of proof of concept / user experience testing we have incorporated a method to measure the user's performance while exercising skills they've learned on the simulator. To do this we had each of the two drivers complete six runs on a racetrack with each run including an out-lap, warm-up lap, and a hot lap. We will average the hot-lap times to compare the runs between the 2U IG and the VM. Both drivers were already familiar with the track that was selected and were confirmed to have highly consistent lap times from run to run in our preliminary testing. This consistency can be further verified by looking at the lap-time range from the three runs on each system type. A curtain has been set up between the operator station and the simulator so the simulator can be switched between the virtualized IG and the 2U physical IG at random between runs without the driver's knowledge of which system they are connected to. Each driver will also complete a user experience survey at random points between runs to provide their subjective feedback on the overall experience. If the virtualized IG can provide an immersive experience similar to the 2U physical IG, the drivers should have similar lap times on both systems. If the drivers report issues during the survey or if their lap times on the virtualized IG are slower by an amount greater than the range between runs from each respective system, there is likely some form of performance issue or glitch that we did not detect during benchmark testing.

Example #3: Latency Testing

The third example provides a method for measuring end-to-end latency on rendering systems. This can be accomplished with an application from NVIDIA called 'Latency Display Analysis Tool' (LDAT). The LDAT application utilizes a hardware device equipped with a luminance sensor to accurately measure the motion-to-photon latency in a rendering application. Even though this is an NVIDIA application, it can be used with GPUs from all vendors. In our example, we will use LDAT's included 'latency test' application to render the output that will be measured. This allows us to compare end-to-end latency on the 2U and VM configurations so we can determine the impact virtualization has on latency.

The first test was performed on the Varjo Aero and represents the 'Direct Connect' configuration that was used in both the benchmark and user experience testing examples. The second test was performed on the HTC VIVE Focus 3 and represents the 'Remote Connect' configuration. For this test, NVIDIA CloudXR was used as the remote access software to allow the HTC VIVE Focus 3 to run as a client device which connects to the 2U or VM over a wireless network. This test will allow us to compare the combined impact of virtualization and remote rendering on end-to-end latency.

RESULTS

Example #1 Results: Digital Combat Simulator Benchmarks

In our first example use case, the average scene render time between the virtualized IG and the physical IG was within 3.87%, favoring the 2U physical IG. While the 2U physical IG had a slight performance advantage, the results fall within our target of 5% and the difference is not likely large enough to have a meaningful impact on the user experience. (See Table 4 and Table 5)

Table 4. Digital Combat Simulator: - FCAT VR Benchmark Results from ALL Test Runs

	Average Scene Render Time (ms)	Average Framerate (FPS)	Average Total GPU Render Time (ms)	Compositor GPU Render Time (ms)	Compositor CPU Render Time (ms)
2U: Run 1	15.63805019	63.95	15.85051423	0.01246399	0.212680178
2U: Run 2	15.54912856	64.31	15.76129283	0.012164208	0.20277965
2U: Run 3	15.63526599	63.96	15.84745368	0.012187619	0.205496072
VM: Run 1	16.13772656	61.97	16.3523596	0.014632998	0.313713503
VM: Run 2	16.19452949	61.75	16.40913277	0.014603217	0.307264038
VM: Run 3	16.29992854	61.35	16.51451068	0.014582019	0.313731008

Table 5. Digital Combat Simulator: FCAT VR Benchmark Results Comparison

	Average Scene Render Time (ms)	Average Framerate (FPS)	Average Total GPU Render Time (ms)	Compositor GPU Render Time (ms)	Compositor CPU Render Time (ms)
2U IG (Baseline)	15.61	64.07	15.82	0.0123	0.2070
Virtualized IG	16.21	61.69	16.43	0.0146	0.3116
% From Baseline	-3.87%	3.72%	-3.83%	-19.02%	-50.53%

Example #2 Results: Asseto Corsa User Experience Testing

In our second example use case where we tested the user experience by the participants performance while carrying out the highly skilled task of completing laps as fast as possible on a racetrack, we found that both drivers performed consistently regardless of which system they were connected to. Driver #1 (Devin Fowler) had range of 1.492 seconds on the 2U and 0.999 seconds on the VM. Driver #2 (Jakob Edringer) had a range of 1.126 seconds on the 2U and 1.133 seconds on the VM. Since the variance from run to run exceeds the difference between the average runs on the 2U IG vs the virtual IG systems, we can consider the results within the margin of error. Devin Fowler had the largest disparity between the 2U and VM systems of 1.267 seconds, however this still falls within the normal variance we would expect between runs which was recorded to be as high as 1.492 seconds in Devin's case. (See Table 6 and Table 7)

