面向图形处理器重叠通信与计算的数据划分方法

张保[1]，曹海军[1]，董小社[1]，李丹[1]，胡雷钧[2]
(1. 西安交通大学计算机科学与技术系，710049，西安；2. 高效能服务器和存储技术国家重点实验室，250013，济南)

摘要：针对“主核心+协处理器”式异构并行系统采用数据平均划分再分批执行的方法来解决主协式处理架构的额外通信开销时未能充分利用系统资源的问题，提出了一种新的数据比例划分方法.结合系统通信带宽和图形处理器(GPU)的计算能力，将应用数据按比例划分为大小不同的数据块后分批提交给GPU处理，使系统的传输资源PCI-E总线和计算资源GPU在一段时间内并行工作，从而实现了应用通信与计算的重叠.在处理按照比例划分的数据块过程中，尽可能充分利用系统的传输资源和计算资源，以减少数据传输和计算的相互等待时间.实验结果表明，采用数据比例划分方法后的应用性能明显提高，可以有效地重叠通信与计算时间，矩阵相乘和快速傅里叶变换总执行时间比未划分时分别减少了5%和30%左右，比平均划分时分别减少了3%和6%左右.

关键词：图形处理器；重叠通信与计算；数据划分
中图分类号：TP399 **文献标志码**：A **文章编号**：0253-987X(2011)04-0001-05

Novel GPU Data Partitioning Method to Overlap Communication and Computation

ZHANG Bao[1], CAO Haijun[1], DONG Xiaoshe[1], LI Dan[1], HU Leijun[2]
(1. Department of Computer Science and Technology, Xi′an Jiaotong University, Xi′an 710049, China;
2. State Key Laboratory of High-End Server and Storage Technology, Jinan 250013, China)

Abstract: A novel data partitioning method is proposed to address the problem that the "CPU+GPU" heterogeneous parallel processing system cannot fully utilize its resources when average-partition data blocks in batches is processed to deal with the extra overhead for communication. Application data is processed by GPU after being partitioned into blocks with different sizes in proportion by taking the communication bandwidth and the GPU computing capacity into account. Therefore, PCI-E bus and GPU can work in parallel in a period of time to overlap communication and computation. The partitioned data blocks can utilize system resources as much as possible, and hence the mutual waiting time between data transferring and computing can be reduced. Experimental results show that application performance is raised significantly by effectively overlapping communication and computation. Comparisons with no-partition and average-partition show that matrix multiplication's performance is improved by about 5% and 3%, while Fast Fourier Transform's performance is enhanced by about 30% and 6%, respectively.

Keywords: graphic processing unit; overlap communication and computation; data partition

近年来，随着多核技术的发展，图形处理器 (GPU)的可编程性和通用计算能力日趋增强[1]，从

收稿日期：2010-08-26. 作者简介：张保(1987—)，男，硕士生；董小社(联系人)，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(2009AA01Z108，2009AA01A135，2006AA01A109)；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：2011-02-21 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20110221.1605.007.html

而为CPU+GPU模式的“主核心+协处理器”式异构并行计算系统提供了契机,其中CPU和GPU负责各自所擅长的任务,从而提升应用的整体性能.

这种主协式处理架构需要额外的通信开销,一种有效的解决途径是将数据划分后分批交给GPU执行,使得系统的传输资源和计算资源可以并行工作,以实现应用通信与计算重叠.然而,用户在划分数据时大多采用基于数据平均划分的方法,忽略了应用和系统资源的特性,因此所划分的数据块往往不能充分利用系统资源,从而可能导致应用性能优化不能达到理想的效果.

针对该问题,本文提出了一种新的数据划分方法.该方法考虑了系统的通信带宽以及GPU的计算能力,将数据按比例划分为不同大小的数据块,在此基础上有效地重叠通信与计算的时间.

本文利用CUDA[2]来估测所提出的数据划分方法.CUDA是NVIDIA公司提出的面向新型GPU架构的编程接口,使用它所提供的流处理技术可以实现总线通信与GPU计算的同时执行.

