### GPU程序效率瓶颈的主要矛盾研究优化

——访存效率与GPU数据传输速率

第7小组 杨滕超 储岱泽 莫益萌 刘冲 陈华机

**摘要：**随着图形处理器（GPU）在计算机系统中的广泛应用，GPU程序效率成为了一个重要的研究领域。本文关注了GPU程序效率瓶颈中的两个主要矛盾：访存效率和GPU数据传输速率，通过分析相关文献和实际案例，从降低内存访问延迟以及提高内存访问带宽两个角度论述了优化GPU访存效率的方案，并介绍了PCIe总线以及CPU与GPU的异构多核架构方案对于GPU数据传输速率的优化。

1. **引 言**

随着科学计算和数据处理需求的不断增长，图形处理器（GPU）作为一种强大的并行计算设备已经广泛应用于各个领域。相对于传统的中央处理器（CPU），GPU具有大规模并行处理能力，能够高效地执行大规模数据计算和图形渲染任务。GPU的出色性能和高效能力使其成为许多领域的首选计算平台，例如计算机视觉、深度学习、科学模拟、密码学等。

在利用GPU进行计算时，优化GPU程序的效率是至关重要的。然而，GPU程序的效率瓶颈存在着一系列复杂的问题和挑战，其中最主要的矛盾涉及到访存效率和GPU数据传输速率。GPU的并行架构特性使其在处理大规模数据时具有优势，但访问全局内存的延迟和带宽限制成为了GPU程序效率的主要限制因素。

研究GPU程序效率瓶颈的主要矛盾具有重要的意义。首先，深入理解GPU程序的效率瓶颈和解决方案可以帮助开发者充分利用GPU的计算潜力，提高计算性能和加速应用程序的执行速度。其次，针对GPU程序的效率瓶颈进行优化可以降低资源消耗，并在大规模数据处理和科学计算任务中节省宝贵的时间和资源。此外，对GPU程序效率瓶颈的研究还可以为GPU架构和系统设计提供指导，推动GPU技术的发展和创新。

本论文分析了GPU的重要性以及其体系结构，并深入探讨了GPU程序效率瓶颈的主要矛盾，重点关注访存效率和GPU数据传输速率。通过分析和研究现有的优化技术和方法，我们将提出一些解决GPU程序效率瓶颈的策略。

1. **关于GPU的介绍以及体系结构的分析**

**2.1 GPU的重要性**

GPU是英文单词“Graphics Processing Unit”的缩写，中文翻译为“图形处理器”，是一种在个人电脑、工作站、游戏机和一些移动设备上做图像和图形相关运算工作的微处理器，是显示卡的“大脑”。随着当今科技的发展，人们对VR、区块链、3D建模等技术的进步有了更多的需求，而这些领域的进步都依赖于图形算法处理速度的进步，而凡是跟图形计算相关的过程都需要用到GPU。

然而，GPU除了具有图形处理的功能之外，还是公认最优秀的AI加速器。由于其优秀的并行计算以及浮点运算能力，其在深度学习的训练和推理领域也有很广泛的运用。可以说不论是画质的要求还是AI处理的需求，GPU都是必不可少的，要想发展更多的科技企业就必须得不断优化GPU程序处理的效率，突破瓶颈。

**2.2 GPU体系结构分析**

接下来我们通过GPU与CPU的对比来认识GPU为什么会有如此强大的并行计算以及浮点计算的能力。

CPU和GPU是两种不同的计算机体系结构，它们的区别主要在于处理器的设计和用途。CPU以追求单线程性能为目标，处理器核心较少，每个核心具有较强的通用计算能力和缓存系统；而GPU则主要用于并行计算，处理器核心较多，每个核心的计算能力相对较弱，但通过并行化技术可以达到高效的计算速度。正如下图所示，其中绿色的是计算单元，橙红色的是存储单元，橙黄色的是控制单元，我们可以明显的对比出CPU不仅被Cache占据了大量的空间，而且还有着复杂的控制逻辑，相比之下计算能力只是CPU很小的一部分；而GPU则采用了众多的计算单元和超长的流水线，并且省去了Cache。

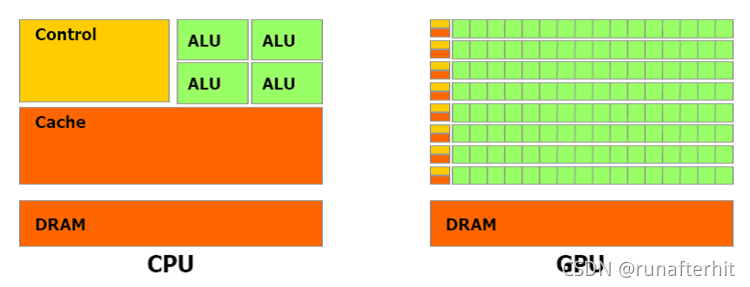


图1 CPU与GPU体系结构对比

从体系结构角度来看，CPU和GPU的区别也很明显。CPU的体系结构通常采用冯·诺依曼结构，即一个通用的、顺序执行的中央处理器和一组存储器件组成。而GPU的体系结构则通常采用SIMD（单指令多数据）结构，多个核心同时执行同一个指令，但是每个核心所处理的数据可以不同。因此，GPU的核心数量通常比CPU多得多。

