TechTarget | Custom Media

WHITE PAPER

Consistency and Security: AMD's approach to GPU virtualization

By Gabe Knuth

GPU-accelerated virtual desktops are becoming more commonplace as the software we use becomes more demanding for graphics hardware and the ROI improves. Without proper selection and management of GPU resources for virtual desktop environments, the user experience when running these applications can be suboptimal, resulting in not only a bad experience but poor adoption or worse: a failed project.

Today, it's more important than ever to factor in GPU resources to your desktop virtualization projects, but if you're not GPU-savvy, you may need some help. In this white paper, we'll take a look at the approach AMD uses so you can better understand what's going on behind the scenes. The goal is to arm you with the information you need for success as you head into a GPU virtualization project.

One of the reasons AMD's virtual GPU solution is unique is because AMD uses something called SR-IOV to enable hardware-assisted GPU virtualization for scheduling, video decoding and security, leading to a highly predictable user experience.

Since the term SR-IOV might not be familiar to you, let's start there.

What is SR-IOV?

Virtualization started off as a purely software-based solution, and over time, more capabilities have been added to hardware. These changes eliminated the need to dedicate costly CPU and memory resources to the act of virtualization and freed up resources that allowed organizations to put high-intensity workloads in virtual environments. First, virtualization capabilities were added directly to the CPU, which worked well for simple workloads that only used CPU. When additional hardware

RADEON PRO AMD brianmadden

capabilities like networking or graphics entered the fray, though, those devices became shared resources. That's where SR-IOV comes in.

SR-IOV, which stands for single-root input/output virtualization, is a PCI-SIG interconnect standard for the hardware isolation of PCI Express (PCIe) resources. There are lot of complex diagrams showing how SR-IOV works, but essentially, it allows you to dedicate portions of PCI Express hardware to virtual machines (VMs) in a secure way that ensures consistent performance for each user.

SR-IOV works by breaking the hardware into two types of functions: Physical Functions and Virtual Functions. The *Physical Function*, or PF, of a device represents the primary function of the device, which in this case is a physical GPU. It has full configuration resources, which means that it's possible to configure and control the PCIe device via the PF. It contains the device's SR-IOV capabilities, manages how a device is used, and manages traffic between it and the Virtual Functions.

Virtual Functions, or VFs, represent predefined slices of physical resources. In the case of a GPU, this could mean frame buffer memory and GPU cores. There can be multiple VFs within a virtualized GPU, each one isolated from the other. Each VF has its own routing ID, or RID, that allows an I/O scheduler running on the GPU to keep track of which operations belong to it.

With this approach, data can be securely moved from the VM, through the VF, and to the GPU hardware (the PF) without affecting other VFs or VMs. VMs talk directly to the VF as if it is the actual hardware (because it is the actual hardware), an approach that removes the need to have software in between the hypervisor and the VM.

While the SR-IOV framework can be applied to many types of PCIe devices, we need to

focus on how it specifically works with graphics virtualization. Graphics presents an interesting challenge because of the large amounts of data and transfer speeds required to provide a good experience. To pull it off, you need to add features like memory separation, scheduling of operations, and additional processing power to handle not just graphics, but GPU compute and video encoding, too. AMD has built all of this into its MxGPU cards.

The AMD approach

By combining its own technology with SR-IOV, AMD has assembled a unique approach to graphics virtualization that offers distinct hardware advantages over software virtualization of GPUs, which may or may not be important to you. Let's dig in.

With SR-IOV, the physical separation is accomplished by allocating specific registers and memory for each VF. This allows the PCIe device to present multiple instances of itself to the hypervisor or specific VMs, with each instance tied back to a VF. The number of VFs that can be presented depends on the configuration of the GPU. AMD splits its up into a maximum of 16 VFs, so you can support as many as 16 VMs per GPU, or 32 VMs per card.

Upon initialization, the GPU is pre-partitioned into a number of VFs. When a new VM is being launched by the hypervisor, an available VF is designated to that VM. The VF will be assigned to the VM as its own GPU, upon which all the GPU functionality will be enabled within the VM. This process is repeated each time a new VM is launched and a VF is designated until all the VFs on the GPU have been assigned.

Since a virtualized GPU is shared across many VMs, the GPU hardware tracks the data for each VF and saves the context of each VF when it is time to switch. Switching is accomplished via a hardware-based scheduler that AMD built onto the card. In addition to doling out time on the GPU itself, the scheduler switches VFs between the video encoder and security hardware that AMD put on the card.

This differs from software-based approaches that share those resources amongst all the users on a host. Additionally, because each VM is tied to a VF, a VM or VF hang can be detected and resolved at the individual VF level instead of having to reset the entire card—meaning other virtual user sessions are not disrupted.

Let's take a closer look at scheduling.

With AMD's approach, hardware logic built into the GPU is responsible for scheduling operations between the PF, the VFs and the GPU engines. Because the scheduling is done in hardware, each VF is guaranteed to receive the same slice of time as the other VFs to perform an operation.

Since each VF has its own memory and data space, the scheduler attaches those spaces to the GPU for a given time slice, runs for a certain amount of time, and then swaps out one VF's memory and data for another's. Once the scheduling parameters are set, they are immutable, which means you have the same slice of time regardless of the task each VF is doing.

Scheduling in this manner produces a few benefits. First, it means that a single user can't run away with the GPU, which would degrade performance in other VMs on the same host. It also means that even the first user to connect sees consistent, predictable performance throughout their session, regardless of how many users are also using the GPU. Both of these are increasingly important as enterprises start to deliver GPU capabilities to more users.

Other platforms give the first user 100% of the resources on the card, which would then be reduced to 50% as the next user logs in, 33% when the third user logs in, 25% for the fourth user, and so on. That means the user's performance continually degrades until the card reaches its maximum number of users. With AMD's approach, it is possible to dedicate all the resources of the card to the first user, depending on your configuration profile.

Once the data, routing and scheduling has been accomplished for each VF, then the GPU's main processing engines kick in to service the requests of the resource-hungry applications in each VM.

MxGPU security

We've talked a lot about the isolation between VFs (and therefore VMs) as it relates to AMD's GPUs, but this goes beyond organization and scheduling. Security is also a key benefit of this approach, and as the bad guys look for ways into your organization, AMD's approach to graphics virtualization can help provide an extra layer of insurance.

Since the GPU hardware keeps track of which VMs are talking to which VFs, using configuration registers that only the PF has access to, it knows when a new VM is connected and clears the frame buffer memory and the VF-specific GPU registers before connecting a new VM. This prevents unwanted, leftover data from being passed into the new VM.

Additionally, each VF is mapped to dedicated physical system memory, where it's protected by IOMMU (I/O memory management unit). IOMMU, the generic name for Intel's VT-d and AMD's IOV technologies, assigns and maintains memory mapping between the physical hardware and the hypervisor.

Even the AMD GPU's firmware and microcode are protected to prevent intrusion. Any software, including Windows or hypervisor drivers, attempting to load its own firmware code onto the GPU itself must be properly signed and authenticated by the onboard security processor. Since the authentication uses hardware root of trust, there is no way to side-load software/firmware.

Conclusion

It's important to understand what's going on behind the scenes, and hopefully this white paper has given you additional background into the inner workings of SR-IOV and the technology AMD has used to build its MxGPU platform. With MxGPU, AMD delivers a consistent, predictable user experience with a platform that is also extraordinarily difficult to exploit. Though this kind of security might not be at the top of your list of priorities right now, it should be factored into your decision-making process as you assemble a GPU-enabled VDI platform.