MxGPU with AMD S7150 under Proxmox VE 5.x

Contents

Introduction

This is a Testing Report and How-To for using the MxGPU feature of an AMD S7150 Graphics card under PVE 5.x. These cards can provide hardware-accelerated 3D graphics to multiple VMs with a single card instead of using one card per VM (normal PCI passthrough) or using a software 3D graphics card (QXL/Spice).

AMDs open source GIM driver[1] is needed on the host.

WARNING: Our tests showed that this may be unstable and experimental, please see the 'Notes' section below for more details.

Hardware Notes

We tested the card in the following configurations:

Cookies help us deliver our services. By using our services, you agree to our use of cookies. More information OK

Works	Hardware Type	Mainboard	CPU	Memory	Errors	Notes
No	Consumer	ASUS Z170-A	Intel 6700k	32GB DDR4 Memory	Loading GIM failed with a PCI Bus Error that it did not have sufficient resources.	The firmware of the Mainboard is not suited for this use.
No	Low-end Server	Supermicro X10SDV-6C-TLN4F	Intel Xeon D-1528	32GB DDR4 Memory	PCI Bus Errors during use resulting in guest and host crashes.	The Platform is not suited for this use.
Yes	High-end Server	Supermicro H11SSL-i	AMD Epyc 7351P 16-Core Processor	64 GB DDR4 Memory	Linux guest instability	OPROM for this card has to be set to Legacy.

Host Configuration

Make sure that the 'amdgpu' module is blacklisted before installing the card. This can be done via a file in /etc/modprobe.d/. For example put

```
blacklist amdgpu
```

into /etc/modprobe.d/blacklist-amdgpu.conf

Do not forget to update the initramfs[2] afterwards.

After that, you have to compile and install GIM[1]. For this, you need at least the packages 'git', 'pve-headers', 'gcc' and 'make'.

See their documentation about how you can compile and configure the module.

You can install the module via DKMS[3] (dynamic kernel module support), then the module gets automatically recompiled on every kernel upgrade.

After installing the module, you can do

```
modprobe gim
```

and now you should see the virtual functions via 'lspci'. Example output:

```
...
41:00.0 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] TongaXT GL [FirePro S7150]
41:02.0 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] TongaXTV GL [FirePro S7150V]
41:02.1 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] TongaXTV GL [FirePro S7150V]
...
```

Cookies help us deliver our services. By using our services, you agree to our use of cookies. More information

Those Devices (FirePro S7150V) can now be passed through via the standard PCI passthrough mechanism[4].

Client Configuration

Windows 10

Create a new VM, pass through a Virtual GPU, and install Windows 10.

After that, enable Remote Desktop and install the Radeon Pro Drivers[5].

After a reboot of the VM, you can now connect via Remote Desktop to the VM and use the graphics card.

Rendering with OpenCL in Blender in Windows 10

Running Unigine Valley Benchmark in Windows 10

Ubuntu 18.04

Create a new VM without passing through the Virtual GPU yet, and install Ubuntu 18.04.

After that, install the amdgpu-pro driver from AMDs homepage[5].

While testing, we found that the AMDGPU Pro driver Version 18.40[6] works most of the time. (18.30 did load but produced many guest kernel errors and prevented the use of it; 18.50 resulted in guest kernel oopses).

Install a desktop environment (for example XFCE with the meta-package xubuntu-desktop) and a display manager (for instance lightdm).

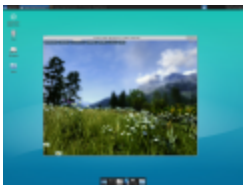
Install a VNC Server[7] (or similar) to be able to access a local X server and configure it to start automatically.

Now power off the VM, add the virtual function and start the VM again.

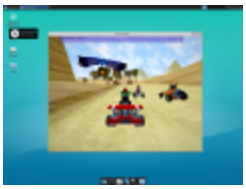
Note: Depending on the exact guest kernel and driver version, there may be some kernel errors and warning even if it is working.

At this point, you should be able to connect via VNC (or another protocol) and use the virtual GPU.

Cookies help us deliver our services. By using our services, you agree to our use of cookies. **More information**



Unigine Valley on Ubuntu 18.04



Super Tux Kart on Ubuntu 18.04

Notes

Stability

In our tests the Linux guest drivers were very unstable. It worked with a single Ubuntu guest, but led to chrashes/hangs (of the guests and the host) after another guest was started, regardless of the client OS of the other guests.

The Windows guest drivers worked more stable, but there were occasional resets/crashes and sometimes blue screens in the guest after starting multiple Windows guests, this only occurred when at least one Linux guest was started since boot, so only starting and using Windows guests should work. (The relevant bug report is here: https://github.com/GPUOpen-LibrariesAndSDKs/MxGPU-Virtualization/issues/16)

Debugging

For debugging purposes, AMD includes the useful tool 'GRU' with the sources of GIM[1]. This is found in the 'utils/gru' folder and can simply be built with 'make'.

