TECHNICAL WHITE PAPER
October 2024

Improving VDI Workload Consolidation with VMware vSGA and Intel Data Center GPU Flex Series

vmware®
by Broadcom

Contents

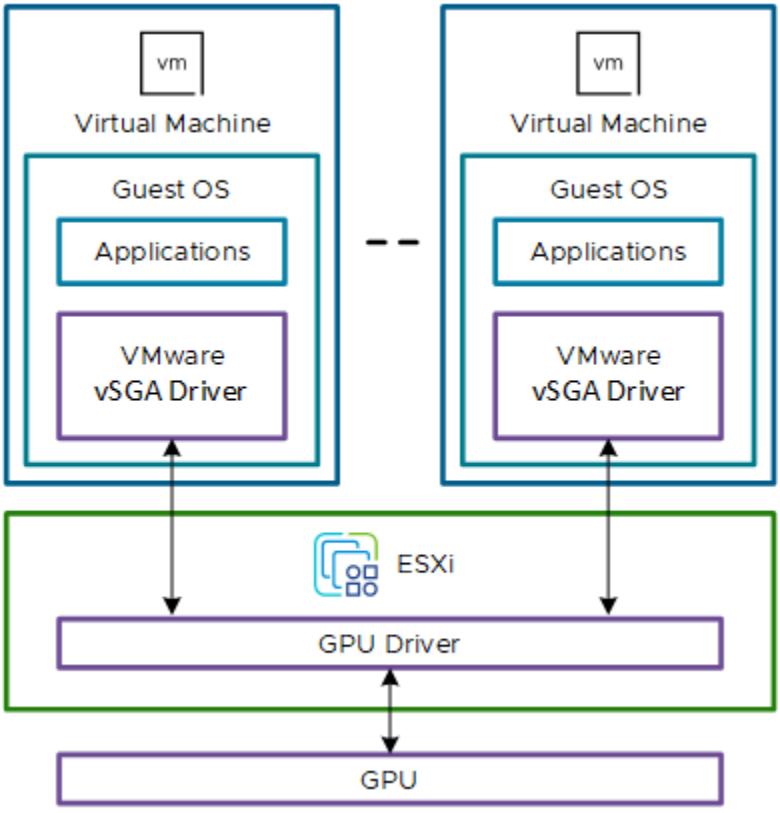
Executive summary

VMware® vSGA with Intel® Data Center GPU Flex Series offers a scalable, high-throughput, shared 3D graphics stack on VMware® ESXi™. We ran tests using Microsoft Office, Adobe Acrobat, WebGL, Web video, and CAD viewer applications to demonstrate that vSGA's performance (measured as user experience) is better than a CPU-only 3D graphics stack and comparable to the performance of SR-IOV on Intel® Data Center GPU Flex 140.

Introduction

VMware virtual shared graphics acceleration (vSGA) is a feature of VMware vSphere® and VMware Cloud Foundation® that enables sharing a physical GPU across multiple virtual desktops to accelerate 3D graphics applications in Windows and Linux VMs. vSGA is designed to improve processing speed when the graphics load is low to moderate; that is, the applications don't need to render complex models. Figure 1 shows the high-level architecture of the vSGA solution for VDI workloads.

Figure 1. vSGA architecture



Advantages of vSGA

vSGA results in enhanced management, streamlined consolidation, and workload efficiency. Because it is a feature of vSphere, you can:

- Enjoy lower TCO with 2x the consolidation of VMs on an ESXi host with vSphere/vSGA vs SR-IOV/passthrough.
- Share a single graphics card among multiple VMs with vSphere's ability to consolidate VMs on a single ESXi host and dynamically grow or shrink the number of VMs per host (elasticity).
- Accelerate hardware for VMs with different performance requirements on a single GPU.
- Suspend and resume VMs.
- Migrate VMs using vMotion.
- Ensure compatibility between the guest operating system and ESXi host driver across vSphere versions.
- Use vSphere DRS for the initial placement of VMs and load-balancing.

Testbed setup

The testbed had the following hardware and software set up and installed.

Table 1. Testbed configuration

Component	Configuration
Server	Dell PowerEdge R650 server with 2 Intel® Xeon® Gold 6330 Processors, 512GB RAM
GPU	2 Intel Flex Series 140 GPUs
vCPUS	4
Memory	8GB
Disk	256GB
OS	Windows 11 Enterprise
Applications installed	MS Office 365, Google Chrome browser, Adobe Acrobat Reader
VRAM	96MB
vSGA (3D Memory)	1GB
SR-IOV/Flex 140 (GPU profile)	1GB
Virtualization software	VMware vSphere 8.0 U3

Table 2. Application testing framework

Application	Operations/benchmarks tested	Automation mechanism	FPS, smoothness, and image-quality metrics referred to in graphs and charts
Microsoft Word	Open, close, page up, page down, edit, save	COM API from C#	Metrics are presented cumulatively and referred to as VDI because these applications are most often associated with users of VDI desktops.
Microsoft Excel	Open, close, page up, page down, edit, save, compute simple formulas on columns	COM API from C#	
Microsoft PowerPoint	Open, close, present, edit, save	COM API from C#	
Adobe Acrobat Reader	Open, close, browse, page up, page down	In-house AutoIT script	
Web video	Viewing VMware instructional videos on vmware.com	In-house AutoIT script	Web video
CAD viewers: eDrawings Viewer & FreeCAD	Open, zoom in, zoom out, rotate, pan left, pan right	In-house AutoIT script	eDrawings or FreeCAD
WebGL benchmarks	Unity, BMark, Aquarium	—	WebGL

We installed a Windows 11 Enterprise VM on a 2-socket Dell R650 server with 2 Intel Data Center GPU Flex 140 graphics cards. Using the application testing framework shown in table 2, we ran tests with MS Office, Adobe Acrobat, Web video playback, WebGL benchmarks, and CAD viewers.

