临沂大学博士教授文库
LINYIDAXUE BOSHI JIAOSHOU WENKU

图形处理器绿色通用计算关键技术

TUXING CHULIQI LVSE TONGYONG JISUAN GUANJIAN JISHU

王海峰 著

山东人民出版社
全国百佳图书出版单位 国家一级出版社

临沂大学博士教授文库

LINYIDAXUE BOSHI JIAOSHOU WENKU

图形处理器绿色通用计算关键技术

王海峰　著

山东人民出版社

全国百佳图书出版单位　国家一级出版社

图书在版编目(CIP)数据

图形处理器绿色通用计算关键技术/王海峰著.
—济南:山东人民出版社,2015.1
ISBN 978-7-209-08734-6

Ⅰ.①图… Ⅱ.①王… Ⅲ.①图形软件 Ⅳ.①
TP391.41

中国版本图书馆 CIP 数据核字(2015)第 021210 号

责任编辑:马 洁

图形处理器绿色通用计算关键技术
王海峰 著

山东出版传媒股份有限公司
山东人民出版社出版发行
社 址:济南市经九路胜利大街 39 号 邮 编:250001
网 址:http://www.sd-book.com.cn
发行部:(0531)82098027 82098028
新华书店经销
山东省东营市新华印刷厂印装

规 格 16 开(169mm×239mm)
印 张 10.75
字 数 180 千字
版 次 2015 年 1 月第 1 版
印 次 2015 年 1 月第 1 次
ISBN 978-7-209-08734-6
定 价 32.00 元

如有质量问题,请与印刷单位调换。电话:(0546)6441693

目　录

导　言

一、本书的写作目的

随着大数据时代的来临，在各种计算机研究领域都需要进行大规模数据实时处理，如社会计算中的数据实时分析、网络安全中的异常流与内容检测、图像处理中的海量视频分析等。由于图形处理器 GPU 通用计算的大力发展，并且 GPU 非常适合处理数据密集型的计算任务，因此 GPU 和 GPU 集群已经成为大规模数据实时处理的重要解决方案。比如，在各类数据中心或者超级计算机系统中，都存在大量图形处理器来参与通用计算。然而，在实时处理大规模数据时拥有大量图形处理器的并行计算系统中，能量成为一种需要关注的计算资源，对计算可靠性和系统扩展性起约束作用。因此有效的能耗管理和优化成为 GPU 通用计算中亟需解决的问题，也是绿色节能计算的要求。

本书关注 GPU 通用计算的绿色节能主题，集中围绕能耗管理及优化这一中心目标，对各种 GPU 和 GPU 集群通用计算的关键技术展开讨论。对于一个科学问题的探讨要经历摸索规律、掌握事物的发展规律后进行预测，最后对其进行科学的控制和管理。基于这个思考问题的线索，首先从 GPU 计算能耗测量开始，从不同的角度研究计算功耗的预测，为能耗优化和控制打下基础；再以单节点 GPU 的能耗优化为研究目标，推广到多 GPU 系统和 GPU 集群的实时能耗控制；然后考虑能耗优化对计算可靠性的影响，提出一个综合计算性能、能耗优化和可靠性的控制模型及其实现，层次分明地研究 GPU 通用计算中的能耗分析和优化问题。其中，GPU 功耗测量和预测是能耗管理与优化的基础，能耗优化和实时控制是整个研究的重点，在能耗优化的过程中保证计算性能和可靠性的损失最小是研究的难点。

本书的研究工作从理论角度有以下创新之处：

（1）将程序切片法用在计算功耗的预测方面，提高了预测精度。另外，运用小波神经网络的方法来解决预测模型对各种 GPU 体系的适应能力。

（2）利用极大熵函数法组合计算性能、能耗优化和计算可靠性三种指标，以模型预测控制策略为核心构建能耗实时控制系统。

从工程应用角度而言，本书的研究内容可转化为以下应用成果：

（1）在编译器层次建立一个功耗预测模型，增强现存编译程序。提供基于编译角度的功耗预测及功耗预判，为程序员提供一种优化代码的方法。

（2）在应用程序源代码设计完成后，在集成开发环境中以插件形式对源代码进行功耗的分析及预测，能够确定能耗瓶颈的代码区域，减少程序员对功耗硬件测量仪器的依赖及编译器的帮助。

