第46卷 第2期 西 安 交 通 大 学 学 报 Vol. 46 No. 2
2012年2月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Feb. 2012

CPU-GPU 系统中基于剖分的全局性能优化方法

张保，董小社，白秀秀，曹海军，刘超，梅一多
（西安交通大学电子与信息工程学院，710049，西安）

摘要：针对将应用移植到 CPU-GPU 异构并行系统上时优化策略各自分散、没有一个全局的指导思想的问题，提出了一种基于剖分的全局性能优化方法. 该方法由优化策略库、剖分工具库和策略配置模块组成. 优化策略库将应用移植到异构并行系统上的性能优化过程划分为访存级、内核加速级和数据划分级 3 级优化；针对 3 级优化剖分工具库提供了 3 级剖分机制，通过运行时的剖分技术获取剖分信息；策略配置模块根据所获取的信息指导用户在每级优化中选择合适的优化策略. 实验证明，基于剖分的全局性能优化方法可以明确地指导将应用移植到 CPU-GPU 异构并行系统上的全局优化过程，利用该优化方法后，以矩阵相乘和傅里叶变换为例的应用性能提升明显，最终性能相对于访存级优化最高可提高 30%左右.

关键词：CPU-GPU 异构并行系统；全局优化；3 级优化；3 级剖分
中图分类号：TP399 **文献标志码**：A **文章编号**：0253-987X(2012)02-0017-07

Profiling Based Optimization Method for CPU-GPU Heterogeneous Parallel Processing System

ZHANG Bao, DONG Xiaoshe, BAI Xiuxiu, CAO Haijun, LIU Chao, MEI Yiduo
(School of Electronics and Information Engineering, Xi′an Jiaotong University, Xi′an 710049, China)

Abstract: A profiling based optimization method for CPU-GPU heterogeneous parallel processing system is proposed to address the problem that the present optimization strategies get sectional thus failed to guide a global optimization. It is composed of the optimization strategy library, the profiling tool library, and the strategy deploy module, and the optimization strategy library divides the performance promotion process into a three-level optimization, including the memory-access level, the kernel-speedup level, and the data-partition level. The profiling tool library realizes three-level profiling mechanisms towards three-level optimizations to obtain application information, and the strategy deploy module guides users to choose an adaptive strategy with the information obtained by profiling tool library. Experimental results show that the proposed one is able to guide the optimization process of applications transplanted to heterogeneous parallel system. The performance for matrix multiplication and fast Fourier transform are improved obviously, and the final performance is heightened by 30% compared with the memory-level optimization.

Keywords: CPU-GPU heterogeneous parallel system; global optimization; third-level optimization; third-level profiling

近年来，随着图形处理器(GPU)的计算能力日 趋增强，其主要用途由图形渲染过渡到通用计算[1]

收稿日期：2011-08-12. 作者简介：张保(1987—)，男，硕士生；董小社(通信作者)，男，教授，博士生导师. 基金项目：国家高技术研究发展计划资助项目(2009AA01A135，2009AA01Z108)；中央高校基本科研业务费专项资金资助项目(08142007).
网络出版时间：2011-10-18 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20111018.1150.005.html

方面,从而使得 GPU 的含义逐渐演变为通用处理器. GPU 属于众核架构,拥有数以百计的计算单元,NVIDIA 公司最新开发的基于 Fermi 架构的 Tesla C2050 的单精度浮点处理能力已超过 1TFlops (Tera Floating Point Operation Per Second),同时其双精度浮点处理能力也得到了加强,最高可达到 600GFlops.

GPU 可以提供强大的计算能力,但由于其缺少分支预测单元,不善于处理复杂的逻辑事务,因此,"主核心+协处理器"模式的 CPU-GPU 异构并行系统应运而生,CPU 负责复杂的逻辑事务处理,GPU 则用于密集型数据的计算. 这种 CPU 作为主核心、GPU 作为协处理器的理念使得该系统在计算能力方面拥有着巨大潜力. 目前,世界超级计算机 top500 中排名第一的"天河一号 A"即采用由 CPU 和 GPU 组成的异构混合系统.