We measured the user experience[1] while running these applications on three configurations:

- CPU-only 3D graphics
- 3D graphics support using vSGA on Flex 140
- 3D graphics support using SR-IOV for Flex 140

[1] The methodology for measuring user experience is described in Appendix A.

Our goal was to demonstrate that vSGA on Intel Data Center GPU Flex Series offers a user experience that is much better than a CPU-only graphics stack and comparable to that with SR-IOV.

Metrics

We characterized the visual user experience using three metrics:

- **Frames per second (FPS):** This is the number of *distinct* frames displayed per second on the client side.
- **Smoothness:** This shows how slowly the screen changes in response to user input. It is only possible to understand smoothness when comparing the value measured on a system under test (SUT) to the value measured on a reference system also running the same workload. Because some workloads can cause jerky changes in the screen content, a random measure of smoothness might incorrectly report that the SUT is not smooth when the workload is actually causing the jerky changes in the images. A score of 1.0 means the smoothness is perfect. Scores range from 0.0 to 1.0.
- **Image quality:** We define the quality of an image as the absence of blurring and blemishes in the form of salt and pepper noise, splotches, black artifacts, or pincushion distortions. We use a pair of CNN models to look for blurring or blemishes. Our image quality metric is a number between 0.0 and 1.0, with 1.0 being a perfect score.

We defined the user experience while executing a particular application (Office, Acrobat, Web video, CAD viewers, and WebGL) in a given configuration (CPU only, vSGA, or SR-IOV) as the **geometric mean** of the values of the three metrics: FPS, smoothness, and image quality. The overall user experience was between 0.0 and 1.0, with 1.0 being the highest score.

Note: The user experience is only a comparative score and not an absolute value.

We denoted the user experience of the SR-IOV stack for Flex 140 as the reference system. We compared the reference system's performance to vSGA on Flex 140 and the CPU-only configuration to the SR-IOV stack.

For more details, see Appendix A: Measurement details.

Results

Note: For the graphs in this section, **SRIOV-PT/Flex140** is the column label for the SR-IOV passthrough to Flex 140 configuration. **vSGA/Flex140** is the column label for the vSGA with Flex 140 configuration, and **CPU only** is the column label for the CPU-only configuration (no graphics adapter).

VDI vs Web video performance

Figures 2-5 show that the performance of the vSGA stack on Flex 140 is better than that of a CPU-only graphics stack and shows similar performance to that of the SR-IOV stack on Flex 140.

We tested VDI and web video performance using the techniques described in table 2. The applications included:

- **VDI:** Microsoft Office and Adobe Acrobat Reader
- **Web video:** Instructional videos from https://www.vmware.com

Figure 2. Normalized frames per second performance results for VDI vs Web video. SRIOV-PT/Flex140 has the same FPS rates for each, and the vSGA/Flex140 and CPU-only configurations have lower FPS rates for VDI. Web video FPS is similar across the board.

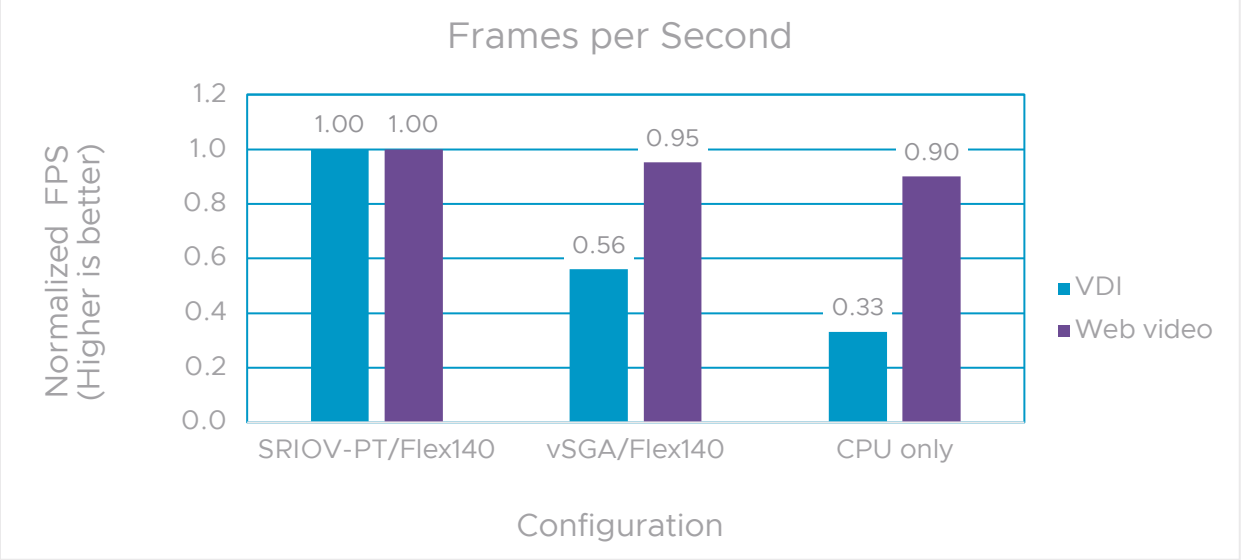


Figure 3. Normalized smoothness performance results for VDI vs Web video. SRIOV-PT/Flex140 has the same performance for each. vSGA shows better smoothness with VDI than Web video, but they are quite similar. The CPU-only configuration also shows VDI and Web video with similar smoothness, but Web video edges out VDI.