You can use this Tool to see which functions are in use and how much resources are used. Also, it provides a mechanism to reset the card and its functions. Here is an example output:

```
GRU
Copyright (C) 2017~2018  Advanced Micro Devices, Inc.

Type 'help' for help. Optional launch parameter is index of card to use.
GRU> status

+-----+--------------+----------+------------+-----------+--------------------+
| GPU | Name         | Cur Volt | GFX EngClk | Mem Usage | Current DPM Level  |
|     | BusId        | Temp     | Avail VF   | GFX Usage | Power Usage        |
+=====+==============+==========+============+===========+====================+
|   0 | S7150        | 0.5750 V | 313.10 MHz | 49.79 %   | 1                  |
|     | 0000:41:00.0 | 57.00 C  | 2          | 45.17 %   | 33.37 W            |
+-----+--------------+----------+------------+-----------+--------------------+
GRU> list

+-----+--------------+---------+------------+--------+------------+-----------+
| GPU | Name         | DPM Cap | FB Size    | Max VF | GFX Engine | PL Speed  |
|     | BusId        | PWR Cap | Encoder    | ECC    | MAX Clock  | PL Width  |
|   0 | S7150        | 8       | 8190 M     | 4      | GFX8       | 8 GT/s    |
```

Cookies help us deliver our services. By using our services, you agree to our use of cookies. More information

```
|     | 0000:41:00.0 | 109 W   | None       | No     | 1000 MHz   | x16       |
+-----+--------------+---------+------------+--------+------------+-----------+
GRU> open 0000:41:00.0
GRU>GPU:41:00.0> list

+----+--------+--------------+------------+-----------+---------+-------------+
| VF | Type   | BusId        | Name       | VF State  | VF Size | GFX EngPart |
+====+========+==============+============+===========+=========+=============+
|  0 | S7150  | 0000:41:02.0 | MxGPU_V1_4 | Active    | 1968 M  | 25%         |
+----+--------+--------------+------------+-----------+---------+-------------+
|  1 | S7150  | 0000:41:02.1 | MxGPU_V1_4 | Active    | 1968 M  | 25%         |
+----+--------+--------------+------------+-----------+---------+-------------+
|  2 | S7150  | 0000:41:02.2 | MxGPU_V1_4 | Available | 1968 M  | 25%         |
+----+--------+--------------+------------+-----------+---------+-------------+
|  3 | S7150  | 0000:41:02.3 | MxGPU_V1_4 | Available | 1968 M  | 25%         |
+----+--------+--------------+------------+-----------+---------+-------------+
GRU>GPU:41:00.0> status

+----+-------------+--------------+--------------+--------------+-------------+
| VF | Type        | BusId        | Active Time  | Running Time | Reset Times |
+====+=============+==============+==============+==============+=============+
|  0 | S7150       | 0000:41:02.0 | 0:41:48      | 0:59:29      | 0           |
+----+-------------+--------------+--------------+--------------+-------------+
|  1 | S7150       | 0000:41:02.1 | 0:14:13      | 0:31:50      | 0           |
+----+-------------+--------------+--------------+--------------+-------------+
|  2 | S7150       | 0000:41:02.2 | 0:0:0        | 0:0:0        | 0           |
+----+-------------+--------------+--------------+--------------+-------------+
|  3 | S7150       | 0000:41:02.3 | 0:0:0        | 0:0:0        | 0           |
+----+-------------+--------------+--------------+--------------+-------------+
GRU>GPU:41:00.0>
```

References

1. GIM Open Source Driver. https://github.com/GPUOpen-LibrariesAndSDKs/MxGPU-Virtualization
2. https://pve.proxmox.com/pve-docs/chapter-qm.html#qm_pci_passthrough_update_initramfs
3. Debian DKMS documentation. https://wiki.debian.org/KernelDKMS
4. PCI Passthrough Documentation https://pve.proxmox.com/pve-docs/chapter-qm.html#qm_pci_passthrough
5. AMD Radeon Pro Drivers for S7150 https://www.amd.com/en/support/professional-graphics/firepro/firepro-s-series/firepro-s7150-active-cooling
6. AMDGPU Pro Driver 18.40. https://www.amd.com/en/support/kb/release-notes/rn-prorad-lin-18-40
7. https://help.ubuntu.com/community/VNC/Servers

Retrieved from "https://pve.proxmox.com/mediawiki/index.php?title=MxGPU_with_AMD_S7150_under_Proxmox_VE_5.x&oldid=10244"

This page was last edited on 8 January 2019, at 12:45.

Cookies help us deliver our services. By using our services, you agree to our use of cookies. **More information**