此外，GPU还具有比CPU更强大的内存和存储带宽，这也是其能够实现高效并行计算的重要基础。GPU通常具有独立的显存，内存传输速度较快，可以更快地读取和写入数据。

综上所述，GPU的体系结构与CPU有着明显的区别。而要优化GPU的程序效率，则需要重点在GPU的访存效率以及数据传输效率着手。

**三、对GPU访存效率提升技术的研究与优化**

随着图形处理单元（GPU）在高性能计算和数据密集型应用中的广泛应用，GPU访存系统的优化变得越来越重要。GPU的性能受到访存系统的影响，而访存系统的效率又直接影响着GPU的整体性能。因此，研究人员对GPU访存系统进行了广泛的研究，旨在提高访存效率，从而加速GPU应用程序的执行。

目前，对GPU访存系统的研究已经取得了一些重要的成果。这些成果主要包括利用数据局部性、分块访问、多面体模型等方法来提高访存效率，以及通过改进存储控制器、动态管理显存等方式来优化GPU的存储系统。然而，这些方法和技术各有优缺点，并且针对不同类型的应用程序可能存在适用性的差异。因此，对于特定的访存密集应用，我们需要深入研究并结合现有的技术和方法，提出针对性的访存优化策略，以提高GPU的性能和整体效率。

现代计算机内存系统主要可以针对两个方面进行优化，一方面是降低内存访问延时；另一方面是提高内存访问带宽，GPU内存系统的优化也是如此。接下来的部分将从降低内存访问延迟和提高内存访问带宽两个方面进行研究，讲解如何通过提升GPD的访存效率来突破GPU程序效率的瓶颈。

**3.1 降低内存访问延时**

本小节将讲解如何通过降低内存访问延时来提升GPU访存效率。

（一）GPU访存子系统

GPU访存子系统是GPU架构中的关键组成部分，旨在满足高并行性的需求，并提供高带宽和低延迟的数据访问。它由片上存储和片外存储两部分构成。

片上存储包括多级缓存、本地共享存储和寄存器。每个流多处理器都有自己的缓存和本地共享存储器，用于存放线程块内的共享数据和重复使用的数据，以减少对片外存储的访问次数。本地共享存储器具有较高的带宽，而寄存器速度非常快。GPU的缓存一般由两级构成，每个流多处理器拥有自己的缓存。

片外存储包括全局存储、常量存储和纹理存储，它们共享显存容量。片外存储的访问延迟较大，需要较长的时间。为了提高访存效率，显存控制器的设计非常重要。

整个GPU访存子系统的设计目标是通过合理的存储器层次结构和数据迁移策略，最大程度地减少对片外存储器的访问次数，提高数据重用率和整体访存效率。这样可以满足GPU架构对高性能和高吞吐量的要求。

1. 数据访问优化技术

基于访存模式的高效率数据访问是针对GPU平台上的数据访问优化的一种方法。GPU采用并行处理方式，可以隐藏访存延迟，但当大量线程数据饥饿时，计算资源会处于空转状态，无法充分利用。因此，在将应用程序移植到GPU平台上时，需要对数据访问模式进行分析，并利用数据访问优化技术避免重复访问或减少访问延迟，以确保活动线程的数据供给。

1. 数组访存模式

数组是科学计算中最常用的数据结构，其访问方式可以分为规则和不规则两类。通过对应用程序进行分析，可以得出其访存模式的形式化表达。通过将数组访存方式进行组合变换，可以得到多种不同的数组访问情况，并可以利用多面体模型对其进行抽象化表示。

多面体模型是一种代数框架，通过迭代域、访问函数和仿射调度三种操作来表示程序的迭代循环结构。通过对循环边界和数组访问与循环索引和全局参数之间的映射关系进行建模，可以定义一个在多维空间中的迭代空间多面体。循环体对数组的访问可以用齐次方程表示，其中访问矩阵表示循环体对数组的访问方式。

1. 应用程序数据的访存模式

访存优化技术可以分为避免存储访问延迟、减少冗余访问和隐藏存储访问延迟三类。在分析应用程序数据的访存模式时，可以利用多面体模型得到数据的访存矩阵，并通过比较访存矩阵的秩和循环迭代空间的维数来判断数组是否存在重复访问。

通过对访存模式的分析，可以重新构造程序的迭代空间，减少对全局存储的访存次数，从而提高访存效率。具体方法可以是创建临时变量、重组数据访问顺序等，以减少对全局存储的冗余访问。这样可以提高数据局部性，缓解存储墙问题，并进一步优化GPU上的数据访问性能。

1. DPoMAP算法

因此，由学者提出了一种基于访存模式的高效率数据布局算法，命名为DPoMAP（Data Placement Based on Memory Access Patterns）。该算法旨在根据应用程序的访存模式来确定数据在内存中的布局位置，以优化GPU的访存性能。