1 相关工作

基于Cell宽带引擎架构,文献[3]通过减少访存开销以及改善软件的可编程性,提出了一种基于CBEA的高效的异构多核访存技术.针对方体计算,文献[4]提出了基于GPU的并行方体算法GPU-Cubing,算法采用自底向上、广度优先的划分策略,每次并行完成一个cuboid的计算并输出,多个分区同步处理、分区内多线程并行,使得算法获得了良好的加速比.

在应用数据通信与计算重叠方面,文献[5-6]针对网格系统中的工作负载机制展开研究,利用优化策略更大化重叠通信与计算时间,同时减少额外的开销.在基于MPI实现的消息传递程序中,重叠通信与计算的研究围绕两个方面展开[7-8]:一是更有效地重叠通信与计算以提升应用性能;二是简化优化工作的流程,制定一些程序转化规范.文献[9]针对众核处理器Godson-T平台上的快速傅里叶变换应用展开研究,提出了优化算法,通过设置计算与通信重叠等优化策略,达到了节省L2 Cache存储开销,提升快速傅里叶变换性能的目的.

2 CPU和GPU异构并行系统

由CPU和GPU组成的异构并行系统如图1所示,CPU访问主机内存中的数据,GPU访问设备显存中的数据,主机和设备间通过PCI-E总线进行数据传输.

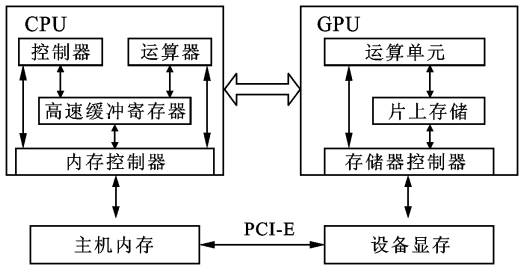


图1 CPU和GPU异构并行系统

在异构并行系统中,处理步骤如下: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.

在一次任务处理过程中,该异构并行系统需要经历多次数据通信.假如将待处理的所有数据从主机内存一次性传输到设备显存之后再启动GPU计算,那么在传输过程中GPU处于空闲状态,而在GPU对数据进行计算时,PCI-E总线又处于空闲状态.由此可见,PCI-E总线和GPU都没有得到充分利用,造成了一定的资源浪费.

解决以上问题的办法是将通信与计算重叠,即将待处理的初始数据分批交给GPU执行.在这个过程中,GPU在对前一个数据块进行处理时,PCI-E总线则正在传输下一个数据块,也处于工作状态.这样,在一段时间内,PCI-E总线与GPU可同时处于工作状态,从而重叠了数据通信与计算的时间,最终减少了应用的整体运行时间.

3 数据划分方法

通过以上分析,为了更有效地使用通信与计算重叠的方法来缩短应用的整体运行时间,本文提出了一种面向GPU的数据划分方法:首先将待处理的初始数据按一定比例进行划分,然后将所划分的数据块分批交给GPU处理,以实现应用通信与计算的重叠,同时按照比例划分数据块,可以充分利用系统的传输资源和计算资源.

在将初始数据划分以重叠通信与计算时,若采用不同的划分方式,所能重叠的通信与计算的时间也不尽相同.另外,在数据分批执行的过程中,数据块的大小不同也可能会产生不同的开销.若数据块太小,则传输频繁,而每次传输需进行设备初始化,

这将造成总的数据传输延迟较长.另外,小数据块的计算不能充分利用系统的计算资源而产生额外的开销.与之相反,若数据块太大,则可能造成计算和传输的相互等待时间过长,不能充分地重叠计算与通信而产生浪费.总而言之,通过科学的数据划分方法来实现通信与计算最大可能的重叠至关重要.

3.1　通信与计算分析

为了实现科学的数据划分,本文将数据划分后所产生的额外开销归结为以下两个方面.

设将用 GPU 计算的数据量已确定,T_{copy}表示将所有数据从主存传输到 GPU 显存需要花费的总时间,若将数据划分为 n 块,其中第 i 块数据从主存传输到 GPU 显存需要花费的时间为 T^i_{copy},则

$$T_{\text{fw}} = \sum_{i=1}^{n} T^i_{\text{copy}} - T_{\text{copy}} \tag{1}$$

式中:T_{fw}表示数据划分后将其按块传输比整体传输需要额外花费的时间.