目前,国内外相关人员基于 CPU-GPU 异构并行系统已经做了大量的研究. 例如:为应用合理分配存储空间,以减少访存延迟;开发数据并行度,以充分利用 GPU 的计算单元;将应用数据分流处理,使用计算时间以更多地隐藏主机端与设备端的通信开销等. 文献[2]针对将应用计算核心移植到 GPU 上进行加速时如何充分利用 GPU 大量的计算单元进行了研究,文献[3]对用户的优化策略进行了精简,以减少用户优化应用时的大量冗余工作. 为消除指令分支以增加数据并行度,文献[4]主张采用线程-数据重映射的方法,同时研究了 CPU-GPU 管道机制和 LAM 算法以隐藏映射开销. 为更加有效地利用 GPU 计算资源,文献[5]采用将数据划分后分批交给 GPU 进行处理的方法,以重叠 PCI-E 总线通信与 GPU 计算,同时提出了以比例方式划分数据从而实现两者最大程度重叠的相关算法.

文献[6]应用文献[2-3]中的优化策略实现了一个 GPGPU 源到源编译器,以实现内存优化和并行管理,为用户实现高效代码提供了便利. 为提升应用性能,文献[7-8]针对应用使用资源进行分析以预测程序执行时间,获取性能瓶颈,进一步指导用户的优化方向.

这些研究工作为 CPU-GPU 应用性能的提升作出了巨大贡献,然而由于优化理论较为零散,用户在针对应用进行优化时往往顾此失彼,不能充分发挥 CPU-GPU 系统的计算潜力;同时由于缺乏一个指导应用优化过程的全局的指导方法,用户只能通过多次试探以找到最优的效果,从而造成了大量的冗余工作. 因此,目前在如何编程以合理使用 CPU 和 GPU 资源从而更好地提升应用性能方面,仍面临着巨大的挑战.

针对该问题,本文提出了一种可以指导全局优化的基于剖分的优化方法,选择矩阵相乘和快速傅里叶变换对提出的优化方法的可行性和有效性进行了验证,其最终性能相对于访存级优化最高可提高 30%左右,比内核加速级优化提高了 10%左右.

1 系统架构概述

1.1 GPU 架构

NVIDIA GT200 系列 GPU 架构如图 1 所示. 其片内拥有 10 个线程处理器群(TPC),每个 TPC 内拥有 3 个多处理器(SM),一个 SM 由 8 个流处理器(SP)、两个特殊功能计算单元(SFU)和一个双精度计算单元(DPU)组成,单精度浮点计算能力为 933G. GPU 片内每个 SM 拥有 16 384 个 32 b 寄存器文件、16 kB 的共享存储器,以及纹理和常量的只读 Cache 等片上存储;片外是由全局存储器、纹理存储器和常量存储器共用的显存空间,访存延迟较高.

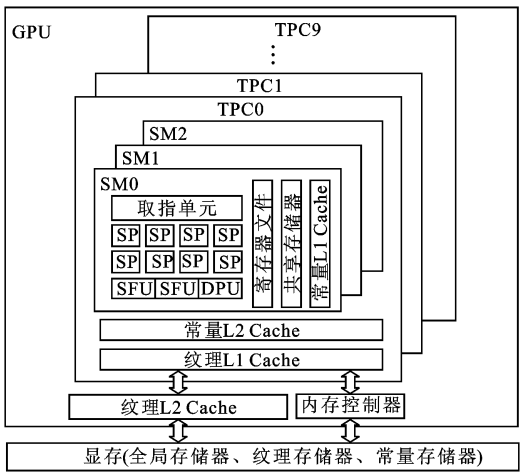

图 1 GT200 GPU 架构(Tesla C1060)

1.2 CPU-GPU 异构并行系统

CPU-GPU 异构并行系统如图 2 所示. CPU 通过主机端内存控制器访问主机内存中的数据,GPU 通过设备端存储器、控制器访问设备显存中的数据,主机和设备端之间采用 PCI-E 总线进行连接. 使用 CPU-GPU 异构并行系统处理数据需经过以下步骤:①CPU 将待处理的初始数据从主机内存传输到设备显存;②GPU 对设备显存中的数据进行加速处理;③数据处理完毕后,CPU 控制将数据处理结果由设备显存传输回主机内存.