具体来说，算法的输入包括平台类型、程序迭代空间的维数D、数组个数n、数组的迭代矩阵F、迭代矩阵的秩R、ArraySize数组大小信息、寄存器文件个数RF和片上共享存储容量SM。输出为各个数组的内存位置Placement。

算法的主要思想是根据平台类型和访存模式，将数据放置在不同的存储位置上，包括片上共享存储（SM）和寄存器文件（RF）。算法通过维护SM\_Occupancy和RF\_Occupancy两个变量来跟踪已占用的SM和RF资源。

在每个数组的迭代维度上进行循环（i为数组索引，Ri为数组的秩），根据平台类型进行条件判断。对于NVIDIA平台，首先检查行重用（F[i]行重用）和SM资源是否可用，如果满足条件，则将数组放置到SM中，并更新SM\_Occupancy。如果不满足行重用条件，则检查列重用（F[i]列重用）和SM资源是否可用，如果满足条件，则将数组放置到SM中，并更新SM\_Occupancy。如果以上两种情况均不满足，则将数组放置到寄存器文件中。

对于AMD平台，只考虑列重用和SM资源的可用性。如果满足列重用条件，则将数组放置到SM中，并更新SM\_Occupancy。如果不满足列重用条件，但SM资源可用，则将数组放置到SM中。如果以上两种情况均不满足，则将数组放置到寄存器文件中。

算法的目标是通过将具有重用模式的数组放置到SM中，尽可能提高SM的利用率，并减少对全局内存的访问。同时，将不具有重用模式的数组放置到寄存器文件中，以减少对存储器的访问延迟。

**3.2 提高内存访问带宽**

现代计算机内存系统主要可以针对两个方面进行优化，一方面是降低内存访问延时；另一方面是提高内存访问带宽。而对于GPU内存系统中的则更侧重于提高访问带宽。为此，现代GPU内存系统被划分为多个可并行访问的内存通道，并且系统将内存地址通过某一特定的地址-通道映射规则映射到不同的内存通道，从而提高整个GPU内存系统的并行访存能力，进而提高GPU的带宽。

通常来说，一系列连续内存访问的的高位变化较少，相反，地址的地位变化较为频繁，因此在计算机系统结构设计时，通常倾向于选取内存地址中较低的部分位用于选择内存通道。但是在实际应用的过程中，由于特定的GPU访存的地址序列，访问的内存地址通过映射转换为通道时，大量访问的请求会集中于同一个通道，导致该通道的冲突，而其他通道处于相对空闲的状态。如此导致有效带宽在实际上仅仅为最大带宽的一部分。

1. 伪随机交织方法
2. 什么是伪随机交织方法

为了解决上述访问请求大量集中于同一个通道的冲突问题，曾经有学者提出内存地址交织(伪随机交织)方法对此问题进行改进。其中主要思想为：在选取用于确定内存通道的若干位的同时（这里称为channel bit）也选取内存地址中的其他位（这里称作bit——用于与channel bit进行异或操作），从而得到新的用于确定内存通道的地址。在统计学的基础上，此处的异或操作是对每次访存的请求所映射到的内存通道呈现伪随机性，是GPU原始访问模式映射为在内存通道中分布得更加均匀的访问模式，从而保证最大程度地提高每个内存通道的利用率。

1. 伪随机交织方法的缺点

但是随着现代GPU执行的任务类型逐渐增多且复杂，上述固定的内存地址交织方法无法满足所有任务类型的需求。在大多数情况下，上述方法使得在一段时间内的内存访问请求在不同通道之间是相对均匀的，但是对于不同的内存地址交织方式可能导致内存地址的重叠，从而产生访存错误；并且GPU一旦启动，固定的内存地址交织方式便无法修改。

1. 基于扩展页表的优化方法

基于扩展页表的优化方法根据不同任务标识选择不同映射方式，关于如何选择选择最佳的映射方式，这里通过信息判断——即产生信息熵最大的那种。因此有学者提出一种基于扩展页表的GPU访存自适应优化方法。该方法可以解决同时存在多种映射、或针对多种应用选取不同映射方式时，内存地址重叠所导致的错误。其核心思想为：通过扩展页表内容，加入任务类型标记，使每一个应用可获得定制化的访存映射方式，可以在保证访存的正确性的同时，实现在GPU运行时候根据应用进行针对性的访存优化。

通体来说，解决思路包括以下步骤：

1. 逻辑运算单元ALU接收GPU应用所发送的完成任务所需的指令，注意到其中需要包含任务类型的相关信息；然后根据指令完成相应的操作，并在访问内存时向内存管理单元发起虚拟地址请求。
2. 内存管理单元MMU根据所接受的虚拟地址请求，通过扩展页表将虚拟地址转换为物理地址，然后将物理地址发送至地址转换单元。同时，基于其中的任务类型信息，系统将查找GPU应用对应的所属任务类型标识，将所述任务类型标识发送至映射方案配置寄存器。其中所属任务类型标识记录在内存管理单元的扩展页表中。
3. 映射方案配置寄存器根据所述任务类型标识查找对应的最佳访存映射方式，然后将所述最佳访存映射方式发送至地址转换单元
4. 地址转换单元根据所述最佳访存映射方式，将所述物理地址映射为新的映射后地址，将所述的新的映射后地址发送至内存子系统，基于所述新的映射后地址访问内存子系统。