相应地,设将用 GPU 计算的数据量已确定,T_{kernel}表示所有数据在 GPU 上计算所需要花费的总时间,若将数据划分为 n 块,其中第 i 块数据在 GPU 上计算需要花费的时间为 T^i_{kernel},则

$$T_{\text{cw}} = \sum_{i=1}^{n} T^i_{\text{kernel}} - T_{\text{kernel}} \tag{2}$$

式中:T_{cw}表示数据划分后将其在 GPU 上按块计算比整体计算需要额外花费的时间.

为使应用的整体性能得到有效提升,将数据划分处理时要尽量避免或减少以上两方面的开销.为更具体地考虑上述两方面因素,需要对所使用的异构并行系统以及待优化的应用进行测试,这包括以下 3 个方面.

第 1,理论上,数据块的传输时间与数据量的大小是正相关的.如图 2 所示,当实际系统中(以本文测试平台 GPU 总线接口 PCI-E 2.0 X16 为例)数据块较小时,由于设备初始化和传输延迟起主导因素,因此传输时间随数据量增加而缓慢增长,当数据块较大时,传输时间则随着数据量的增加呈现近似线性增长.本文定义传输时间由缓慢增长变为线性增长的转折点为主机与设备通信的带宽饱和点.那么,只有当数据量大于带宽饱和点所对应的数据量 m_{f} 时才能够充分发挥通信设备的传输效率.

在实际系统中确定 m_{f} 的方法为:从小数据量开始,逐渐增大数据量,记录每次的数据传输时间,设第 i 次传输的数据量为 d_i,数据传输时间为 t_i,则数据量与数据传输时间的比值为 $k_i = d_i/t_i$,第 $i+1$ 次

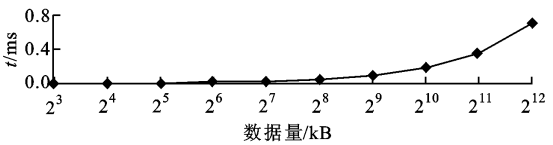


图 2　数据块传输时间与数据量大小的关系

传输的数据量与数据传输时间的比值为 k_{i+1},若 $k_i = k_{i+1}$,则该系统的带宽饱和点所对应的 $m_{\text{f}} = d_i$.

第 2,针对待优化的应用,分析其运行特性,得到将该应用移植到 GPU 上进行计算时使 GPU 内所有的计算单元正好可以满负荷运行一次时的数据量,记为 m_{c}.

第 3,针对待优化的应用,分析其运行特性,得到将该应用划分以实现通信与计算的重叠时,由于其数据相关性造成的可以划分的最小数据块所对应的数据量,记为 m_{l}.

如图 3 所示,若不进行数据划分,即应用在 CPU 和 GPU 组成的异构并行系统上一次完整的执行需要经历 3 个阶段:①数据拷入,将待处理的初始数据从主机内存传输到设备显存,拷入时间记为 T_{i};②计算处理,GPU 的计算单元对显存中的数据进行处理,计算时间记为 T_{c};③数据拷出,将数据处理结果由设备显存传输回主机内存,拷出时间记为 T_{o}.需要说明的是,本文提出的数据划分方法主要针对计算时间大于通信时间的应用.

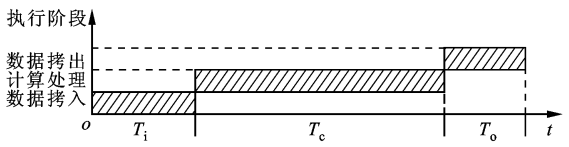


图 3　数据未划分的时空图

3.2　数据划分方法

由图 3 可知,针对某个特定的应用以及实际系统,以上 3 个阶段的运行时间是固定的.然而,根据流水线原理,通过分批执行应用的数据来实现通信与计算重叠,可以使得总的运行时间减少.