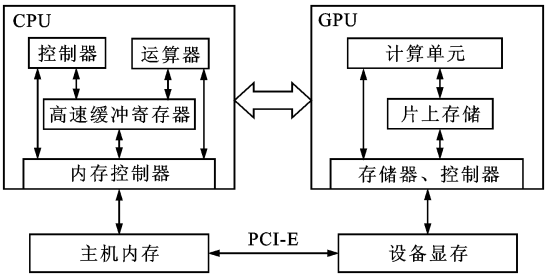

图 2　CPU-GPU 异构并行系统

CPU-GPU 异构混合系统使用 GPU 作为协处理器加速应用计算核心，通过充分发挥 GPU 的计算能力可以很好地提升应用性能. 然而，GPU 复杂的架构成为应用优化的难点，同时该异构系统“主核心＋协处理器”的特征增加了应用数据额外的通信开销. 因此，合理编程以充分开发 GPU 的计算能力，同时最大程度上减少主机和设备端的通信开销，需要一套完整的剖分优化机制. 通过该剖分优化机制，一方面合理组织应用的优化过程以提升其整体性能；另一方面指导用户选择优化技术以明确地规划应用的优化方向.

2　基于剖分的全局优化方法

本文提出的优化方法如图 3 所示. 该方法由优化策略库、剖分工具库和策略配置模块组成，以剖分优化的方式指导用户提升 CPU-GPU 异构并行系统上的应用性能. 本文方法选用 CUDA[9] 作为底层支持库，在优化策略库将应用性能优化的过程划分为访存级、内核加速级、数据划分级 3 级优化，在每级优化中针对应用特性提供了不同的优化策略. 剖分工具库中封装了若干函数，对应 3 级优化策略实现了 3 级剖分机制，用户通过选择其内置的函数在程序中进行插桩以获取剖分信息. 策略配置模块接收剖分信息在每级优化时进行策略配置，显式地提供给用户，以指导全局优化过程.

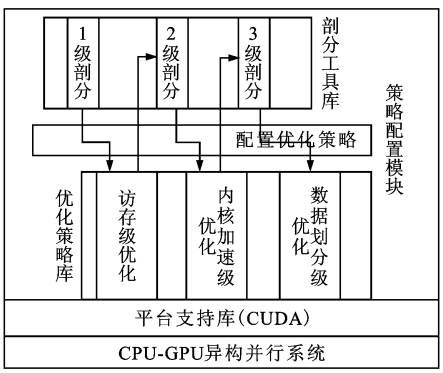

图 3　基于剖分的优化方法

2.1　优化策略库

优化策略库将应用移植到 CPU-GPU 异构并行系统上的性能优化过程划分为访存级、内核加速级、数据划分级 3 级优化. 该库在每级优化中针对应用的不同特性提供了相应的优化策略，以渐近的方式完成应用的全局优化过程，其整体框架如图 4 所示. 根据获取的剖分信息为应用变量在每级优化中选择相应的优化策略.

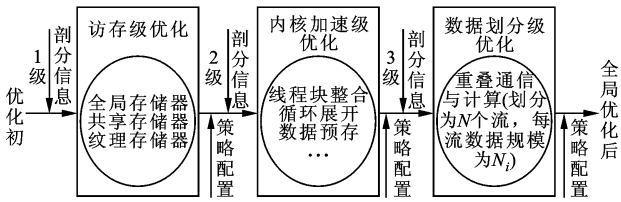

图 4　优化策略库的整体框架

2.1.1　*访存级优化*　GPU 片外是由全局存储器、纹理存储器及常量存储器共用的显存空间，其中纹理存储器和常量存储器存储方式为只读. 片上存储包括寄存器、共享存储器和纹理及常量存储器的高速缓存，其中共享内存有着几乎与寄存器一样快的访存速度. 常量存储器空间太小，而寄存器由每个线程私有，多数用于存储临时变量，不能用于数据共享. 因此，本文所提供的访存级优化策略仅包括全局存储器、共享存储器以及纹理存储器.