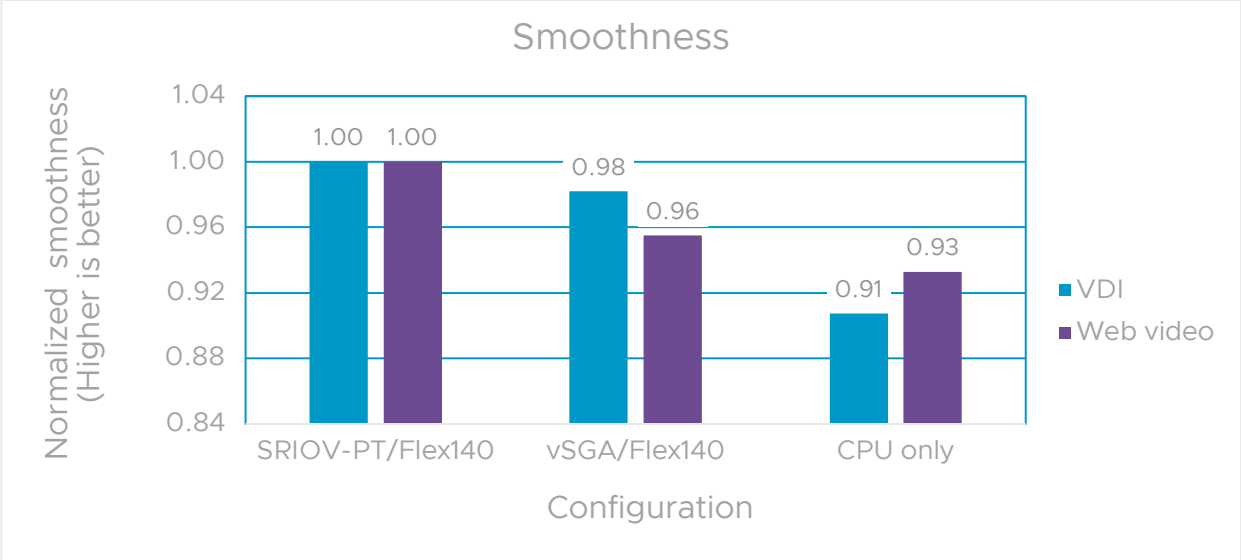


Figure 4. Normalized image quality performance results for VDI vs Web video. Web video performs well across the board with similar results to the reference stack (SRIOV-PT/Flex140). VDI shows almost the same image quality for SRIOV-PT/Flex140 and vSGA/Flex140, but it drops off about 25% with the CPU-only configuration.

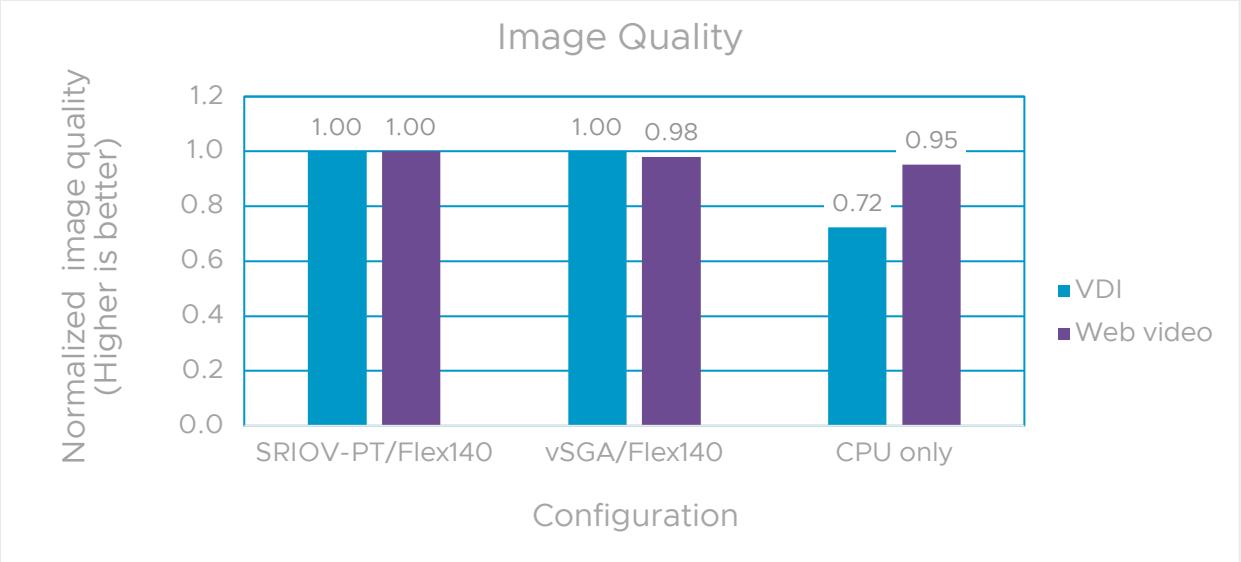
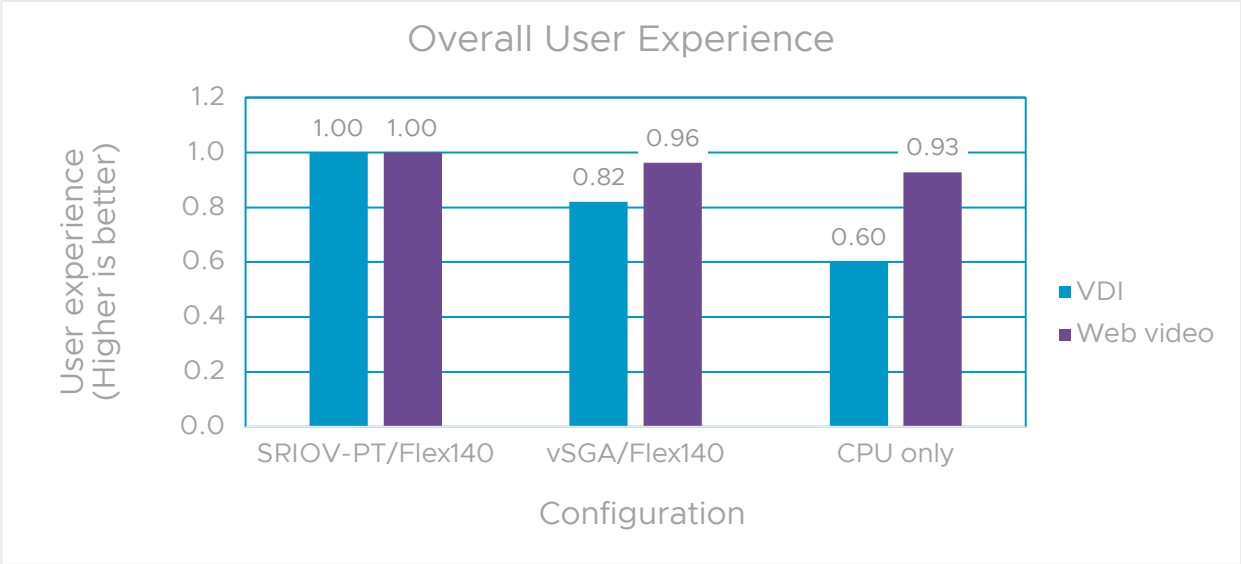