Table 6. Lap Times from User Experience Skill Test

Driver: Devin Fowler	Out Lap	Warm-up	Hot Lap		Driver: Jakob Edringer	Out Lap	Warm-up	Hot Lap
2U: Run 1	01:35.108	01:12.048	01:11.810		**2U: Run 1**	01:25.847	01:28.612	01:24.290
2U: Run 2	01:32.448	01:11.439	01:11.793		**2U: Run 2**	01:23.925	01:23.106	01:22.961
2U: Run 3	01:13.640	01:12.637	01:11.520		**2U: Run 3**	01:24.665	01:23.424	01:23.177
2U: Run 4	01:12.283	01:12.154	01:12.771		**2U: Run 4**	01:24.292	01:24.664	01:22.958
2U: Run 5	01:12.802	01:24.101	01:11.859		**2U: Run 5**	01:23.538	01:23.213	01:23.473
2U: Range	00:22.825	00:12.662	00:01.251		**2U: Range**	00:02.309	00:05.506	00:01.332
2U: Average	01:21.256	01:14.476	01:11.951		**2U: Average**	01:24.453	01:24.604	01:23.372
VM: Run 1	01:13.880	01:26.069	01:12.556		**VM: Run 1**	01:24.279	01:23.016	01:22.088
VM: Run 2	01:20.009	01:11.849	01:11.372		**VM: Run 2**	01:25.163	01:23.396	01:23.217
VM: Run 3	01:11.366	01:10.058	01:10.823		**VM: Run 3**	01:23.430	01:22.250	01:23.126
VM: Run 4	01:10.792	01:20.884	01:11.798		**VM: Run 4**	01:23.791	01:22.094	01:23.005
VM: Run 5	01:11.919	01:23.214	01:10.928		**VM: Run 5**	01:23.600	01:22.352	01:22.856
VM: Range	00:09.217	00:16.011	00:01.733		**VM: Range**	00:01.733	00:01.302	00:01.129
VM: Average	01:13.593	01:18.415	01:11.495		**VM: Average**	01:24.053	01:22.622	01:22.858

Table 7. Final Results of User Experience Skill Test

Average Hot Lap Time	2U IG (Baseline)	Virtual IG	Change
Devin Fowler	01:11.951	01:11.495	00:00.456
Jakob Edringer	01:23.372	01:22.858	00:00.514

Both drivers completed surveys after completing runs on the 2U physical IG as well as the virtualized IG and they both had 100% consistent responses following the runs on each system type. The user experience survey included the following questions:

1. Did you experience any level of motion sickness or nausea after completing the current test phase?
    i. (0 = none, 10 = extreme discomfort)
2. Did you experience any fatigue or tiredness after completing the current test phase?
    i. (0 = none, 10 = extreme discomfort)
3. Did you experience any eye strain or eye fatigue after completing the current test phase?
    i. (0 = none, 10 = extreme discomfort)
4. Did you experience any stuttering or glitches during the simulation?
    i. (0 = none, 10 = constant stuttering or glitches)
5. Was there any perceivable latency or delay with your input on the steering wheel or pedal peripherals?
    i. (0 = none, 10 = severe input delay)
6. Was there any perceivable latency or delay with motion tracking when moving your head position?
    i. (0 = none, 10 = severe delay when moving head)
7. Did the overall system performance limit your capabilities within the simulation?

   i. (0 = not at all, 10 = severely limited user performance)
8. How accurately do you feel the simulation represented the real-world experience?
   i. (0 = highly inaccurate, 10 = indistinguishable from real-world)
9. How did you feel in general after completing the current test phase?

Both participants answered '0' on questions one through seven each time they were surveyed. Both participants responded to the eighth question with a rating of 7/10. After completing each test phase, both users reported that they felt good and were comfortable continuing. They enjoyed the simulation experience and both participants felt the experience was the same as we randomly switched between the two hardware configurations.