上述 3 种存储空间的特性及其使用的优化方法见文献[10]，本文的优化方法要求在访存级优化，根据剖分获取的应用变量的访存信息合理地分配存储空间. 该级优化最主要思想在于，尽量将需要多次访问且访存规则的变量转移至共享内存以减少访存延迟，为访存不规则的只读变量分配纹理存储器以提高访问效率.

2.1.2　*内核加速级优化*　在访存级优化的基础上，针对使用共享内存的变量，选择内核加速级提供的线程块整合优化策略，以可充分利用 GPU 计算资源为底线（为充分利用 SM 上的计算资源，一个 SM 上至少存在 2 个活动线程块或是 6 个活动的 warp，而一个 SM 上线程块的数量又受限于寄存器、共享内存使用量以及一个 SM 可最多承载的线程块及线程数量等多方面因素），在 2 个或多个线程块所访问的数据可以共享时，将其整合为一个线程块，以减少访问显存的次数及访存延迟，提升应用性能.

在应用线程块整合后，针对应用特性，可以继续选择采用其余的优化策略. 例如，展开循环可减少寄存器资源使用量，同时消除地址空间计算指令；数据

预取即将下一次线程需要计算的数据提前转移至寄存器中,以保证计算单元 SP 在更长时间内处于计算状态.

2.1.3 *数据划分级优化* GPU 处理的数据需要经过 PCI-E 总线在主机内存和设备显存之间进行通信.在数据量可以充分利用 PCI-E 总线以及 GPU 计算资源的前提下,为重叠 PCI-E 总线通信与 GPU 计算,需要将初始数据划分后分批交给 GPU 进行处理.

数据划分级优化主要思想在于最大程度地重叠通信与计算的时间,进而减少应用的整体执行时间.在 PCI-E 总线的通信时间大于 GPU 的计算时间时,划分数据以使得通信时间尽可能多地隐藏计算时间;反之,划分数据以使得计算时间尽可能多地隐藏通信时间.理论上,划分的数据块越多,可以隐藏的时间也越多,但由于使用流的开销以及数据块较小时不能充分利用通信或计算资源所致,盲目地划分数据很有可能造成应用的整体性能下降.为避免该情况,本文在数据划分级优化中使用剖分工具库剖分信息,结合文献[6]提到的算法来确定划分流的数目,以及每个流所占有的数据规模.

2.2 剖分工具库

剖分工具库针对优化策略库的 3 级优化实现了 3 级剖分机制,其内封装了若干函数,通过选择相应的函数对程序进行插桩以获取相应的剖分信息,将该信息提供给策略配置模块以配置对应的优化策略.各级剖分的信息及相应的剖分工具如表 1 所示.

1 级剖分工具针对源串行程序的计算核心进行插桩,获取变量访存信息以指导应用变量在设备端分配存储空间,得到访存级优化内核.使用 CUDA 提供的 nvcc 编译器附加——cubin 选项编译访存级优化内核以获取 cubin 文件.2 级剖分工具扫描该文件,获取线程使用资源信息以确定每个 SM 上可承载的线程块数,采用线程块整合等多种策略加速计算,使得加速级优化.3 级剖分工具通过对主程序插入时间桩,获取数据拷入、计算及拷出的时间,提供给策略配置模块配置划分流的数目以及每个流所占有的数据规模.