（3）两种重要的线程调度模型可应用到各种领域的数据密集型并行算法向 GPU 移植工作中去，具有较好的普适性。

（4）一个能够适应实际生产环境的 GPU 集群能耗控制模型，全面考虑计算性能的损失、能耗优化对计算可靠性的副作用。

二、本书的组织

第一章分析图形处理器通用计算中的关键技术，以 GPU 体系结构的发展趋势为线索，详细讨论通用计算编程模型、存储模型、通信模型、负载均衡等重要方面的研究内容、方法、工具，为 GPU 及 GPU 集群能耗优化和可靠性研究奠定基础。另外，介绍 CPU 及 CPU 集群的能耗分析、优化的研究成果及发展现状，使读者可以了解哪些研究方法和手段可以在 GPU 能耗优化控制中得到应用。最后讨论现存的 GPU 及 GPU 能耗分析、管理和控制的研究现状。

第二章以建立一个 GPU 通用计算程序功耗预测模型为目标，在不借助硬件检测设备的情况下可以预测计算功耗，为普通程序员了解所开发程序的功耗情况及瓶颈提供帮助。一般 GPU 通用计算用高级语言编写，最常见的是 C 语言。该方法以编译器产生的中间语言 PTX 为预测分析的对象，先

将高级语言程序通过编译产生 PTX 代码，在此基础上对指令进行分类、统计并用 PTX 指令数来预测计算功耗。为了更准确地获得动态指令数，需要对 PTX 指令进行深入分解，比如对简单循环进行展开等。该方法简单可行、准确性较好。

第三章针对上一章功耗预测模型的局限性，提出一种全新的功耗预测模型。从源代码角度进行分析，该方法允许程序员在不依赖编译器的情况下对计算功耗进行预测。以程序切片法来分解源程序，并提取关于计算功耗的特征，再用非线性回归和小波神经网络两种方法建立预测模型。为进一步改善预测精度，对应用程序进行更细致的划分，分别为分支稀疏和分支稠密的应用程序建立预测模型。该方法不仅提高对各种 GPU 体系结构的适应能力，而且进一步降低功耗预测对硬件和底层编译器的依赖性。

第四章以数据密集型计算为应用背景，详细讨论 GPU 单节点的并行处理策略、模型以及相应算法，所提出的并行处理线程模型可应用到各种实际算法的 GPU 移植中。重点分析了两种所提并行处理策略及模型的功耗复杂性问题，以及各自适应的应用场景。以复杂网络中聚类算法为例来验证研究内容的有效性。在考虑节能的情况下，能耗优化会造成计算可靠性的下降。为此突出讨论了单 GPU 节点计算过程中故障检测和容错机制，分析了容错代价和有效性，在保证计算可靠性的前提下优化能耗。

第五章提出一种针对 GPU 集群能耗优化控制系统，该控制系统以模型预测为核心控制策略，能够适应动态计算负载的变化，实时调整 GPU 能耗状态来消减计算过程中的冗余能耗。该章内容是控制工程与高性能计算的交叉研究领域，具有一定的工程参考价值。然后以网络安全领域和视频检测领域中的实际数据来验证能耗控制系统的有效性，并评估其重要控制参数。构建网络诱骗系统获得实际的网络入侵数据，并以此作为实际工程数据对能耗控制系统进行仿真验证。

在第五章中并未考虑降低节点能耗状态对计算可靠性的影响，然而能耗优化过程中节点计算可靠性会产生损失。为了进一步提高能耗控制模型的实际应用价值，第六章讨论改良的能耗优化控制模型，以最大熵函数产生组合计算性能、可靠性及能耗的综合控制变量，加入到模型预测控制中