一下是对几个方面的展开说明：

* 1. 页表为了兼容不同应用程序在同一个GPU上运行而不产生地址冲突，GPU软件在访问内存时统一使用从0开始的虚拟地址，然后再通过页表转换为物理地址。其中，物理内存以通常以4KB的大小分为若干页，然后通过页表将虚拟地址上4KB的地址空间映射到物理地址上的某个4KB的地址空间。
  2. 同时，程序在GPU启动时，将获得一个任务类型标识。内存管理单元需要根据任务类型信息查询出访存请求所对应的任务类型标识，并且记录于分配给该程序的页表项中，从而区分不同类型的任务的映射方式。于是在映射为物理地址的同时，任务类型标识也被传入地址伪随机交织模块，从而将原始访存模式映射为该应用的定制化的最佳访存模式。
  3. 这里通过利用不同的访存映射方式，将GPU访问请求进行不同的内存地址映射，然后记录GPU最大等待队列的长度作为一个时间段，基于这个时间段确定多个统计时间段，在每个时间段中统计GPU访问请求在不同访问通道上的分布信息熵，然后取得该时间段的平均值，从而确定某种映射后GPU访问请求对应的信息熵，最后将最大信息熵对应的访存映射方式作为最佳访存映射方式。这里信息熵的衡量了一组消息携带信息的量：E = Σ(-pi\*log2(pi))，即消息传输中每种可能的消息i的概率pi的负对数之和。

**四、对GPU数据传输速率的研究与优化**

**4.1 PCIe总线**

1. 使用PCIe总线的原因

现代计算机的核心组件是内存和CPU，外围设备如键盘、显示器和磁盘提供输入/输出(I/O)功能。尽管这些设备（也称为外围设备）缺乏聚合数据或执行逻辑操作的能力，但它们在与人类交互、提供数据持久性、在多个 CPU 之间交换数据以及执行专门任务方面发挥着至关重要的作用。例如，显示器和键盘促进了人机交互，而磁盘则为CPU执行的结果提供了存储空间。网卡可以实现多个CPU之间的数据交换，而GPU可以帮助CPU完成数值计算任务，尤其是图形相关的计算。然而，由于多个外设需要与一个或几个CPU交换数据，后者成为一种稀缺资源。因此，需要一种协议来规范所有外设和CPU之间的通信。在IntelX86架构中，早期的PCI协议就是为这个目的服务的。

（二）PCIe总线外形

对于一般用户来说，PCIe对用户可见的部分就是主板上大大小小的PCIe插槽。

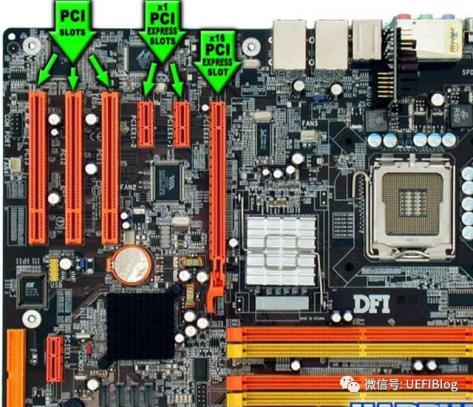


图2 PCIe总线的外形

1. PCIe总线发展历史

作为扩展接口，它主要用于外围设备的连接和扩展，而外围设备吞吐速度的提高，往往会倒推接口速度的提升。接口技术逐步迭代，按时间可分为从ISA、MCA、EISA、VLB、PCI、PCI-X、AGP、PCI Express，这里不再过多探述。

1. PCI和PCIe架构

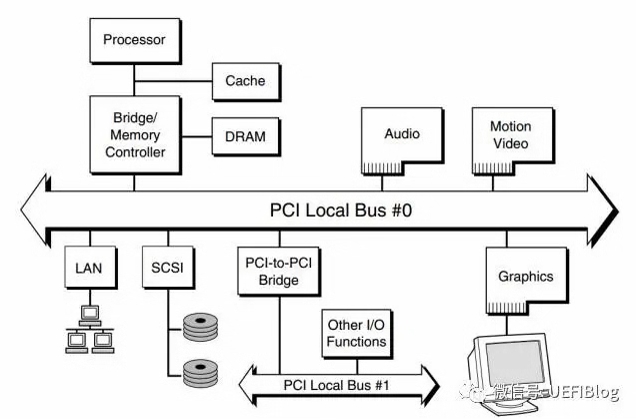


图3 典型的PCI总线树

以上是一颗典型的PCI总线树。各种外设以树的形式挂载到PCI总线上，与CPU或其他外设通信。当一条PCI总线的承载量不够时，可以用新的PCI总线进行扩展，而PCI桥则是连接PCI总线之间的纽带。