表 1 剖分工具库

3级剖分	剖分信息	剖分工具
1级剖分	v_io:变量输入输出特性	V_IO_PRO:判定 v_io
	v_size:变量元素个数	V_SIZE_PRO:判定 v_size
	v_type:变量元素类型	V_TYPE_PRO:判定 v_type
	v_count:变量数据块访存次数	V_COUNT_PR:判定 v_count
	v_step:变量数据块访存步长	V_STEP_PRO:判定 v_step
2级剖分	device_version:设备版本号 use_shared:各线程块共享内存使用量 use_register:各线程寄存器使用量	V_KERNEL_PRO:扫描 cubin 文件
3级剖分	copyin time:拷入时间 kernel time:计算时间 copyout time:拷出时间	FUN_TIME_START:记录开始时间 FUN_TIME_START;记录结束时间

2.3 策略配置模块

本文使用策略配置模块在分析应用剖分信息后实行优化策略配置,在每级以参数的形式显式地提供给用户,以指导其针对移植到 CPU-GPU 异构并行系统上的应用的全局优化过程.根据剖分信息针对各级优化的策略配置方式如图 5 所示.

```
if(v_step_con== 'y' )
 {
   if(v_step==1)
   {
      if(v_count/v_size<=1)
         d_strategy= 'global memory' ;
       else
         d_strategy=global memory' ;
   }
   else
      d_strategy= 'global memory' ;
}
else
{
   if(v_io==0)
      d_strategy= 'texture memory' ;
    else
      d_strategy= 'global memory' ;
}
```

(a)访存级优化策略配置

```
if(data_shared==1)
{
   int group=1;
   int bnum=16*1024/use_shared;
   int tnum=bsize*bnum;
   if(bnum>8 ‖ tnum>1024 ‖
          tnum*use_register>16384)
      bnum--;
   while(bnum>2)
   {
      group++;
      use_shared*=group;
      bnum=16*1024/use_shared;
      tnum=b size*bnum
   }
   if(bnum<2 ‖ bnum/32<6)
      group--;
}
```

(b)内核加速级优化策略配置

```
if(is_stream==1)
{
   if(copyyin_time+copyout+time>
              kernel_time)
   {
     //采用数据平均划分的策略
     N=2;
     N1=Mtotal/2;
     N2=Mtotal/2;
   }
   else
   {
     //采用数据按比例划分的策略
     float a=kernel_time/copyin_time;
     fioat b=kernel_time/copyout_time;
     …
   }
}
```

(c)数据划分级优化策略配置

图 5 策略配置模块

图 5a 描述的是访存级优化策略配置方法,主要根据变量的访存次数及访存步长是否规则等信息,从全局存储器、共享存储器、纹理存储器中为其选择合适的存储空间予以分配. 其中,v_step_cons 代表该变量的访存步长是否恒定;变量的输入输出特性 v_io 有 3 种方式,0 表示输入,1 表示输出,2 表示变量原位变换;d_strategy 标志代表为变量选取的访存策略.

内核加速级优化策略配置方法如图 5b 所示. 本文的剖分工具库仅实现了内核加速级优化中线程块整合策略的配置,以保证每个 SM 上有足够的活动线程块或足够的 warp 为底线. 针对使用共享内存的变量,在 2 个或多个线程块访问显存的数据块可以共享时,将其整合为 1 个线程块进行处理. 其中,data_shared 为 1 时,表示相邻线程块访问的数据可以共享,用 bsize 代表每个线程块的线程数量,首先确定当前一个 SM 上可承载的线程块数量 bnum,用 group 代表将要整合的线程块数量. 在进行线程块整合后,用户针对应用的特性选择内核加速级优化中其余优化策略以加速内核计算.

图 5c 描述的是数据划分级优化策略配置方法. 首先通过 1 级剖分所得到的信息 v_size 和 v_type 计算变量数据块的大小,以判定其是否需要进行数据划分. 当 is_stream 为 1 时,表示需要划分数据;为 0 时,表示不需要划分数据. 在可以进行数据划分时,比较 3 级剖分所得到的数据拷贝时间和计算时间,在拷贝时间大于计算时间时,将数据平均划分为 2 个流;反之,按文献[4]中式(5)～式(8)的流程计算划分流的数目以及各流的数据规模.