Figure 5. The overall user experience is similar across the board for Web video, with about 20% degradation for VDI with vSGA compared to SR-IOV/Flex 140. VDI performance drops about 40% for the CPU-only configuration.



WebGL performance benchmarks

We ran the WebGL benchmarks Unity, BMark, and Aquarium to compare the performance of the vSGA stack with CPU-only 3D graphics and the SR-IOV stack on Flex 140. Figures 6-7 show that the performance of the vSGA stack on Flex 140 is better than that of a CPU-only graphics stack and similar to that of the SR-IOV stack on Flex 140.

Figure 6. Normalized frames per second for WebGL Aquarium. Fish-100 and Fish-10000 render 100 and 10,000 fish, respectively. The reference configuration of SR-IOV on Flex140 shows the same performance as the vSGA configuration. Both outperform the CPU-only configuration.

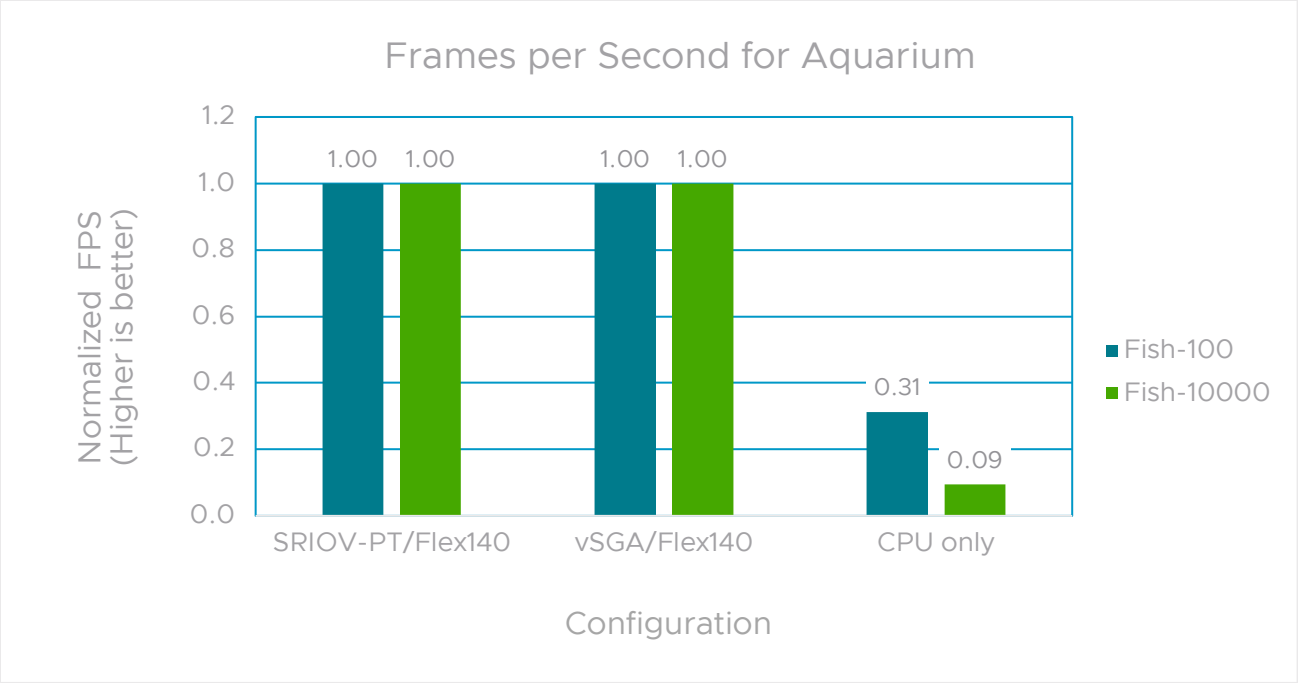
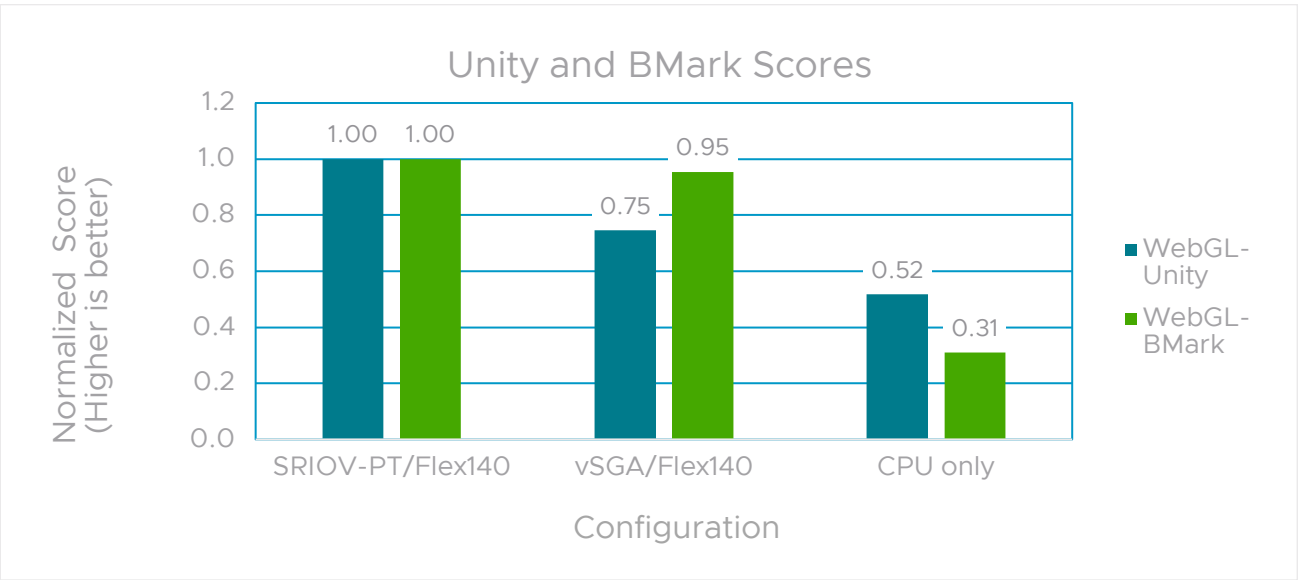


Figure 7. Normalized WebGL scores for Unity and BMark. The reference configuration of SR-IOV on Flex140 shows similar performance to the vSGA configuration, with BMark just 5% lower. Unity is 25% lower than SR-IOV-PT/Flex140. Both BMark and Unity on vSGA/Flex140 outperform the CPU-only configuration.



CAD viewer performance comparison

In a final set of experiments, we viewed the Bike and Turbofan models with eDrawings and the ADEX and clock models with FreeCAD. We ran these CAD viewers on all three configurations and measured the frames per second and image quality. Figures 8-9 show the results. The Bike and clock models are very complex and challenge even the SR-IOV/Flex 140 configuration. Turbofan and ADEX are simple models—all the configurations performed well with these.

Figure 8. Normalized frames per second for the eDrawings and FreeCAD viewers running the Bike, Turbofan, ADEX, and clock models.