PCI总线是个并行总线。在一个时钟周期内32个bit（后扩展到64）同时被传输。地址和数据在一个时钟周期内按照协议，分别一次被传输。引脚定义如下：

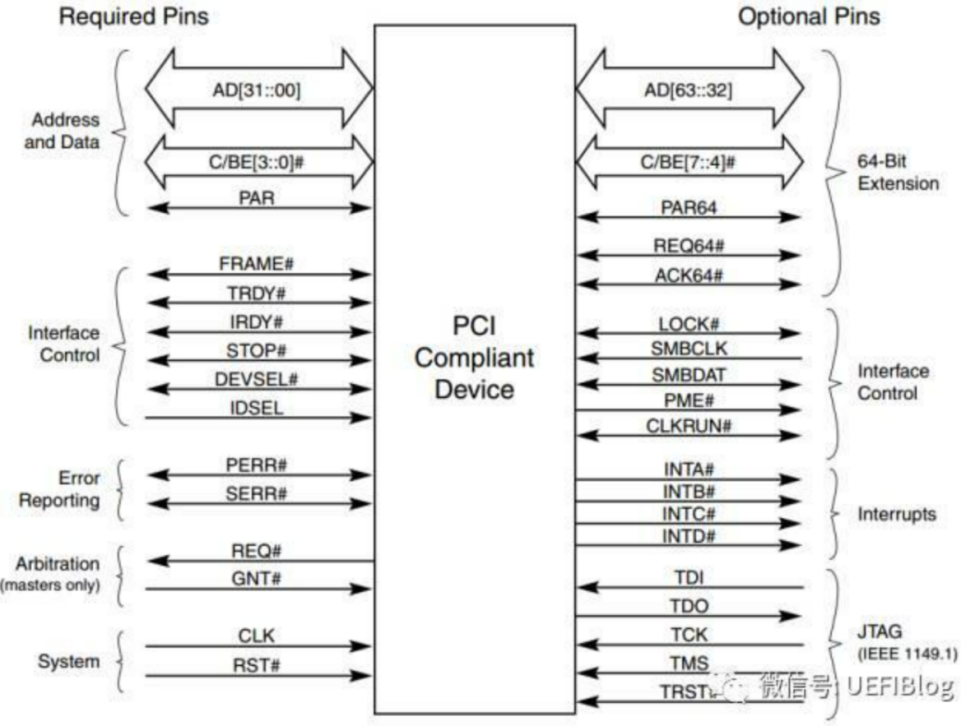


图4 PCI引脚定义

PCI后期越来越不能适应高速发展的数据传输需求，随着频率的提高，PCI并行传输遇到了干扰的问题：高速传输的时候，并行的连线直接干扰异常严重，而且随着频率的提高，干扰越来越不可跨越。PCIe和PCI最大的改变是由并行改为串行。通过提高在一个周期内传输的数据量，在整体效果上PCIe的数据传输速度反而超过了PCI的传输速度，并且频率的降低使得结构更加简单，线路可以更长。

另外一个区别在于使用了点对点的结构，而PCI是总线结构。一个典型的PCIe系统结构图如下：

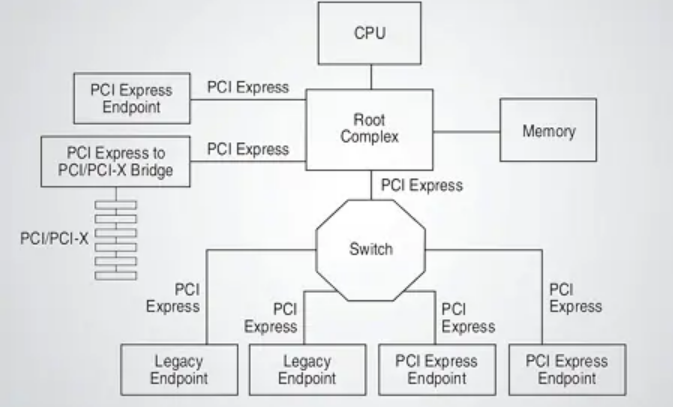


图5 PCIe系统结构图

点对点意味着每一个PCIe设备都拥有自己独立的数据连接，设备之间数据传输并发互不影响，而对于PCI共享总线方式，PCI总线上只能有一个设备进行通信，一旦PCI总线上挂接的设备增多，每个设备的实际传输速率就会下降，性能得不到保证。

1. NUMA（非一致性内存访问）架构

原来所有内存的访问都需要通过北桥芯片来完成，而面对着速度不断提升的需求，越来越多的核心被尽可能地塞进了同一块芯片上，各个核心对于内存带宽的争抢访问成为了瓶颈，由此推出了NUMA架构：