3　验证与测试

3.1　实验环境

本文实验基于浪潮英信 NF 5588 服务器,该服务器配置了 Intel Xeon 4 核 CPU(E5520 1.6 GHz),3×4 GB DDR3 内存,同时配置了 NVIDIA Tesla C1060 GPU,主机与 GPU 设备之间通过 PCI-E 2.0×16 总线进行数据传输. 该服务器安装了 Red Hat Enterprise Linux AS 5.3 操作系统,内置 CUDA Tookit V2.3.

本文实验选用经典算法矩阵相乘和快速傅里叶变换作为测试用例,对提出的 3 级优化方法的可行性进行了验证. 使用本文方法与已有工作访存级优化、内核加速级优化、CUBLAS、CUFFT 库进行了性能比较,其中 CUBLAS 和 CUFFT 是 NVIDIA 公司提供的专门针对矩阵运算和快速傅里叶变换的函数库.

3.2　实验测试

3.2.1　矩阵相乘　首先选择矩阵相乘 $\boldsymbol{C}=\boldsymbol{A}\times\boldsymbol{B}$ 为测试用例,其中矩阵 $\boldsymbol{A}$、$\boldsymbol{B}$ 和 $\boldsymbol{C}$ 均为方阵,选取矩阵维度为 512、1 024、1 536 和 2 048. 根据本文提出的剖分工具库信息,为每级优化配置的优化策略如表 2 所示,用户根据配置的策略完成 3 级优化. 本实验中,数据划分参数的设定参考了文献[5].

表 2　矩阵相乘优化策略配置

矩阵维度	d_stragety	group	is_stream
512	“shared”	6	0
1 024	“shared”	6	1
1 536	“shared”	6	1
2 048	“shared”	6	1

针对矩阵相乘,分别在运用访存级优化、内核加速级优化、本文 3 级优化(包括了数据划分级优化)、CUBLAS 间进行了性能对比测试,结果如图 6 所示,各优化相对于未优化(使用全局存储器实现矩阵相乘,未使用任何优化策略)时的加速比如图 7 所示. 实验结果表明,矩阵相乘的性能有了明显的提升,例如当矩阵维度为 1 024 时,本文提出的优化策略比访存级优化提高了 33%左右,比内核加速级优化提高了 14%,比 CUBLAS 提高了 16%. 同时,本文 3 级优化后矩阵相乘的性能相对于未优化时可提高 12 倍左右.

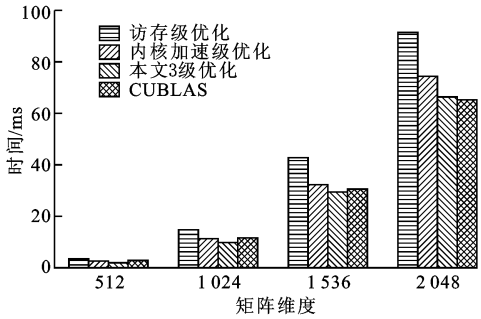

图 6　矩阵相乘测试

在矩阵维度小于等于 1 536 时,本文提出的部分工具库性能高于 CUBLAS 库所实现的矩阵相乘的性能;当矩阵维度增大到 2 048 时,CUBLAS 库表现出了更好的性能. 这是由于本文目的在于实现一个通用的全局优化策略,而 CUBLAS 是专门针对矩阵运算设计的函数库,在访存级优化中 CUBLAS

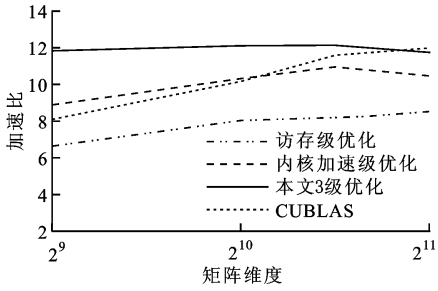

图7 矩阵相乘时各方法加速比的比较

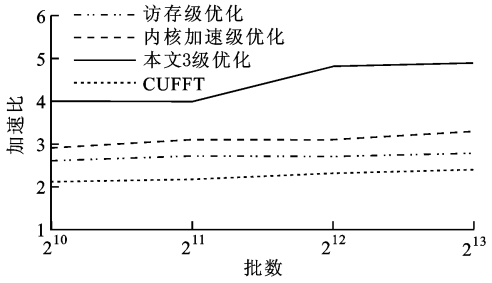

图9 快速傅里叶变换时各方法加速比的比较

使用了寄存器等优化策略.

