题目

作者名[1)] 作者名[2),3)] 作 者[3)]

[1)](单位全名 部门 (系) 全名, 市 (或直辖市) 国家名 邮政编码)
[2)](单位全名 部门 (系) 全名, 市 (或直辖市) 国家名 邮政编码)
[3)](单位全名 部门 (系) 全名, 市 (或直辖市) 国家名 邮政编码)

摘 要 中文摘要内容置于此处 (英文摘要中要有这些内容)，字体为小 5 号宋体。摘要贡献部分，要有数据支持，不要出现“... 大大提高”、“... 显著改善”等描述，正确的描述是“比…提高 X%”、“在…上改善 X%”。

关键词 关键词；关键词；关键词；关键词
中图法分类号 TP391 DOI 号 10.11897/SP.J.1016.01.2021.00001

Title

NAME Name-Name[1)] NAME Name[2),3)] NAME Name-Name[3)]

[1)](Department of ****, University, City ZipCode, Country)
[2)](Department of ****, University, City ZipCode)
[3)](Department of ****, University, City ZipCode, Country)

Abstract Abstract (500 英文单词，内容包含中文摘要的内容).

Key words key word; key word; key word; key word

1 引言

近年来，信息技术的迅猛发展和普及应用所带来的数据爆炸性增长，通用计算领域图形处理器 GPGPU(General-Purpose Computing Graphics Processing Unit) 由于其强大的并行计算能力、高吞吐率以及高性价比, 已经成为高性能计算领域的主流加速器。基于 GPGPU 的计算加速成为了高性能计算、机器学习和数据分析应用的主要工具。这些领域的进一步发展依赖于 GPU 性能的进一步拓展，直到近几年，GPU 一直通过晶体管数量的增加来增加 SM 的数量从而来拓展性能。

但是，随着晶体管缩放速度的放缓，晶体管密度增长速度放缓，同时随着 GPU 接近最大芯片尺寸的限制，单 GPU 性能拓展速度进一步放缓。同时，伴随着数据量的爆炸性增长，很多应用程序已经能够完全占用单 GPU 上的计算资源，这需要 GPU 系统性能的进一步拓展来满足人们的需要。通过将多个单片 GPU 拓展为多 GPU 系统，通过系统集成来提升应用程序性能是一种有效的解决方案。目前系统正逐渐向 CPU 与多 GPU 互联的异构系统拓展。

但是多 GPU 异构系统带来了一系列问题。例如，跨 GPU 的工作分配不容易透明地完成，同时在工作分区、负载平衡和同步方面也面临着一系列挑战。此外，多 GPU 的性能很大程度依赖着多级系统的互联，沿着这些互联传递的同步信息和数据移动显著影响着多 GPU 系统的整体性能和功效。本文重点关注了多 GPU 系统中存在的非对称内存访问的现象，主要包括以下几个方面：（1）GPU 内存架构，以此为基础所存在的三种非对称内存访问的问题；（2）CPU 与 GPU 互联的带宽差距以及解决方案，主要是 GPU 初始化时的数据布局和线程调度；（3）GPU 内部多级内存的非对称内存访问的解决方案，主要包括数据布局，线程调度和互联网络优化；（4）多 GPU 互联的非对称内存访问的解

收稿日期：2021-03-30；修改日期：2021-03-30 本课题得到……基金中文完整名称 (No. 项目号)、……基金中文完整名称 (No. 项目号)、……基金中文完整名称 (No. 项目号) 资助. 作者名，性别，xxxx 年生，学位 (或目前学历)，职称，是/否计算机学会 (CCF) 会员（提供会员号），主要研究领域为 *****、****. E-mail: **********. 作者名，性别，xxxx 年生，学位 (或目前学历)，职称，是/否计算机学会 (CCF) 会员（提供会员号），主要研究领域为 *****、****. E-mail: **********. 作者（**通信作者**），性别，xxxx 年生，学位 (或目前学历)，职称，是/否计算机学会 (CCF) 会员（提供会员号），主要研究领域为 *****、****. E-mail: **********.
第 1 作者手机号码：……，E-mail: **********

决方案，主要包括封装架构，数据布局以及线程调度等方法；（5）目前的发展特点以及未来的坊站方向。本文的第 2 节介绍 GPU 的内存架构，特别是他们之间的带宽差距；第 3 节介绍了几种非对称内存访问的解决方案；第 4 节介绍了目前的主要挑战和未来的发展方向。

2 GPU 内存架构

随着单 GPU 向着多 GPU 拓展，逐渐形成了 CPU 与 GPU 构成的异构系统，其中 CPU 可以与 GPU 进行数据传递，不同 GPU 之间也可以相互通信。而数据传递的带宽差距是导致非对称内存访问的主要原因。为了更好的对 GPU 异构系统的非对称内存访问进行研究，需要对 GPU 的内存架构进一步了解。在 GPU 异构系统中，存在三方面的主要通信，分配内存和线程时的 CPU 与 GPU 的通信，GPU 内部数据的移动，以及 GPU 之间的数据传递因此在这种系统中，存在三种可能导致 NUMA 现象的行为。

（1）由于 CPU 和 GPU 的存储器是分离的，CPU 端的内存被称为 Host（主机）内存，GPU 端的内存被称为 Device（设备）内存。如果需要在 GPU 端进行计算，就需要把待处理的数据拷贝到 device 内存中，待数据处理完成后，还需要把计算结果拷贝到 Host 端做进一步的处理，比如存储到硬盘中或者打印到显示器上。为了完成这一系列操作，需要 CPU 与 GPU 之间进行数据传递，但是 CPU 与 GPU 互联的带宽很低，这导致了非对称内存访问的产生。