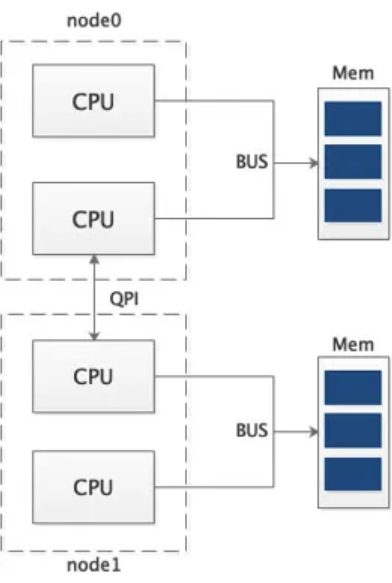


图6 NUMA架构

CPU芯片被划分成不同的组，称为node，每个node有自己独立的内存访问地址，每个node也就有自己的内存单元。

（六）结合NUMA和PCIe的结构

CPU被划分成了多个组，PCIe的结构由点对点的结构树组成，简易的现代服务器的物理连接结构图就形成了：

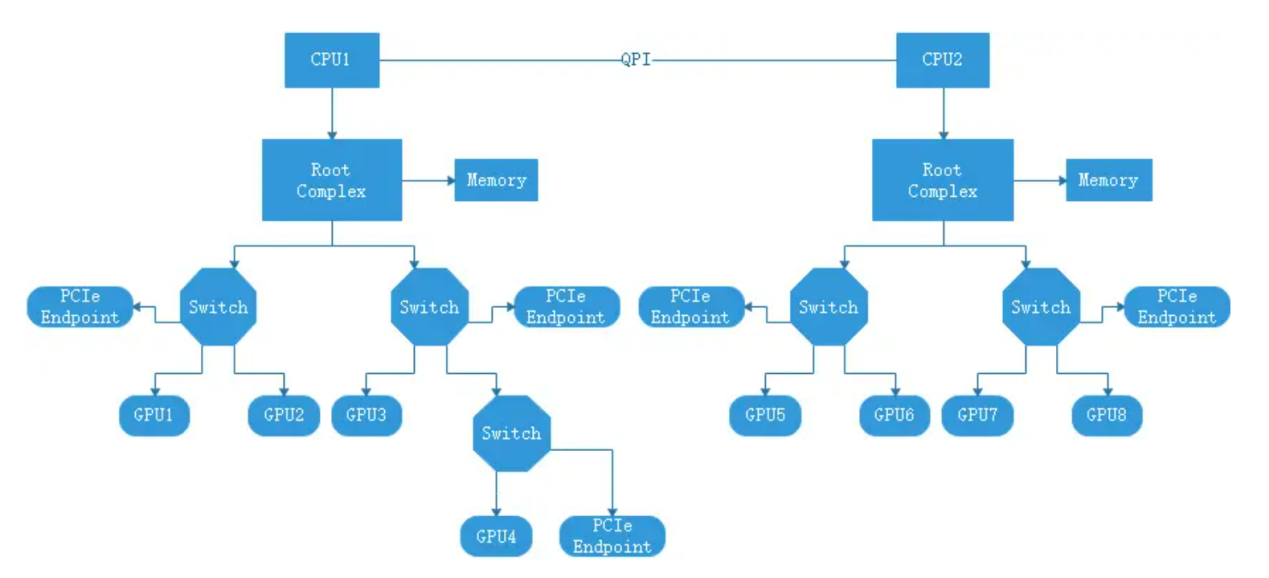


图7 现代服务器的物理连接结构图

这种将内部数据划分片区的方法，减少了不同片区之间的冲突和污染，使得单独某个片区内部的数据传输效率大大提高，可以看到在物理结构上连接越近的结构，相互之间的数据传输速度越快，距离越远效率越低，此处的效率一般体现在通信时延上。

**4.2 CPU与GPU的异构多核架构**

GPU具有良好的并行编程模型，擅长处理大量并行计算，特定是在AI模型训练领域发挥了至关重要的作用，可以弥补CPU计算的不足。

虽然将CPU和GPU组合在一起的PCI－ｅ总线方式，可以极大的提升数据传输的效率，但由于其存储系统分开设置，具有低带宽高延迟的特点，GPU的执行速度仍受制于CPU的数据处理能力。

目前一个主流的架构是将CPU和GPU集成到同一块芯片上进行组成异构多核架构。此建构适用于处理高性能计算和大数据时代背景下的海量数据，使得CPU和GPU间的数据传输更加便捷。区别于同构处理器，异构处理器每个内核有不同的架构，针对不同的需求而设定，可以提高应用的计算性能或实时性能。其中逻辑控制和串行执行由CPU处理执行，需并行执行的大数据则由GPU处理。

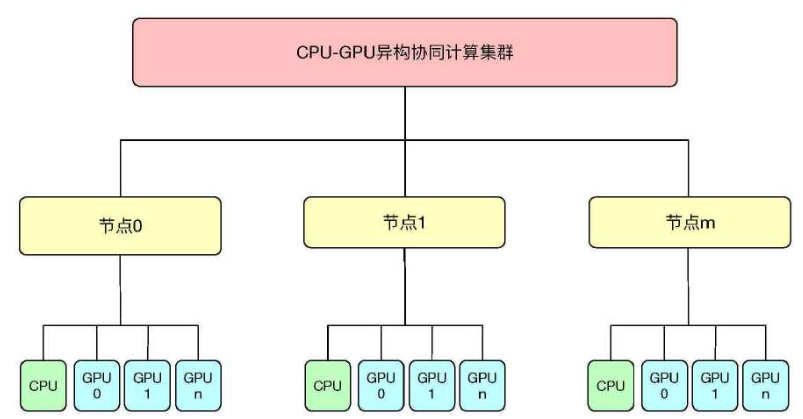


图8 CPU－GPU异构体系架构图

英特尔、英伟达和AMD都在向CPU＋GPU＋FPGA／NPU的方向靠拢，原因在于，在具备CPU＋GPU的计算架构基础上，FPGA的可编程特质能提升计算平台的灵活性，以适应AI时代到来对不同工作负载加速的需求。