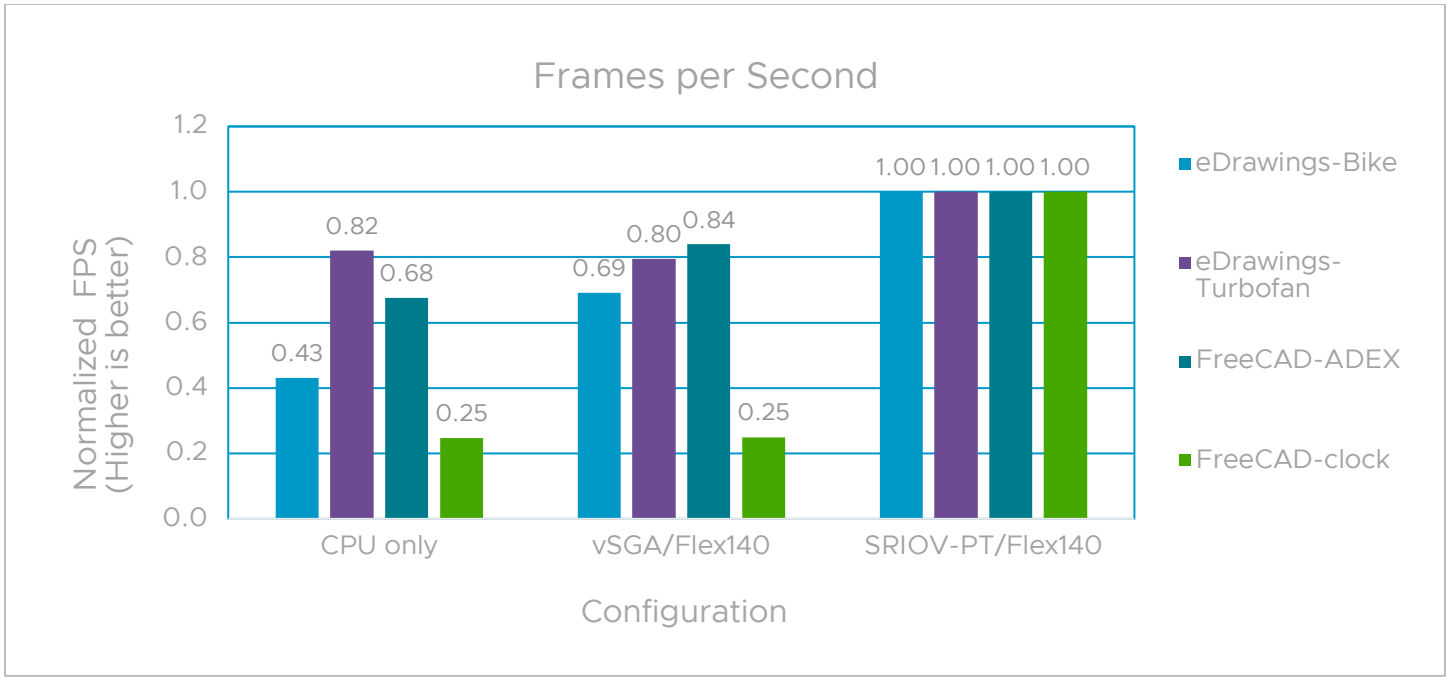


Figure 9: Normalized image quality for eDrawings and FreeCAD running the Bike, Turbofan, ADEX, and clock models.

Conclusion

vSGA's performance, measured as user experience, is better than CPU-only 3D graphics for VDI (that is, MS Office and Adobe Acrobat Reader), Web video, WebGL, and CAD viewer workloads. Performance is also comparable to that of an SR-IOV/passthrough to Flex 140 solution. Therefore, we recommend vSGA as the 3D graphics stack of choice based on its performance and the availability of vSphere virtualization features for these workloads.

Appendix A: Measurement details

The quality of the displayed images and responsiveness to user input primarily determines user experience in VDI applications. We characterized the visual user experience using the three metrics frames per second (FPS), smoothness, and image quality, as described in Metrics earlier in this document.

We measured FPS with a combination of a structural similarity metric (SSIM) and a CNN model to reduce frame overcounting caused by techniques like build-to-lossless or progressive sharpening, which are used by most remote display protocols. We measured smoothness using a time series analysis of the SSIM of successive frames displayed by the remote display protocol. We used a novel approach to measure image quality by counting the number of frames with no blurring or blemishes. We accomplished this using two CNN models, both of which had an F1 score > 0.98, which we built to identify blurred or blemished frames. *Our approach to measuring user experience is advantageous because it mainly relies on deep CNN models, which eliminates the need to make assumptions about the distribution of pixel intensities, the availability of reference images that correspond exactly to the display stacks under comparison, or prefabricated workloads.*