Example #3: End-to-end Latency Test

Based on the results that were recorded using NVIDIA LDAT, we found that virtualization did not have a significant impact on end-to-end latency. When remote rendering is introduced, the increase in latency is more pronounced. In this case the increase is small enough to still allow for a good user experience, however this metric will be highly dependent on the latency of the network that the server and client devices are running on.

Table 8. End-to-end Latency Measurements from NVIDIA LDAT

	Direct Connect	Remote Connect (NVIDIA CloudXR)
2U	26.5ms	55.8ms
VM	26.6ms	55.9ms

CONCLUSION

The benchmarking results in our first example use case show that the average scene render time on the physical 2U system was 3.87% faster than the virtualized IG, which meets our goal of maintaining less than 5% variance. With additional tuning and optimizations to the server and VM configuration it is plausible that this disparity could be reduced even further. We expect the challenges associated with selecting a CPU with sufficient clock-speeds to become easier with future CPU releases, as recent trends have shown that while mainstream CPU platforms have started to reach a plateau in terms of clock-speed, server CPUs have continued to see slow but steady gains in both base and boost clock-speeds as new models are introduced. In the second example use case the users were not able to determine which system they were using, and the system type had no measurable impact on their ability to perform at their expected level. While both users performed slightly better on the Virtualized IG, the opposite was true in our preliminary test runs. However, the small difference in both cases was within the margin of error as determined by the hot-lap ranges on each system type. The third example highlights the extremely small impact virtualization has on end-to-end latency. When remote rendering is introduced to virtualized or non-virtualized systems there is a larger increase in end-to-end latency. We do not expect this increase to be significant enough to impact user performance or user experience but further research with remote rendering use cases will help us determine the point at which increases to end-to-end latency start to negatively impact the immersive experience.

Based on the findings found during the experimentation process for this paper, the results support the theory that virtualized image generators can be and are a suitable choice for replacing at least some physical options. The methods offered for configuring an optimized GPU server for virtualization and the processes for accurately measuring key metrics can be applied to a range of use cases, however our testing was primarily focused on IGs that are directly connected to the displays and peripherals. In our follow-on research we will take a deeper dive into testing configurations where the rendering is being done remotely over a network. For VR/XR applications the options for remote access software are somewhat limited at this time, but there is a large amount of development currently taking place in this area with many new solutions on the horizon, as well as frequent improvements and advancements to current offerings such as NVIDIA's CloudXR. Future research will focus on addressing some of the challenges and limitations associated with remote rendering such as application support, peripheral support, connectivity, comparison of client devices, network throughput and network latency.

REFERENCES

Ali, Q., Muirhead, T., Hsu, J., James , Z., & Hoflich, B. (2021). *Performance Optimizations in VMware vSphere 7.0 U2 CPU Schedule for AMD EPYC Processors.* VMware.

AMD. (2017). *Consistency and Security: AMD's approach to GPU virtualization.* Retrieved from AMD.com: https://www.amd.com/system/files/documents/gpu-consistency-security-whitepaper.pdf

AMD. (2022). *Turbo Core Technology*. Retrieved from amd.com: https://www.amd.com/en/technologies/turbo-core

Intel. (2022). *Turbo Boost Max Technology 3.0.* Retrieved from intel.com: https://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-max-technology.html

NVIDIA. (2022). *GEFORCE RTX 3090 FAMILY.* Retrieved from NVIDIA.com: https://www.nvidia.com/en-gb/geforce/graphics-cards/30-series/rtx-3090-3090ti/#specs

NVIDIA. (2022, March 02). *Virtual GPU Software Supported Products.* Retrieved from Virtual GPU Software Documentation: https://docs.nvidia.com/grid/latest/product-support-matrix/index.html

NVIDIA® Virtual GPU (vGPU) Software Documentation. (2022, 02 02). Retrieved from docs.nvidia.com: https://docs.nvidia.com/grid/index.html

PassMark Software. (2022, April). *CPU Benchmarks*. Retrieved from cpubenchmark.net: https://www.cpubenchmark.net/cpu_list.php

Peak, M. (2017, 05 05). *Virtual Reality Survey*. Retrieved from NC State University Libraries: https://www.lib.ncsu.edu/projects/virtual-reality-survey

Spurijt, R., Brinkhoff, C., & Plettenberg, M. (2020). *End User Computing State of the Union 2020.* vdilikeapro.com.