如图 4 所示,本文提出的数据划分方法的主要思想是:将应用数据划分为 n 个数据块,其中第 i 个数据块的数据量最大,其两侧数据块的数据量则逐渐减小(确定 n 与 i 的方法见 3.3 节).同时,前 $i-1$ 个数据块中每个数据块的计算时间正好可以隐藏其后一个数据块的拷入时间,而后 $n-i$ 个数据块中每个数据块的计算时间正好可以隐藏其前一个数据块的拷出时间.在所有数据块中,第 1 个数据块和第 n 个数据块的数据量相对较小,因此为了使得所有数

据块能够充分利用传输带宽和 GPU 内的计算单元,需要使得第 1 个数据块和第 n 个数据块的数据量接近于 m_f、m_c 和 m_l 中的大者,在这种情况下,几乎所有数据块的传输和计算都能够充分利用传输带宽和 GPU 内的计算单元,从而保证了式(1)、(2)的额外开销尽可能地小,避免了由于额外开销过大而对应用的整体性能造成影响.

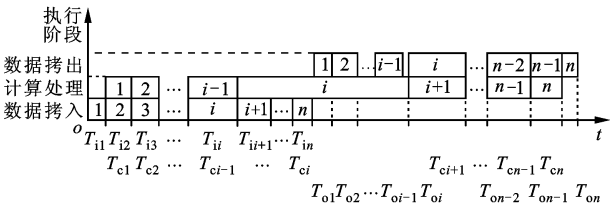


图 4 数据划分为 n 个数据块的时空图

根据以上提出的数据划分思想,为了达到 n 个数据块中第 i 个数据块的数据量最大、两侧数据块的数据量逐渐减小的效果,本文设定各数据块之间的大小呈比例关系,即

$$1:a:a^2:\cdots:a^{i-1}:(a^{i-1}/b):(a^{i-1}/b^2):\cdots:(a^{i-1}/b^{n-i})$$
$$\text{where } 1<i<n \tag{3}$$
$$a=T_c/T_i;\ b=T_c/T_o \tag{4}$$

3.3 参数确定

根据本文提出的数据划分思想可知,数据划分最终结果的理想情况是:第 1 个数据块和第 n 个数据块的数据量接近于 m_f、m_c 和 m_l 中的大者. 假设用 M_1 表示第 1 个数据块的数据量,M_n 表示第 n 个数据块的数据量,则有

$$M_1\approx M_n\approx\max\{m_f,\ m_c,\ m_l\} \tag{5}$$

为估测 n 和 i 的值,可以将 M_1 和 M_n 视作相等,并统一用 M 表示,应用中待处理数据的总数据量记为 M_{total}. 由式(3)中前 $i-1$ 个数据块的大小比例可知,前 $i-1$ 个数据块的数据量分别为 $M, aM, \cdots, a^{i-2}M$,则可以推出,第 i 个数据块的数据量为 $a^{i-1}M$;由式(3)中后 $n-i$ 个数据块的大小比例可知,后 $n-i$ 个数据块的数据量分别为 $b^{n-i-1}M,\ b^{n-i-2}M, \cdots, M$,则可以推出,第 i 个数据块的数据量为 $b^{n-i}M$,所以有

$$a^{i-1}=b^{n-i} \tag{6}$$

同时,所有数据块的数据量之和即为待处理数据的总数据量 M_{total},所以有

$$\left(\sum_{p=1}^{i-1}a^{p-1}\right)+(a^{i-1})+\left(\sum_{q=i+1}^{n}b^{n-q}\right)=\frac{M_{total}}{M} \tag{7}$$

因此,由式(6)、(7)可以估测出 n 和 i 的值,再结合总数据量 M_{total},按照式(3)将数据划分为 n 个数据块,各个数据块的大小如下式所示

$$M_k=\begin{cases}\dfrac{a^{k-1}}{\left(\sum\limits_{p=1}^{i}a^{p-1}\right)+\left(\sum\limits_{q=i+1}^{n}a^{i-1}/b^{q-i}\right)}\times M_{total} \\ 1\leqslant k\leqslant i \\ \dfrac{a^{i-1}/b^{k-i}}{\left(\sum\limits_{p=1}^{i}a^{p-1}\right)+\left(\sum\limits_{q=i+1}^{n}a^{i-1}/b^{q-i}\right)}\times M_{total} \\ i+1\leqslant k\leqslant n\end{cases} \tag{8}$$

4 验证与测试

4.1 实验环境

本文实验环境基于浪潮英信 NF 5588 服务器,其配置了 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 X16 总线进行通信;该系列 GPU 设备支持数据传输与计算同时执行.