An ordered sequence of screenshots from the system under test served as the input for our measurement mechanism. We did not include a mechanism for taking screenshots in our tool because many other tools could accomplish this. To calculate the three metrics of interest—FPS, smoothness, and image quality—we usually examined the sequenced collection of screenshots, though we did not always save JPEG image files. It was crucial to examine the screenshot sequence in the order they were taken; otherwise, the results might have been spurious. The expected frame rate for the screenshots was approximately 45 frames per second, which was significantly higher than the frame rate used for VDI sessions. This guaranteed that no frames were missed in the collected sequence, but it also meant that many of the saved screenshots were duplicates because nothing had changed in between successive samplings.

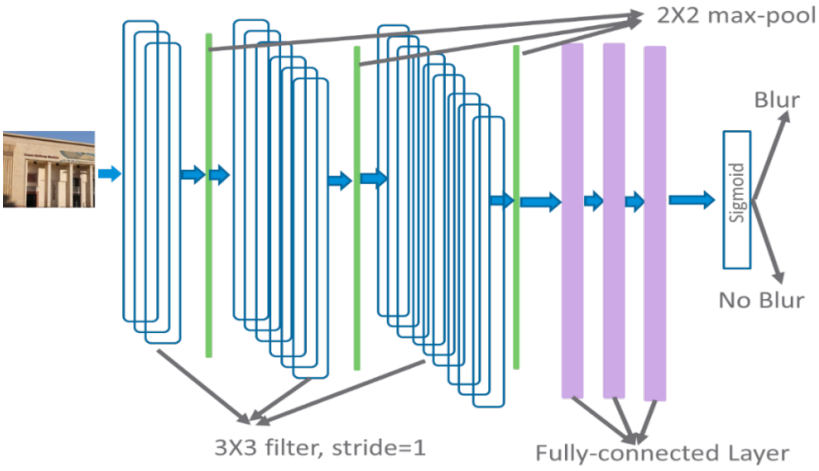
FPS

This is the number of **distinct** frames displayed on the client side. FPS, in our opinion, is a fundamental measure of user experience; low values of FPS might indicate poor user experience. Since we used a no-reference approach with no watermarks on the screen, we counted frames by identifying the number of screenshots with distinct content. To do this, we compared successive screenshots, two at a time; if any two were identical, the first screenshot was discarded as a duplicate. The traditional or popular approach to measure FPS is to compute the SSIM of every pair of screenshots in the recorded sequence; if the SSIM value turns out to be **less** than an arbitrary threshold, the first screenshot of the pair is counted as a unique frame; otherwise, it is discarded as a duplicate. The drawback to this approach stems from how VDI protocols and most modern lossy compression schemes function, which is by using build-to-lossless or sharpening mechanisms, whereby they first display a blurred image, which is then sharpened. This leads to overcounting because blurred frames subsequently sharpened are all counted as distinct frames.

