# AMDGPU虚拟内存管理

## 概述

GPUVM 是 GPU 上提供的 MMU 功能。GPUVM 类似于老的ASIC上的GART，然而，它不是为整个 GPU 提供单个全局 GART 表，而是可以在任意时间激活多个 GPUVM 页表。GPUVM 页表可以包含混合的 VRAM 页和系统页（包括内存和 MMIO），系统页可以映射为 snooped（缓存的系统页）或 unsnooped（未缓存的系统页）。

每个活跃的 GPUVM 都有一个与之关联的 ID，并且每个 VMID 都有一个页表链接。当执行命令缓冲区时，内核会告诉引擎使用哪个 VMID 来处理该命令缓冲区。VMID 是在提交命令时动态分配的。用户空间驱动程序维护自己的地址空间，内核在它们提交命令缓冲区时会相应地设置它们的页表并分配 VMID。硬件最多支持同时激活 16 个 GPUVM。

每个 GPUVM 由 1-2 级或 1-5 级页表表示，具体取决于 ASIC 系列。GPUVM 支持每页的 RWX 属性，以及其他功能如加密和缓存属性。

VMID 0 是特殊的。它是用于内核驱动程序的 GPUVM。除了由页表管理的一个孔径外，VMID 0 还具有其他几个孔径。有一个直接访问 VRAM 的孔径，还有一个传统的 AGP 孔径，它只是将访问直接转发到匹配的系统物理地址（或在存在 IOMMU 时的 IOVA）。这些孔径提供了对这些内存的直接访问，而无需页表的开销。VMID 0 由内核驱动程序用于内存管理等任务。

GPU 客户端（即 GPU 上的引擎）使用 GPUVM VMID 访问内存。对于用户应用程序，每个应用程序可以有自己独特的 GPUVM 地址空间。应用程序管理地址空间，内核驱动程序管理每个进程的 GPUVM 页表。如果 GPU 客户端访问无效页，将会生成 GPU 页故障，类似于在 CPU 上访问无效页。

## VMID

VMID也就是页表寄存器的ID，上文提到硬件支持最大16个GPUVM，也就是硬件最多支持16组页表的配置。页表的配置和MMU的配置是分开的，MMU的数量和页表的数量不一样。

VMDID0用于 驱动。

VMID1-7用于 graphics/compute。

VIMID8-15用于 amdkfd。

对于纯计算卡，2个VMID用于编解码，其他都留给amdkfd。 在MI300X相关的驱动代码中，用了4-15个VMID用于amdkfd。XCP会按照奇偶均匀的分配VMID。

/\* For GFX9.4.3 and CPX mode, first XCD gets VMID range

\* 4-9 and second XCD gets VMID range 10-15.

\*/

## MMU

在AMDGPU中MMU或者叫MMU工作组（看不到硬件实际上的MMU数量）被称为各种HUB， MMHUB0 GFXHUB等。在各款芯片中，最多有8个GFXHUB + 4个MMHUB0 + 1个MMHUB1。其中GFXHUB是用于图形和计算的， 实际上只有MI300X有8个GFXHUB，也就是芯片有几个XCD就有几个GFXHUB，估计以后会更多。MMHUB0主要用于视频编解码和SDMA场景，MMHUB1作用类似MMHUB0， 似乎只有一款芯片有，可以忽略。

### VIMID和VM的关系

每个进程都有一个VM，也就是页表。 每个VM关联一个PASID。但是在硬件上只有将PASID和VMID的映射关系设置到寄存器中后，硬件才能使用这个VM。也就是对于一个VMID，在一个时刻只能为某一个VM服务。

VMID的使用

在nocpsched模式用户在创建queue时，驱动分配vmid。在cpsched模式，vmid似乎是mes来设置的（不明）。

# Intel gma500 gart驱动实现

[显卡内存管理机制及驱动实现(Intel gma500为例)](https://happyseeker.github.io/kernel/2016/07/22/memory-management-from-graphic-card-view.html)

# linux 内核 TTM和GEM

<https://www.cnblogs.com/hellobb/p/11023873.html>

<https://www.cnblogs.com/shoemaker/p/linux_graphics04.html>

[Linux drm内存管理(一) 浅谈TTM与GEM，为什么我们需要TTM和GEM？](https://blog.csdn.net/qq_40937426/article/details/127857533)