来改进能耗优化控制系统，考虑计算可靠性的控制模型改善了能耗状态调整机制对计算可靠性造成的损失。该方法突破了传统的多目标转化为单目标方法的局限性，能够正确辨识候选解的优劣，动态调整 GPU 集群的工作状态，使其达到计算性能最优、稳定性最好、能耗最低的目标。

研究 GPU 通用计算的读者，可先读第一章，再根据需要有选择地阅读。对有软件工程和编译原理研究背景的读者，选择读第二、三章的内容来了解在程序层面上优化计算功耗的一些方法、策略和实现思路。对于数据密集型计算的工程师和科技工作者可选读第四章，因为在大量的工程技术应用领域和科学计算领域中都存在数据密集型的持续计算任务，相信该章内容对并行算法的设计和向 GPU 移植的工作都有一定的指导意义，并且提供了计算能耗复杂性的分析方法。有控制工程研究背景的读者和在数据中心工作的科研人员可选读第五、六章内容，了解在大数据计算应用中控制异构集群能耗的方法和优化控制技术。

另外，读者需要区分功耗和能耗两个名词的不同。功耗是单位时间的能耗，一般可用功率表示，单位是瓦（W）。能耗是 GPU 运行一段时间的能量消耗，单位是焦（J）。本书第二、三两章主要讨论 GPU 应用程序的功耗预测模型，而第四、五、六章主要分析 GPU 能耗及能耗优化控制。在一些应用场景和上下文讨论中，功耗是衡量 GPU 能耗的一种方法。

第一章 绪 论

1.1 引言

计算机图形处理器 GPU(Graphics Processing Unit)具有极高计算性能和相对廉价的成本,以超过摩尔定律的速度更新硬件,2003 年后图形处理器在通用计算领域 GPGPU(General-Purpose Computation on Graphics Processing Units)取得长足发展。在学术界出现了典型处理器芯片,如斯坦福大学的 Imagine、Merrimac 及国防科技大学的飞腾 FT64[1],而飞腾处理器成功应用于"天河一号"超级计算机的设计中。在工业界 Nvidia、AMD 等公司持续更新其 GPU 硬件产品,利用推广软件的方式来扩大各自硬件产品的生态圈,Nvidia 在通用计算领域中成果显著。GPU 在通用计算的发展过程中为了提高处理器性能,晶体管的集成度越来越高,出现功耗快速增长的趋势。功耗的迅速增长会进一步提高芯片的制冷成本和高温条件下任务失效率,导致整个计算系统的可靠性下降。另一个方面,单个 GPU 存储和计算性能的瓶颈,使得多 GPU 系统和 GPU 集群的应用逐渐发展起来。对于大规模的集群系统来说,过高的功耗更是造成了巨大的能量消耗。由于 GPU 通用计算程序运行时的能耗在急剧增加,而过高的功耗给计算机或集群系统的性能、可靠性诸多方面提出了挑战。这一问题也给 GPU 通用计算提出更高要求,需要对 GPU 体系、编程模型、通信、存储管理等进行深入分析,研究 GPU 通用计算的绿色节能问题,重点关心能耗优化。在解决这一问题之前,先从下面几个问题的回答着手。

(1)为什么要优化 GPU 计算功耗?

从图形处理器设计来看,随着晶体管技术的发展,芯片内连线宽度不断地下降,每个门电路宽度小到几十纳米,使得处理器片内的计算单元密度不断提高,已经达到几十亿个晶体管的规模。随着片内集成度的急速提高,功耗也在急剧增长,因此降低图形处理器功耗成为一项重要的研究课题。另一方面,对于单个 GPU 和大规

模 GPU 集群系统来说，过高的功耗给系统的可靠性、散热等方面带来了严峻的考验，并造成了巨大的能源消耗。在当前提倡绿色计算的高性能领域中，能耗优化成为备受关注的热点问题。

(2)在什么应用场景中优化 GPU 功耗?