（2）对于单 GPU 来说，数据存在在 DRAM，为了加速，还有 L1 和 L2 的 cache，这导致的 NUMA 现象。

（3）对于多 GPU 来说，数据包含 local 和 remote 两种，这导致的 NUMA 现象。

图带宽差距比对。

NUMA 现象在 CPU 已经出现过，在 GPU 中的不同点。本文主要介绍 GPU 的 NUMA 现象的解决策略。

3 GPU 的 NUMA 现象

3.1 CPU 与 GPU 互联

对于 GPU 来说，首先需要 CPU 来进行数据分配。

3.1.1 数据放置

CPU 和 GPU 上的异构计算传统上对每个设备使用固定的角色：GPU 通过利用其大量内核来处理数据并行工作，而 CPU 处理非数据并行工作，如顺序代码或数据传输管理。

由于使用 GPU 运行的主要是并行的应用程序，为了减少数据移动，数组上线界确认移动。

但是数组上下界准确率，提出了计算网格分解一维，同时充分考虑 DRAM 的共享。

扩展到多维，增加对内核外数据的考虑

同时，由于 GPU 发生页故障，需要 PCIe 向 CPU 相应，提出了远程数据的 MSHR 以及数据预取机制。

3.1.2 互联网络

拓扑结构，利用多 GPU 的通信模式和覆盖网络

3.2 GPU 内部

在 GPU 内部中，由于速度不匹配，人们更希望能在本地的 L1 cache 中获得自己所需要的数据，而不是通过互联网络前往更低级 cache。因此，我们从程序，数据，以及传输三种角度来考虑。

3.2.1 数据迁移

页面分配

对于单 GPU 来说，由于 L1 cache 的容量有限，只能将程序所需要的部分重要数据放在 L1 cache 中。对于程序所需要的数据，需要对数据进行静态分配或者对数据进行动态迁移到使用数据的内核中。

首先 GPU 上运行的主要是并行程度高的应用程序，循环和数组，可以通过对程序进行预先分析，从而实现对数据的静态分配。介绍论文《Compiler Support for Selective Page Migration in NUMA Architectures》通过编辑器分析程序中的循环和数组变量。

但是静态分配无法分析 GPU 在运行过程中产生的信息，比如说数据迁移和计算放置等信息，这可能会产生一定的错误分配从而导致性能下降。过去曾经在 CPU 提出过的动态页面迁移，在 GPU 中很难取得很好的效果，在 GPU 中进行动态页面分析和迁移的成本可能非常高，并且 GPU 中进行页面动态迁移可能导致的页面锁，流水线刷新，提高开销。

数据保护

由于 L1 cache 的容量有限，对于已经进行页面

分配存放到 L1 cache 中的数据来说，可能会由于未来访问的数据而被替换掉，这些数据如果在未来被使用，就需要重新到更低级 cache 中访问，从而导致性能下降。为了保护数据的局部性，从而保证数据复用，需要采取一定的策略来进行数据保护。

对于 cache 来说，为了预防抖动导致可能会复用的数据被刷新，因此需要采取合适的策略来保护具有空间和时间局部性的数据。

介绍论文《Access Pattern-Aware Cache Management for Improving DataUtilization in GPU》缓存保护（得知复用的周期），缓存旁路等方法。

同时，传统的 L1 cache 是每个 SM 所私有的，但是实验证明，大多数内核所需要的数据可以从其他内核中找到。共享 L1 cache。

3.2.2 线程调度

为了更好地对数据进行复用，除了对数据进行调度外，还可以通过对线程调度来实现。

指令链卸载到存在数据的 GPU 之中。PIM，介绍论文《2016 Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities》

更好的利用 cta 间的局部性，从而复用 L1 cache。介绍论文《2017 Locality-Aware CTA Clustering for Modern GPUs》

3.2.3 互联网络

对于 L1 cache 中发生失效的数据，必须通过互联网络发送到 L2 中进行访问。

增加带宽

虚拟通道，轮询使用

减少阻塞

对于 GPU 内部来说，内核发出的内存请求的数据如果在 L1 中失效，会通过互联网络将内存请求发送到 L2 中。MC 的分配，介绍论《Bandwidth-efficient On-chip Interconnect Designs for GPGPUs》

3.3 GPU 互联的 NUMA 现象

当单 GPU 向多 GPU 拓展时，可能导致的 NUMA 现象。多 GPU 的情况与单 GPU 的情况类似，但是相对于单 GPU 来说，复杂性。因此为了解决 NUMA 现象，无法从提高互联的带宽角度来入手，这依赖于工艺水平的提高，但是可以从多 GPU 封装结构，以及数据复用和线程调度来解决。

3.3.1 封装架构

对于多 GPU 来说，未来的拓展方向主要有两个

将单 GPU 划分为多个 module

将多个 GPU 通过交换机连接起来

通过 API 来自由选择 GPM 和多 GPU 互联

3.3.2 数据放置

页面迁移

对于多 GPU 来说，页面一开始是通过 CPU 来进行分配的，为了减少多 GPU 之间通信。

首先数据分解

保证数据结构对齐

基于带宽的页面分配

在程序运行过程中，通过页面迁移，包括数据预取等

数据保护

多 GPU 之间共享数据，划分 DRAM 的 cache，以及缓存旁路（确定那些数据是远程数据）等方法来提升性能。

3.3.3 线程调度

全局 CTA 调度，减少 GPU 之间的通信

4 挑战与展望

目前 GPU 的 NUMA 仍面临很多挑战

（1）与理想情况仍有性能差距，多 GPU 互联性能超过线性相加的性能。

（2）保持 cache 一致性问题

未来的发展方向

（1）提出更合理的多 GPU 封装方法

（2）更好地结合软硬件方法，提升性能

（3）保持 cache 一致性的基础上降低通信开销