异构架构仍面临不小的挑战：由于CPU和GPU拥有不同的设计结构，访存行为各具特征，增加了CPU的deadline缺失率；由于CPU程序在时间和空间上具备程序性原理，对缓存容量大小较为敏感，而GPU程序的并行执行对缓存容量变化反映并不明显，但GPU的高并发又使得在传统的LRU算法下，GPU会占据大部分共享缓存空间，迫使CPU的性能受到局限。处理好异构并行计算体系中各节点内CPU和GPU的数据传输任务成为了突破瓶颈的关键因素。

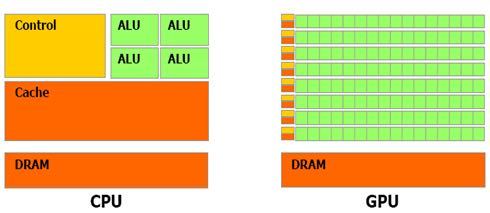


图9 CPU和GPU的设计结构对比

MPI、OpenMP是目前较为流行且行之有效的解决方案。MPI（Massage Passing Interface）是消息传递函数库的标准规范，具有消息传递式并行程序设计和并行计算粒度大的特点，不同进程间通过消息交换数据。OpenMP（Open Multi-Processing）可用于显示地指示多线程、共享内存并行性。对于高性能计算应用程序，二者通常相互结合以实现分布式内存并行，OpenMP用于每个节点上的计算密集型工作，MPI用于实现节点间的通信和数据共享。

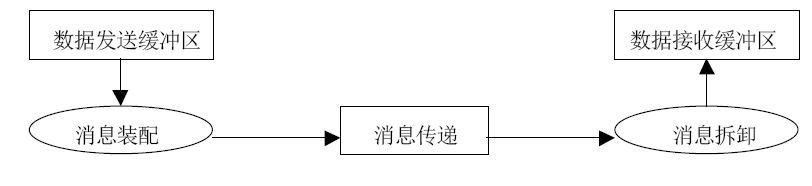


图10 MPI消息传递过程

CPU和GPU的数据通信在独立内存阶段，GPU程序执行前后数据需要显式地在二者间传递，数据传输和处理串行化且编程复杂；在引入统一虚拟地址阶段，远端直接访问的数据仍在原设备段内存，GPＵ无法利用数据局部性，整体性能无显著提升；而后提出的统一内存方式，简化编程并优化性能，实现自动整页迁移。

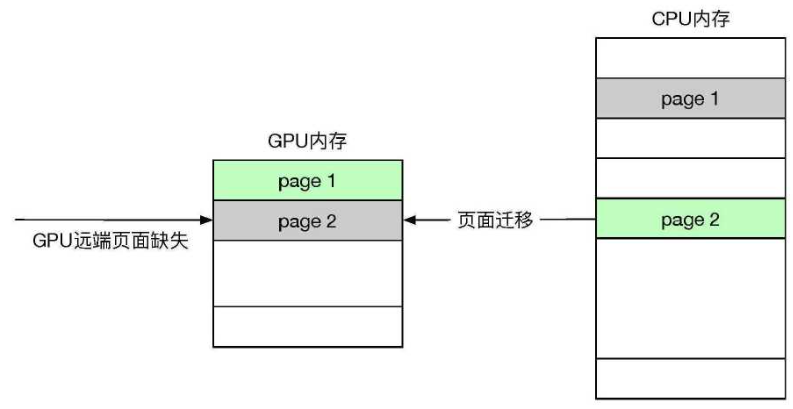


图11 GPU端发生远端页面缺失时系统自动进行页面迁移

整页迁移的方式在页面变大时存在性能风险，而Lusing等人通过重叠已传输数据的计算与剩余数据的传输来提高性能，但此方法对程序员不透明，下面将介绍新提出的较为高效可行的方法——自动部分页迁移。

在整页迁移中，有3种页面状态：有效、CPU种有效和GPU中有效。为了执行部分页迁移，添加部分有效的页面转换状态，记录页面的有效范围并根据页面迁移操作实时修改，并对迁移过程的管理进行修改，以实现单一有效范围的部分页迁移和多个有效范围的部分页迁移。此方法对整页迁移有显著优势，既能在高性能工作负载时满足其对于大页面的需求，也可以避免页面增加时可能出现的性能下降。

CPU和GPU的通信严重制约运算速度，仍需对其进行优化，从理论角度可以从下述方面进行进一步优化：

从硬件层面：

1. 数据预取和缓存技术：利用程序的局部性原理提前将数据从内存加载到缓存中，避免无效的等待时间，从而降低延迟并减少数据传输时间。
2. 共享内存：CPU和GPU往往采用不同的内存架构，而一块共享的物理内存空间可以便于数据的传输。
3. 数据对齐：CPU和GPU进行数据传输时，数据块的大小可能不同，提前进行数据对齐，可以降低对数据的处理时间。
4. DMA传输：CPU的工作较为繁忙且响应速度低于GPU，通过直接传输可以绕过CPU直接将数据通过内存传输给GPU，降低CPU的负载，提高传输效率和整体工作效率。