本文服务器安装了 Red Hat Enterprise Linux AS 5.3 操作系统,内置 CUDA Toolkit V2.3. 实验通过 CUDA API 函数将应用数据使用的主机端内存分配为页锁定内存,在此基础上使用 CUDA 提供的流处理技术、拷贝函数和内核函数启动的异步特性来实现主机-设备端通信与计算重叠.

CUDA 编程指南基于数据平均划分的方法,已经被多数编程人员采用. 同时,为体现将数据划分后执行所能达到的性能优势,实验中对数据未划分、平均划分以及本文的数据划分方法进行了对比测试.

4.2 矩阵相乘

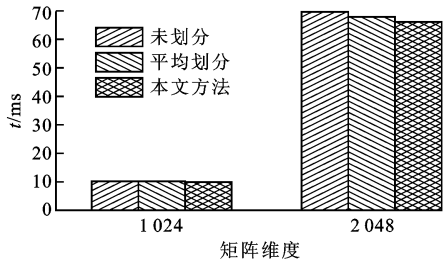
本文的矩阵相乘测试用例为 $\boldsymbol{C}=\boldsymbol{A}\times\boldsymbol{B}$,其中 $\boldsymbol{A}$、$\boldsymbol{B}$ 和 $\boldsymbol{C}$ 均为 1 024、2 048、4 096 和 8 192 的方阵. 测试中数据平均划分时的数据块数等于采用本文方法划分时的数据块数,不同之处在于,前者的数据块之间大小相同,而后者数据块之间的大小遵循式(8).

通过对所搭建的由 CPU 和 GPU 组成的异构并行系统以及所要实现的矩阵相乘算法进行分析,可以方便简单地测试获得矩阵相乘在不同维度时的参数,如表 1 所示. 由式(6)、(7)推出 n 和 i 的值后,即可根据式(8)计算出需要划分的各个数据块的大小. 所划分的数据块可以实现通信与计算更有效地重叠,此过程实现简单,说明本文提出的数据划分方法是可行的.

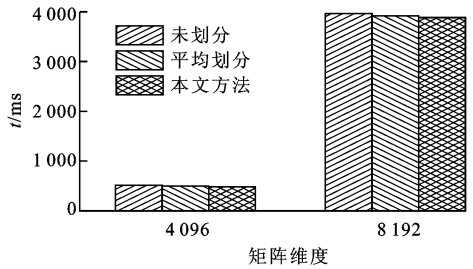
表 1　矩阵相乘算法参数

短阵维度	m_f	m_c	m_l	a	b	n	i
1 024	512	360	64	12.3	12.1	3	2
2 048	512	360	128	22.9	22.3	3	2
4 096	512	360	256	43.7	43.0	3	2
8 192	512	360	512	87.1	85.4	3	2

如图 5 所示,实验结果表明,在每种情况下,采用本文提出的数据划分方法后矩阵相乘的性能均有明显提高,其总执行时间比未划分时减少了 5%左右. 这是因为数据划分能够实现通信与计算的重叠,避免了通信与计算相互等待而造成的时间开销. 另外,由于本文提出的数据划分方法考虑了有效利用通信带宽和设备的处理能力,能更有效地重叠通信与计算的时间,因此其性能相对于平均划分时提高了 3%左右. 由此可见,本文提出的数据划分方法可以有效地提升应用的整体性能.