Our approach began with SSIM to identify potential distinct frames. If two frames were candidates for being counted as distinct, we checked if the first frame was a blurred version of the second. If it was, we did not count the first frame, avoiding overcounting. The pseudo-code is shown below.

##The sequence of screenshots is a sequence of frames. We compute the SSIM of two successive frames, say I_n, and I_{n+1}
frameCount = 0
For each frame I_n in the sequence of screenshots
If SSIM (I_n, I_{n+1}) < 0.99 ## I_n and I_{n+1} are probably distinct frames, so we would count I_n as a frame. But, before
we do that we want to check if I_n is a blurred version of I_{n+1}
We feed the first image I_n into an ML model M_0 (Figure 11) to test if it is blurred.
If I_n is blurred ## Given I_n is blurred, it could be a blurred version of I_{n+1} and then it does **NOT** get counted.
We feed I_n and I_{n+1} into a second ML model, M_1 (Figure 12) to test if I_n is a blurred version of I_{n+1}
If I_n is **NOT** blurred version of I_{n+1}
frameCount ++
EndIf
Else ## I_n is NOT blurred so, it is a good, valid frame that we count.
frameCount++
EndIf
EndIf
EndFor
FPS is just frameCount/Time over which screenshots were recorded.

Figure 11. We used the M_0 model to determine whether an image was blurred. The model had 9 layers. The input images were 256X256 pixels.

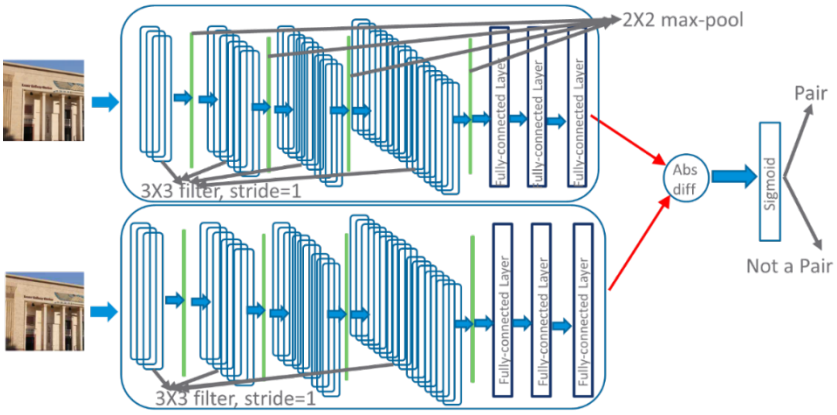


$$Smoothness \propto \frac{E_{0.7}}{E_{total}}$$

Smoothness

We used pairwise sequences of screenshots from the SUT to compute a time series: a sequence of SSIM values. Given this time series, we computed the Fourier transform and then calculated the energy in the lower 70% of the spectrum, $E_{0.7}$. We used the result to compute the total energy in the series, E_{total}. The smoothness measure was proportional to the ratio of $E_{0.7}$ and the total energy in the time series, E_{total}. This ratio was between 0.0 and 1.0, with a score of 1.0 being a perfect value. A low value for this ratio could be due to the nature of the workload, or due to a poor display stack. The only way to distinguish between these two scenarios was to compare the smoothness value from the SUT to that measured on a reference system with a top-of-the-line GPU-enabled stack running the same workload. The smoothness metric from the SUT was normalized using a reference value from the reference system running the same workload. This helped to eliminate the impact of the workload characteristics on the smoothness measure.

Figure 12: The M_1. model took a pair of images as input. If the first image was a blurred version of the second, it classified them as *a pair*. If not, it classified them as *not a pair*. The model had two identical halves, and each half had 11 layers plus a layer to take the absolute difference of the vectors from the last fully connected layer, which then fed into a sigmoid layer. The input images was 256X256 pixels.

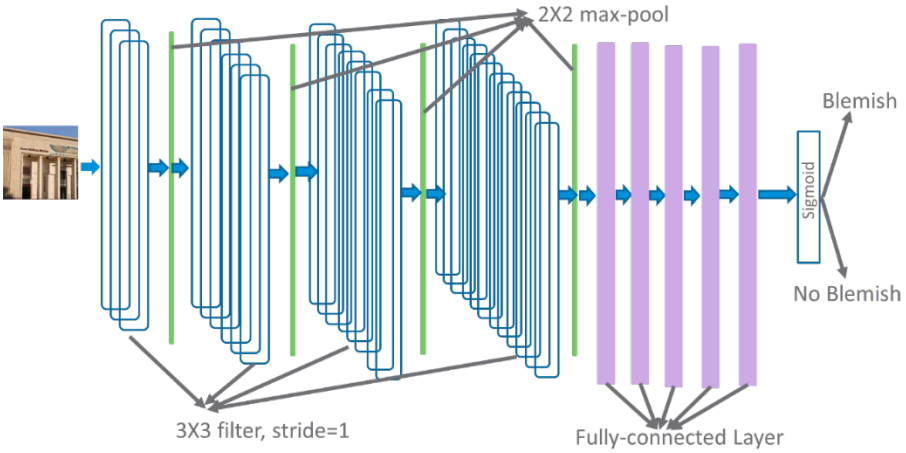


This approach works to measure smoothness because when a time series has many sharp jumps, it would have more high-frequency components and, hence, less energy in the lower 70% of the spectrum. Therefore, comparing the energy in the low-frequency region to the total energy in the time series gave us a measure of how smooth the time series was.