GPU 适合处理数据密集的计算任务，这是由 GPU 同构众核体系的特点决定的。在处理数据密集型任务时 GPU 并发启动大量轻型线程，以海量并发线程来掩盖访存延迟并获得较高的性能。大数据中有一种重要的应用场景，即实时海量数据流的处理，比如网络安全领域中的数据包检测、异常 P2P 网络流检测，网络传输质量管理中的视频流或音频流质量损失检测，社会化网络日志分析和复杂网络等科学研究领域中的实时数据流计算等。在此只关注这种应用场景下的单节点 GPU 和 GPU 集群的计算能耗优化问题，不涉及 GPU 芯片设计中的硬件优化和散热设计相关的能耗优化等方面。

(3)如何优化 GPU 计算功耗?

这正是我们努力回答的问题。从 GPU 的芯片体系、编程模型、存储管理、通信及负载均衡等通用计算关键性技术入手，先建立能够评测及预估计算任务功耗的模型，然后以功耗预测模型分析通用计算程序的功耗行为及能耗分布，分别对单节点 GPU 计算能耗和 GPU 集群能耗提出了优化解决方案。对功耗优化引起的可靠性问题进行了研究，设计故障检测及容错方案。

1.2 GPU 通用计算关键问题

GPU 作为一种通用处理器，讨论其能耗必须从其在通用计算中的各种基础问题入手[3]。首先是从硬件的体系结构了解 GPU 的发展和演化历程，体系结构中存储子系统对计算功耗的影响重大，同时也是计算性能优化和可靠性保证的基础。在体系结构的基础上编程范式及具体编程模型也是 GPU 通用计算的一个关键技术，而且结合具体的编程模型及算法可研究功耗优化问题，因为任何计算任务都需要用特定的编程模型来实现，所以从算法角度研究 GPU 通用计算功耗优化是一个重要的方向。本书将一个单节点 GPU 的计算扩展到多 GPU 系统和 GPU 集群系统，因此各节点之间通信问题与计算任务的负载均衡问题也需要突出考虑，特别是在通信过程中的能耗优化，以及结合能耗优化的负载均衡机制的研究，都将成为未来绿色计算