(a)1 024×1 024 和 2 048×2 048 方阵



(b)4 096×4 096 和 8 192×8 192 方阵

图 5　采用不同划分方法的矩阵相乘运行时间

4.3　快速傅里叶变换

本文以 2^{18} 点快速傅里叶变换(FFT)为测试用例,对不同批数(即 15、30、45 和 60)的情形进行了对比测试. 快速傅里叶变换测试与矩阵相乘测试获得参数的过程相同,不同批数时的参数如表 2 所示,测试结果如图 6 所示.

表 2　快速傅里叶变换参数

批数	m_f	m_c	m_l	a	b	n	i
15	512	360	2 048	3.7	3.6	5	3
30	512	360	2 048	3.8	3.7	5	3
45	512	360	2 048	3.7	3.6	5	3
60	512	360	2 048	3.7	3.6	7	4

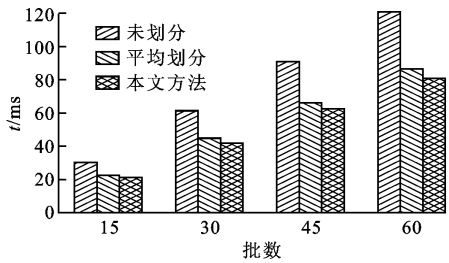


图 6　采用不同划分方法的 FFT 运行时间

实验结果表明,在每种情况下,采用本文提出的数据划分后 FFT 的性能均有明显提高,其总执行时间比未划分时减少了 30%左右;另外,采用本文提出的数据划分方法相对于平均划分时其性能提高了 6%以上.

5　结　论

本文提出了一种面向 GPU 重叠通信与计算的数据划分方法,结合所使用系统的通信带宽以及计算能力,将待优化的应用中需要处理的数据按照一定比例划分后分批交给 GPU 执行,从而有效地重叠了通信与计算时间,进而提升了应用的整体性能. 以矩阵相乘和快速傅里叶变换为例进行数据划分和性能测试,实验结果表明,本文提出的数据划分方法可使应用的整体性能得到有效提升,且数据划分过程简单可行.

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

[3]　冯国富,董小社,丁彦飞,等. 面向 Cell 宽带引擎架构的异构多核访存技术[J]. 西安交通大学学报,2009,43(2):1-5.

FENG Guofu, DONG Xiaoshe, DING Yanfei, et al. A memory access technology of heterogeneous multi-core system based on cell broadband engine architecture[J]. Journal of Xi′an Jiaotong University, 2009, 43(2): 1-5.

[4] 周国亮,陈红,李翠平,等. 基于图形处理器的并行方体计算[J]. 计算机学报,2010,33(10):1788-1808.
ZHOU Guoliang, CHEN Hong, LI Cuiping, et al. Parallel data cube computation on graphic processing units[J]. Chinese Journal of Computers, 2010, 33(10):1788-1808.

[5] YANG Yang, RAART K V, CASANOVA H. Multi-round algorithms for scheduling divisible loads[J]. IEEE Transactions on Parallel and Distributed Systems, 2005,16(11):1092-1102.

[6] TAO Yongcai, JIN Hai, WU Song, et al. Adaptive multi-round scheduling strategy for divisible workloads in grid environments[C]// Proceedings of the 23rd International Conference on Information Networking. New York, USA:ACM, 2009:260-264.

[7] SHET G A, SADAYAPPAN P, BERNHOLDT E D, et al. A framework for characterizing overlap of communication and computation in parallel applications [J]. Cluster Computing, 2008,11(1): 75-90.

[8] ANTHONY D, LORI P, MARTIN S. MPI-aware compiler optimizations for improving communication-computation overlap[C]// Proceedings of the 23th International Conference on Supercomputing. New York,USA: ACM, 2009: 316-325.

[9] 周永彬,张军超,张帅,等. 基于软硬件的协同支持在众核上对1-DFFT算法的优化研究[J]. 计算机学报,2008,31(11):2005-2014.
ZHOU Yongbin, ZHANG Junchao, ZHANG Shuai, et al. Software/hardware co-design for 1-D FFT optimization on many-core architecture[J]. Chinese Journal of Computers, 2008,31(11):2005-2014.

(编辑 武红江)