Image quality

We defined image quality as the absence of blurring and blemishes in the form of salt and pepper noise, splotches, black artifacts, or pincushion distortions. We used a pair of CNN models, M_0 and M_2, to look for blurring or blemishes.

Figure 13: We used the M_2 model to determine whether an image had a blemish: black artifacts, salt-and-pepper noise, pincushion distortion, or splotches. The model had 13 layers. The input images were 256X256 pixels.

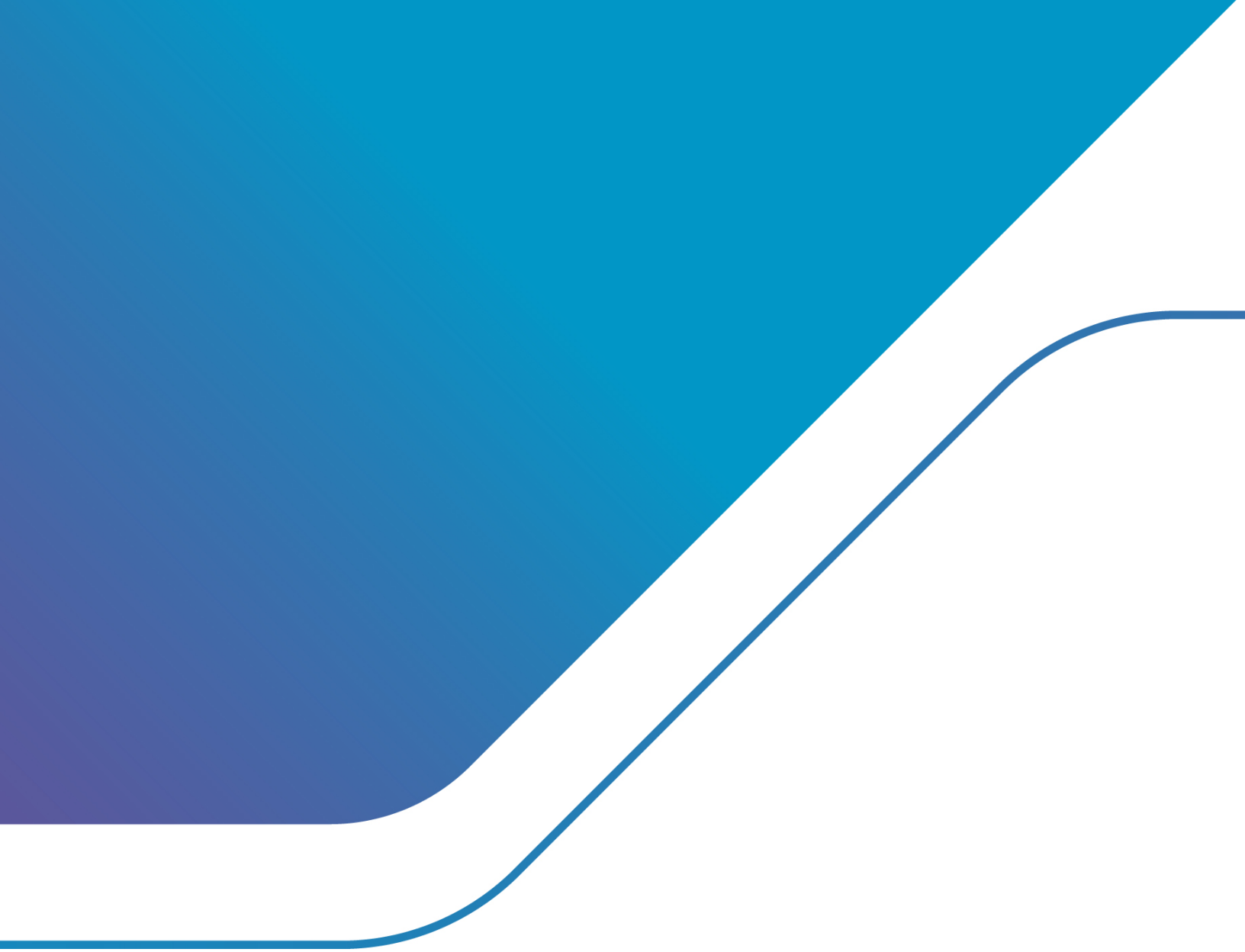


About the author

Hari Sivaraman is a software engineer in Broadcom's VMware Cloud Foundation (VCF) Performance team. Hari has extensive experience designing and running experiments to measure and predict the performance of computer systems, and he has built analytical and simulation models to estimate performance and answer "what-if" questions. He works on accelerated computing for graphics and ML/AI in cloud management systems. Hari has written and collaborated on several papers and blog articles about AI/ML and GPU acceleration topics regarding VMware vSphere performance. He has 25 patents and has been awarded VMware's Most Prolific Inventor award twice.

Acknowledgments

The author thanks the managers and colleagues who reviewed, analyzed, edited, and improved this technical white paper.



vmware® by Broadcom

Copyright © 2024 Broadcom. All rights reserved.
The term "Broadcom" refers to Broadcom Inc. and/or its subsidiaries. For more information, go to www.broadcom.com. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies. Broadcom reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design. Broadcom believes the Information herein is accurate and reliable. However, Broadcom does not assume any liability arising out of the application or use of this information, or the application or use of any product described herein. It does not convey any license under its patent rights or the rights of others.