3.2.2 傅里叶变换 以512点快速傅里叶变换为测试用例,选择批数为1 024、2 048、4 096和8 192对本文方法进行测试验证,优化策略配置如表3所示.对访存级优化、内核加速级优化、本文3级优化(包括了数据划分级优化)、CUBLAS进行了性能对比,结果如图8所示,各方法相对未优化(使用全局存储器存储数据,不使用优化策略)时的加速比如图9所示.实验结果表明,快速傅里叶变换在逐次应用3级优化时性能层次提升明显,3级优化后的性能相对于访存级优化性能提高了30%～43%,同时比NVIDIA提供的CUFFT库的性能提高了40%以上,相对于未优化时的性能可达到4～5倍.

表3 快速傅里叶变换优化策略配置

批数	d_stragety	group	is_stream
1 024	“shared”	2	1(平均划分为2个流)
2 048	“shared”	2	1(平均划分为2个流)
4 096	“shared”	2	1(平均划分为2个流)
8 192	“shared”	2	1(平均划分为2个流)

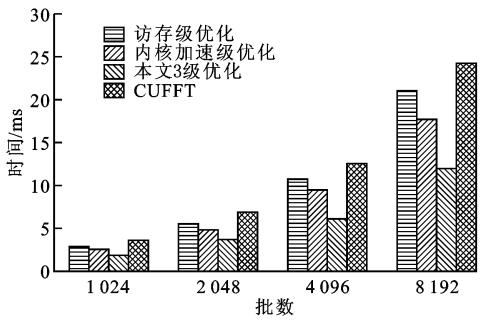

图8 快速傅里叶变换测试

3.2.3 剖分优化开销评估 使用剖分方法和工具对源程序进行剖分优化将不可避免地带来一定的开销,因此本文选择矩阵相乘为测试用例对剖分优化开销做进一步评估分析.不同维度的矩阵相乘执行时间如表4所示.根据实验结果,在每种情况下,加剖分后的执行时间比未剖分的执行时间大约增加了20%,然而根据剖分结果优化后的性能比未优化时提高了约10倍以上.因此,剖分优化虽然有开销,但对应用性能的提高非常明显.从另一角度看,针对某个应用所做的剖分,可以把剖分信息保留下来为以后的计算提供优化依据.从以上两个方面来看,本文提出的基于剖分的全局性能优化方法具有一定的价值和意义.

表4 矩阵相乘剖分优化开销

维度	时间/s			
	未剖分	剖分	未优化	优化
512	1.23	1.56	0.022	0.002
1 024	10.98	14.2	0.116	0.010
1 536	39.46	50.56	0.355	0.029
2 048	181.89	207.2	0.782	0.066

4 结 论

本文提出了一种基于剖分的面向CPU-GPU异构并行系统的优化方法,分为优化策略库、剖分工具库以及策略配置模块3部分.优化策略库将应用移植到该异构并行系统上的优化划分为访存级、内核加速级、数据划分级3级优化.剖分工具库针对3级优化实现了3级剖分,通过选择函数进行插桩以获取剖分信息,并将其传递给策略配置模块以显式地指导每级优化时的优化策略,以渐近的方式完成应用的全局优化过程.

参考文献:

[1] 吴恩华. 图形处理器用于通用计算的技术、现状及其挑战[J]. 软件学报,2004,15(10):1493-1504.
WU Enhua. State of the art and future challenge on general purpose computation by graphics processing

unit[J]. Journal of Software, 2004, 15(10): 1493-1504.

