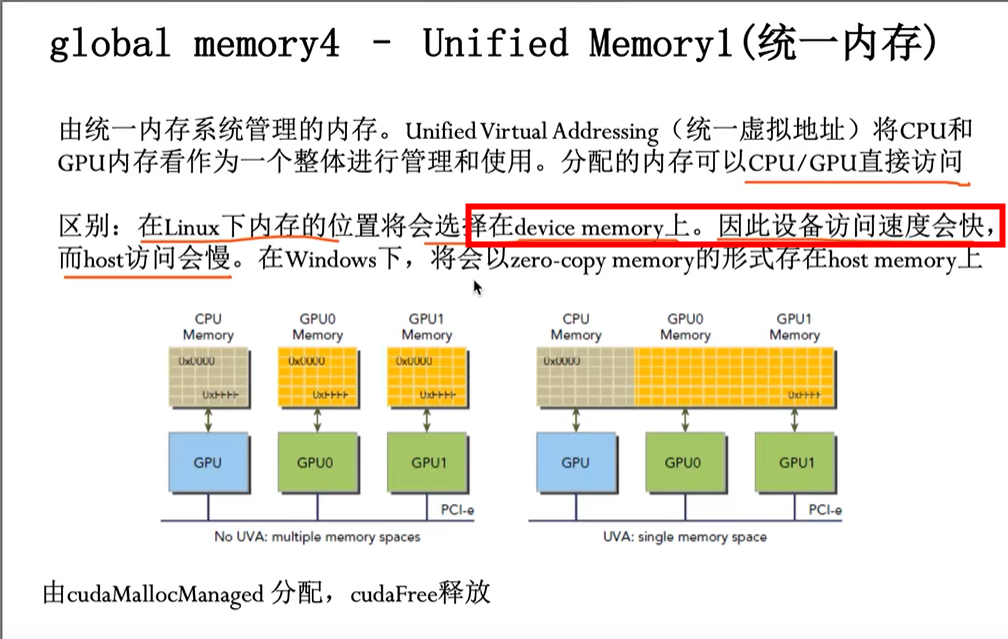
# 统一内存



# 利用异构内存管理简化 GPU 应用程序开发

异构内存管理（HMM）是一种 CUDA 内存管理功能，它扩展了 [CUDA 统一内存](https://developer.nvidia.cn/zh-cn/blog/unified-memory-in-cuda-6/" \t "https://developer.nvidia.com/zh-cn/blog/simplifying-gpu-application-development-with-heterogeneous-memory-management/_self) 的编程模型，包括系统分配内存在具有 PCIe 连接的 NVIDIA GPU 的系统上。系统分配内存是指最终由操作系统分配的内存；例如，通过 malloc，mmap，C++ 新操作员（当然使用前面的机制），或为应用程序设置 CPU 可访问内存的相关系统例程。

以前，在基于 PCIe 的机器上， GPU 无法直接访问系统分配的内存。 GPU 只能访问来自特殊分配器的内存，例如库cudaMallocManaged。

启用 HMM 后，所有应用程序线程（ GPU 或 CPU ）都可以直接访问应用程序系统分配的所有内存。与统一内存（可以被认为是 HMM 的子集或前身）一样，不需要在处理器之间手动复制系统分配的内存。这是因为它会根据处理器的使用情况自动放置在 CPU 或 GPU 上。

在 CUDA 驱动程序堆栈中， CPU 和 GPU 页错误通常用于发现内存应该放在哪里。同样，这种自动放置已经在统一内存中发生了——HMM 只是将行为扩展到覆盖系统分配的内存以及cudaMallocManaged记忆力

这种直接读取或写入整个应用程序内存地址空间的新能力将显著提高基于 CUDA 之上构建的所有编程模型的程序员生产力： CUDA C++、Fortran、 Python 中的标准并行性、ISO C++、ISO Fortran、OpenACC、OpenMP 和许多其他模型。

事实上，正如即将到来的示例所示，HMM 将 GPU 编程简化到 GPU programming 几乎与 CPU 编程一样可访问的程度。一些亮点：

编写 GPU 程序时，功能不需要显式内存管理；因此，一个初始的“初稿”程序可以是小而简单的。显式内存管理（用于性能调优）可以推迟到稍后的开发阶段。

对于不区分 CPU 和 GPU 存储器的编程语言， GPU programming 现在是实用的。

大型应用程序可以被 GPU 加速，而不需要大型内存管理重构或更改第三方库（源代码并不总是可用的）。

顺便说一句，[NVIDIA Grace Hopper](https://www.nvidia.cn/data-center/grace-hopper-superchip/" \t "https://developer.nvidia.com/zh-cn/blog/simplifying-gpu-application-development-with-heterogeneous-memory-management/_self) 通过硬件实现了所有 CPU 和 GPU 之间的内存一致性，从而本地支持统一内存编程模型。对于这样的系统，不需要 HMM，事实上，HMM 在这种情况下会被自动禁用。可以将 HMM 视为一种基于软件的方式，它提供了与 [NVIDIA Grace Hopper Superchip](https://developer.nvidia.cn/zh-cn/blog/nvidia-grace-hopper-superchip-architecture-in-depth/" \t "https://developer.nvidia.com/zh-cn/blog/simplifying-gpu-application-development-with-heterogeneous-memory-management/_self) 类似的功能。