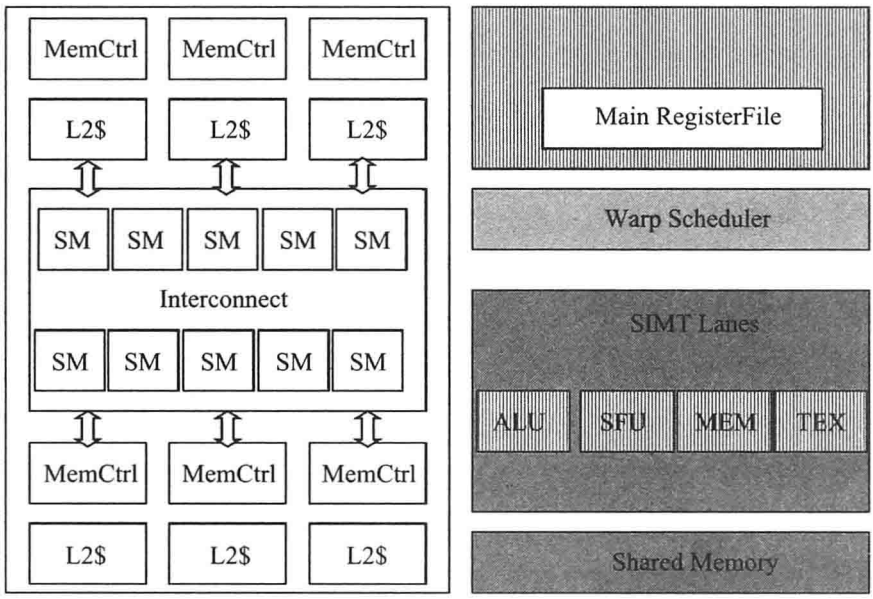
领域中的研究热点。

1.2.1 GPU 体系及发展

从芯片体系的演化能看出 GPU 向通用计算的发展趋势。在早期固定图形流水线模式下,随着图形计算需求复杂性的提高,处于图形流水线中的顶点处理器、几何处理器、像素与子素处理器等可编程性得到增强[2],不仅使处理器摆脱固定流水线的禁锢成为可能,而且逐步表现出通用计算的能力。为解决 GPU 片内负载均衡问题,统一渲染处理器(Shader Processor)取代各种可编程部件,这就出现了通用计算 GPU 的雏形[5]。而流处理器在图形处理器中的应用奠定了 GPU 通用计算的基础。流处理器是在流计算模型上充分考虑并发和通信的计算体系。对计算单元精简动态控制设计,如去除分支预测、乱序控制逻辑、内存预取等功能,通过降低硬件复杂性达到提高运算单元密度的目的。为维护数据局部性和并发性,流处理器有多层存储体系,一般分片上本地存储器、流寄存器文件和片外存储器三部分。流处理器的控制部分即流控制器,负责指令发射、流数据装载和与主机通信;外部存储控制器负责片外存储器与外部 DRAM 的数据交换。各种流处理器的主要差别在于算术计算单元的类型和数量、ALU 集群数量以及各级存储器的容量与带宽。

Imagine[6,7]是斯坦福大学的研究项目,并且在 Intel 公司和德州仪器公司的支持下实现第一款流处理器芯片。Imagine 中有 8 个 ALU Clusters,而每个计算簇中计算单元不对等,有 6 个浮点计算单元,其中有 3 个加法器、2 个乘法器。为解决片外通信昂贵、芯片上通信面积大于计算面积的问题,Imagine 通过三级存储结构减少片外通信,充分提高片内运算能力。Merrimac 项目中基于流处理器构建超级计算来提高科学计算性能,在 Merrimac 中通过网络连接 16 个流处理器,每个流处理器有 16 个 ALU Clusters,计算簇内设计对等计算单元,有利于负载调度,适合计算密集型的科学计算任务[8]。国防科技大学张春元教授领导的 MASA 团队设计了可编程 64 位流处理器 MASA[9,10],MASA 继承流体系的基本特征和层次带宽等特性,改进存储模型和计算功能单元。采用共享存储空间和非阻塞传输技术来解决中间数据导致的标量核与访存带宽压力提高的问题;计算单元中增加特定操作,并保证对等性,进一步提高核运算的性能。国防科技大学杨学军院士[1]领导开发的飞腾 64 位流处理器,在流体系结构基础上设计针对科学计算的加速方案,提供消息传递和流通信两种通信方式,并且利用自主研发的网络接口可将大量飞腾处理器连接起来构成超级计算

机。超级计算机“天河一号”中应用大量飞腾处理器,标志着国产流处理器逐渐成熟并走向工业应用道路。



(a)芯片体系　　(b)SM 结构

图 1-1　现代 GPU 体系结构

Nvidia 的主要市场对手是 AMD。以 Radeon 系列产品为例介绍 AMD 芯片。AMD GPU 属于矢量处理器[12],五个流核心(Stream Core)组成一个超长指令体系的线程处理器,其中四个普通流核心负责矢量并行操作,一个特殊流核心(T-Stream)处理特定数学计算。多个线程处理器组成一个单指令多数据处理器阵列及计算单元(Compute Unit)。存储系统由全局显存、一级和二级缓存、共享存储器和寄存器构成,每个计算单元有一级缓存和共享存储器,多个计算单元共享一个二级缓存[13]。AMD 与 Nvidia 的明显区别是流处理器体系不同:AMD 以计算单元规模来提高性能,采用单指令多数据结构,趋向指令并行方向;Nvidia 则在有限计算单元前提下优化并行体系来提高性能,采用多指令多数据,偏向线程并行度方向,因此 Nvidia GPU 更高效地执行分支程序,通用计算能力较强。其次,Nvidia 缓存及管理机制比较强大,如图 1-1(a)所示。最小线程执行粒度不同,执行线程粒度越小并行度越高,掩盖延迟的能力越强。Nvidia 执行线程粒度是 16 个线程,而 AMD 是 64 个线程。未来 Nvidia 继续向通用计算发展,而 AMD 则用自身技术优势关注片内异构核融合设计。AMD 在 2011 年 7 月上市的代号为 Fusion 的 APU,在解决 CPU 与 GPU 片间通信瓶