[2] RYOO S, RODRIGUES C I, BAGHSORKHI S S, et al. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA[C]// Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. New York, USA: ACM, 2008:73-82.

[3] RYOO S, RODRIGUES C I, STONE S S, et al. Program optimization space pruning for a multithreaded GPU[C]// Proceedings of the Sixth Annual IEEE/ACM International Symposium on Code Generation and Optimization. Boston, USA: ACM, 2008: 195-204.

[4] ZHANG EDDY Z, JIANG Yunlian, GUO Ziyu, et al. Streamlining GPU applications on the fly: thread divergence elimination through runtime thread-data remapping[C]// Proceedings of the 24th ACM International Conference on Supercomputing. New York, USA: ACM, 2010: 115-126.

[5] 张保，曹海军，董小社，等. 面向图形处理器重叠通信与计算的数据划分方法[J]. 西安交通大学学报，2011，45(4)：1-6.
ZHANG Bao, CAO Haijun, DONG Xiaoshe, et al. Novel GPU data partitioning method to overlap communication and computation[J]. Journal of Xi′an Jiaotong University, 2011, 45(4): 1-6.

[6] YANG Yi, XIANG Ping, KONG Jingfei, et al. A GPGPU compiler for memory optimization and parallelism management[C]// Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation. New York, USA: ACM, 2010: 86-97.

[7] MALONY A D, BIERSDORFF S, MAYANGLAMBAM S. An experimental approach to performance measurement of heterogeneous parallel applications using CUDA[C]// Proceedings of the 24th ACM International Conference on Supercomputing. New York, USA: ACM, 2010: 127-136.

[8] BAGHSORKHI S S, DELAHAYE M, PATEL S J, et al. An adaptive performance modeling tool for GPU architectures[C]// Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. New York, USA: ACM, 2010: 105-114.

[9] NVIDIA Corporation. NVIDIA CUDA Programming guide[EB/OL]. [2010-07-15]. http://www.nvidia.com/object/cuda_home_new.html.

[10] 董小社，冯国富，王旭昊，等. 基于 Cell 多核处理器的层次化运行时支持技术[J]. 计算机研究与发展，2010，47(4)：561-570.
DONG Xiaoshe, FENG Guofu, WANG Xuhao, et al. Research on multilayer runtime library technology for Cell/B. E. processor[J]. Journal of Computer Research and Development, 2010, 47(4): 561-570.

[本刊相关文献链接]

应用回归分析的数据关联算法. 西安交通大学学报，2011，45(8)：92-96.
面向图形处理器重叠通信与计算的数据划分方法. 西安交通大学学报，2011，45(4)：1-4.
海量信息异常检测问题的异常概率排序算法. 西安交通大学学报，2011，45(4)：36-40.
一种面向多处理器系统的在线低功耗调度算法. 西安交通大学学报，2010，44(8)：15-19.
面向服务级别协议的核实与规划服务质量系统框架研究. 西安交通大学学报，2010，44(6)：1-5.
利用投影时序逻辑的多内核进程调度建模与验证. 西安交通大学学报，2010，44(3)：52-57.
面向属性约束的自动信任协商模型. 西安交通大学学报，2009，43(8)：1-5.
面向 Cell 宽带引擎架构的异构多核访存技术. 西安交通大学学报，2009，43(2)：1-5.
基于思维进化的集群作业调度方法研究. 西安交通大学学报，2008，42(6)：651-654.
支持多种 Linux 版本的动态内核性能测试技术. 西安交通大学学报，2008，42(6)：674-678.
自适应大规模服务器集群监控系统的构建. 西安交通大学学报，2008，42(4)：399-403.
网格测试引擎：一种构建网格测试环境的基楚构架. 西安交通大学学报，2007，41(8)：884-888.
按序选取的并行存储系统数据分布方式研究. 西安交通大学学报，2007，41(8)：899-902.

（编辑　杜秀杰）