## HMM 之前的统一内存



## HMM 之后的统一内存

<https://developer.nvidia.com/zh-cn/blog/simplifying-gpu-application-development-with-heterogeneous-memory-management/>



# HMM and CDM

2017 LSF/MM的第一个内存管理的topic是关于device memory不可寻址的问题，这里并非指CPU不可访问，而是指system memory与device memory缺乏cache-coherent语义，在使用的时候存在诸多不同以及由此引发了很多问题（比如DMA前需要pin住system memory）。目前有两个尝试的解决方案：异构内存管理(HMM) 与 cache-coherent设备（CDM）。

首先简单介绍一下HMM的功能，主要包括两类：

(a). address space mirroring, 即保证CPU page table（即CPU侧mmu）与 device page table（即device侧mmu，逻辑上的概念，比如set\_pte()与clear\_pte()两个操作可以解释为dma\_map\_page()与dma\_unmap\_page()）同步。底层同步的机制则依赖于kernel已有的mmu notifier特性，带来的好处之一就是DMA 与 page migration可以同时存在（此前需要pin住内存，然后DMA）。目前HMM实现了一套框架，update device page table具体实现则需要相应的device driver负责填充ops（参考例子：hmm\_dmirror字符设备）。(b). migration to and from device memory，即在system memory与device memory间无缝地进行页迁移，流程与普通的swap类似：CPU访问内存时触发page fault，进而将数据从device memory DMA至system memory。

本次会议讨论的几个点：(1). HMM对硬件feature的依赖，目前明确的是硬件需要能设置页表的访问权限，比如允许CPU或GPU（但不能同时）拥有excute权限；(2). HMM与IOMMU的异同：IOMMU对I/O设备进行了安全隔离，保护系统，HMM则需要捕获write fault，CPU与device触发时执行的处理逻辑也不一样；(3). 基于KSM机制，正在开发的一个write protection特性允许同一个系统上的多个GPU设备访问相同的数据区域。这里引入了一个问题：传统的写异常将触发COW，而HMM则是保证写者拥有唯一的写权限。fork()系统调用基于COW来复制地址空间，在HMM里则是先触发page fault的进程（child或parent）将取得page的所有权，这将依赖于触发的时间点，引入不确定性。为了避免该问题，Gorman建议强制进程通过madavise()系统调用设置MAD\_DONTFORK来避免该问题，这样所有的内存都将属于父进程。(4).Dan Williams提到HMM将GPU内存放置到ZONE\_DEVICE区域，目前persistent memory也使用了该区域，二者共用逻辑的话容易导致一些bug，目前只能通过仔细review来避免错误。(5). 目前还没有具体的driver使用了HMM，NVIDIA GPU的Nouveau-based驱动有希望在4.12的合并窗口中提交，HMM的相关patch仍然暂时保留在-mm分支。

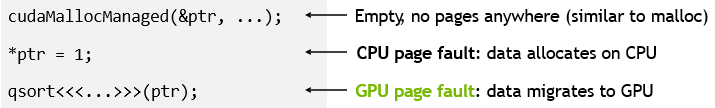
## CDM

CDM是IBM提出的另一个思路，主要通过硬件来解决目前所遇到的问题。在一些嵌入式系统中，device memory可以看成是一个没有CPU的NUMA节点，且提供cache-coherent语义。目前仍在开发中：<https://lwn.net/Articles/713035/> 。本次会议讨论的几个点：(1). 如何在Linux中使用这些设备？预期是提供给用户可选择的内存分配机制：用户决定申请CDM内存或者普通内存。NUMA balance需要关闭，因为直接在CDM内存与普通内存间进行页迁移存在问题。如果一定要页迁移，可通过DMA来加速。(2). 已有patch通过cpuset机制来隔离CDM；整个系统由于缺乏对完整的内存信息，无法实现内存均衡。zone链表将CDM隔离开来。THP页迁移的功能还有待进一步讨论。

# Pascal/Valta 架构上的 Unified Memory

硬件上引入PAGE MIGRATION ENGINE，支持Virtual Memory Demand Paging，其实就是gpu硬件层面支持页面错误处理机制【handle thousands of simultaneous page faults】。因为没有该特性的话，应用要提前将数据都加载到GPU显存中，这样势必会带来很大的开销。

**再以一个例子来说明：**



第一句代码：申请GPU显存，此时在GPU上不会创建页面。

第二句代码：CPU访问内存，出现也错误，数据将会在CPU中分配。

第三句代码：GPU内核访问，出现页错误，数据会migrate到GPU中。

**特点：**

* 如果系统不支持UVA，则系统会向CPU发送一个中断。
* Unified Memory driver会决定会映射还是迁移数据。

# Refercens

|  |  |
| --- | --- |
|  |  |
|  | Heterogeneous Memory Management (HMM)[¶](https://docs.kernel.org/5.19/vm/hmm.html" \l "heterogeneous-memory-management-hmm" \o "Permalink to this heading)  <https://docs.kernel.org/5.19/vm/hmm.html> |
|  |  |
|  |  |