Cache优化：缓存在数据传输过程中意义重大，优化缓存可以降低数据的复制次数和降低数据传输的延迟，从而提高传输效率。

下面将从软件和算法的层面

1. 通过数据压缩和解压缩减少数据传输的大小：可以减少数据的传输规模，从而提高传输效率。比如先定量化再压缩的无损压缩算法ZFP算法和先预处理再压缩的有损压缩算法SZ算法都能在一定的压缩比例下保持较高对数据进行压缩。
2. 采用分布式计算方法：在处理大规模数据的计算任务时，将数据细分，然后分布在不同的节点上进行计算，可以有效减少数据传输的规模和次数，提高计算效率，弥补在CPU和GPU之间的通信不足问题。
3. 对数据的局部性进行优化：CPU程序在时间和空间上的局部性原理使得其对访存容量较为敏感，所以在CPU和GPU的通信时，可以优化数据的局部性，相邻的数据在数据块中靠得比较近自然也就可以减少数据传输的时间，提高传输效率。
4. 对数据流水线进行优化：和指令的流水执行类似，采用数据流水线技术将数据的处理过程细分为多个不同阶段，并且在每个阶段之间添加缓冲区，减少阻塞和等待的时间。

**五、小 结**

以上便是本文的全部内容，本文针对GPU程序效率瓶颈中的访存效率和GPU数据传输速率的矛盾，提出了一系列优化方案和策略。通过合理地设计和实施这些优化技术，可以显著提高GPU程序的执行效率和性能。

然而，GPU技术的发展仍在不断演进，未来的研究方向包括进一步优化访存架构和数据传输机制、改进GPU与主机之间的通信效率以及探索新的硬件架构和编程模型等。随着GPU技术的不断成熟和创新，我们相信GPU将继续在科学计算、数据处理和人工智能等领域发挥重要作用，并为计算性能和应用性能提供更大的突破和创新空间。

**六、参考文献**

1. 汪梦萱. CPU-GPU异构架构下共享内存管理策略的研究[D].北京工业大学,2020.DOI:10.26935/d.cnki.gbjgu.2020.000351.
2. 张诗情,杨耀华,沈立,王志英.通过部分页迁移实现CPU-GPU高效透明的数据通信[J].计算机工程与科学,2019,41(07):1168-1175. [通过部分页迁移实现CPU-GPU高效透明的数据通信 - 中国知网 (cnki.net)](https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=cjfd2019&filename=jsjk201907004&dbcode=cjfq)
3. 刘士建.(2018).基于CPU-GPU异构架构下Cache优化技术的研究(硕士学位论文,北京工业大学)
4. 张瑞,田密.利用访存模式构建GPU高效率数据访问[J].延安大学学报(自然科学版),2020,39(03):30-36.DOI:10.13876/J.cnki.ydnse.2020.03.030.
5. 赵彦博. 基于访存特征分析的GPU程序数据布局及性能优化技术研究[D].山东大学,2022.DOI:10.27272/d.cnki.gshdu.2022.004287.
6. 陈道琨,杨超,刘芳芳等.面向GPU平台的并行结构化稀疏三角方程组求解器[J/OL].软件学报:1-11[2023-05-16].https://doi.org/10.13328/j.cnki.jos.006720.
7. 李松林，孟平凡，时昊，刘杨，李然月.基于扩展页表的GPU访存自适应优化方法及装置[P].中国专利：CN114185818A,2022.03.15
8. 老狼.(2017).深入PCI与PCIe之一：硬件篇.知乎.检索与2023.5.10，来源<https://zhuanlan.zhihu.com/p/26172972>
9. 函谷叨客.(2021). 【研究综述】浅谈GPU通信和PCIe P2P DMA.知乎.检索于2023.5.10，来源<https://zhuanlan.zhihu.com/p/430101220>

[10]迪捷软件.(2022).带你了解PCIE通信原理.知乎.检索于2023.5.10，来源<https://zhuanlan.zhihu.com/p/454282470>

**七、成员贡献**

**2051498 储岱泽(2051498@tongji.edu.cn）：**关于GPU的介绍以及体系结构的分析、文档整理，PPT制作。（100%）

**2051828 莫益萌(2051828@tongji.edu.cn)：**对GPU访存效率提升技术的研究与优化——降低内存访问延时（100%）

**2151298杨滕超(2151298@tongji.edu.cn）：**对GPU访存效率提升技术的研究与优化——提高内存访问带宽（100%）

**2152810 刘 冲(2152810@tongji.edu.cn)：**对GPU数据传输速率的研究与优化——PCIe总线结构（100%）

**2153174 陈华机(2153174@tongji.edu.cn)：**对GPU数据传输速率的研究与优化——CPU与GPU异构多核架构（100%）