颈、处理标量和矢量负载都表现不凡。但是，现在 APU 中 x86 与 SIMD 核心之间的存储区域未能实现无缝集成，成为下一代 Fusion 研究重点[13]。

Intel 公司也尝试进入图形处理器领域，提供代号 Larrabee 的高端 GPU 体系设计方案，然而最终并未投片生产。Larrabee 中标量和矢量计算单元都有寄存器，标量单元是顺序执行、双发射 CPU 核心，支持预取、任务与指令调度和缓存一致性策略等功能；矢量处理器单元负责处理各种从简单到矢量计算的指令；Larrabee 具有完整一、二级缓存及管理机制，因为基于 x86 结构的设计，所以更具有通用计算能力[14]。虽然 Larrabee 未能成功以 GPU 形式出现，但是 Intel 宣布已经集成到“高性能协处理器家族”中，SC2011 大会后发布 Knight Corner 50 核的原型芯片。

总之，通过 GPU 芯片体系发展可以看出以下趋势：增加计算资源密度，提高存储体系性能和功能，增强通信能力和可靠性，降低功耗等。

1.2.2 GPU 编程模型

从片内与片间两个角度分析编程模型的发展。对于早期 GPU 片内，在数据并行编程模型基础上，程序员将各种算法映射到绘图流水线中，以此挖掘 GPU 硬件并行性。随着可编程部件功能增强，出现以顶点、子素处理器等硬件为基础的渲染模型（Shader Model）。为解决片内负载均衡问题，统一各种可编程部件后出现统一渲染模型（United Shader Model）[5]。CTM[15]对 GPU 片内编程模式有重要影响，抽象化 GPU 硬件来降低对底层的依赖，把开发控制策略返还编程人员。此后，由于更适合通用计算的流处理器 SP（Stream Processor）[2]的出现，流式编程模型（Stream Programming Model）逐渐成熟，并在 Brook 和 CUDA 编程语言中应用[16,17]。流编程模型的优点是捕捉应用程序的两种局部性：核内局部性，在核的执行过程中所有引用的数据都集中在核内；生产消费关系局部性，由于数据在各个核之间流动，有效组织核之间的逻辑顺序能把生产消费关系控制在局部范围内。

GPU 长期以协处理或加速器的方式存在，因此 GPU 与 CPU 之间的耦合关系对片间编程模型有影响，如 Nvidia GPU 与 CPU 之间是松耦合关系，而 Larrabee 与 Fusion 是紧耦合关系。松耦合体系侧重向用户提供灵活的编程模型，如 OpenCL[18]存在数据并行模型、任务并行模型和混合编程模型，根据不同应用程序的性能特点灵活选择适当编程模型；CUDA 也针对片间松耦合体系，基本编程模型是数据并行模型[19]，而且是细粒度数据并行。此外，CUDA 提供流的机制，用户运用多流可以达到

任务并行执行,因此在某种程度上 CUDA 给出一种任务并行编程的途径。松耦合体系存在一个问题,由于 CPU 即是计算管理者又是执行者,因此 CPU 未被充分利用。而片间的紧耦合体系有效解决该问题,平衡编程模型能在 GPU 与 CPU 之间合理分配计算负载,充分利用 CPU 计算资源;实现以线程为调度单位的细粒度数据和任务并行计算。

随着 GPU 片间通信能力增强,片间编程模型必然向分布式方向演化。例如 CUDASA中的分布式编程模式有效解决负载均衡和全局存储管理、通信的问题,降低 GPU 集群应用程序的开发难度[20]。

比编程模型更具体的就是 GPU 开发语言。从专门的着色语言,面向流计算模型的流语言,到现在的面向通用计算领域的编程语言,以下分三个方面介绍。GPU 编程语言起源于着色语言(Shading Language),如 GLSL,HLSL,Cg[5]。着色语言是独立图形处理硬件的高级编程语言,为了开发者灵活、方便控制并行图形渲染。它存在两个局限性:对底层图形库依赖性强,可移植性差;开发难度大,不仅要掌握并行开发技术,而且要了解硬件结构和图形库细节。针对流编程模型出现一些流编程语言,如 StreamIt[21],StreamC/KernelC[22],Brook/BrookTran[23]。流编程语言为 GPU 通用编程语言的发展奠定基础。随着非图形计算应用的增多,各种领域的用户对通用编程语言的需求增大。两大 GPU 厂商分别推出不同通用计算语言,Nvidia 的 CUDA 和 AMD 的 Brook + 。CUDA 以 C 语言为基础,由于解决以下两个问题,从而迅速增加用户数量:封装图形编程底层转化算法;降低编程难度,用户无需考虑计算模型和操作资源的限制,利于集中设计特定应用领域算法。AMD 的 Brook + 由从斯坦福大学的流编程语言 Brook 基础上扩展而来,提供流机制、各种数据对象和核并行计算方式,在 AMD 各产品系列中得到应用。学术界也有较好研究成果,清华大学侯启明[24]提出 BSGP 通用编程语言,将 Bulk Synchronous Programming (BSP)模型应用到流处理器编程中,与 CUDA 获得相似计算性能。

编程语言在向抽象化发展。因为 CUDA 与 OpenCL 用户仍然需要管理存储器、建立和调度核运算,所以逐渐有研究者关注提高抽象设计能力的编程语言,如 PyCUDA[25],JCUDA[26],hiCUDA[27]等,将现有并行编程语言与灵活的脚本语言相结合,创建更抽象的编程语言。Glift[28]提供灵活、高效的数据结构实现机制,通过添加库函数、元语言等方法开发软件,属于一种抽象编程语言。

1.2.3 存储模型与分析

现代 GPU 存储子系统由不同存储区域构成，主要分为片外和片内存储区两部分，如图 1-2 所示[12]。片外存储区由通用存储器与常量存储器构成，通用存储器由早期纹理存储器演化而来，如 Nvidia 的全局显存；片内存储区包括缓存与用于局部数据共享和重用的快速存储器。片内存储器具有局部特性，在体系中属于一个多线程计算单元，即 Nvidia 流多处理器或 AMD SIMD 引擎。GPU 存储子系统有以下特点：GPU 不具备主动发起访问片外存储器的能力，与 CPU 存在较高的通信和访存耦合度；GPU 片上缓存是不满足数据一致性的二维纹理 Cache，而且容量小。因为 GPU 存储子系统性能决定计算性能，所以厂商一直致力于 GPU 存储性能的提高。增加片外存储控制器寻址范围，Nvidia 的 Fermi 可访问 6GB 全局显存；增强片内缓存硬件功能，如 Fermi 的二级缓存体系和可灵活配置的一级缓存机制。本节主要对 GPU 存储模型分两方面讨论，即 GPU 片上存储管理与分布式存储管理。

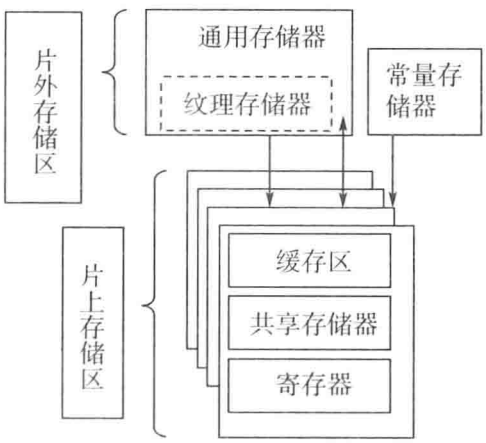


图 1-2 GPU 存储子系统

存储性能一直是 GPU 计算性能的瓶颈，定量分析是一种常用的研究方法。因为厂商未公开片内存储器硬件机制，所以需要合理的黑盒模型来分析存储性能，如 3C 模型分析片内缓存行为[29]。在建立量化分析模型时需要考虑应用程序的访存特性，根据不同访存特征对应用程序进行分类，针对不同类别来分析性能损失。由于应用程序访存模式的多样性，因此提高定量分析模型的普适性成为一个难点。建模过程中需要综合考虑，不能忽视任何一个区域对计算性能的影响，如片内共享存储