

一种基于 GPU 通用计算的容错方法

徐丹妮, 贺占庄

(西安微电子技术研究所, 陕西 西安 710054)

摘 要: 为确保 GPU 通用计算(GPGPU)程序在 CPU-GPU 异构平台上运行的可靠性,设计了一种以软件方法实现的容错模型.在分析 GPGPU 程序运行过程中瞬时故障的产生模式以及错误的传播路径后,对 GPGPU 程序运行所依赖的 CPU 端和 GPU 端分别进行容错设计,并针对 GPGPU 程序的运行特点,设计能够降低容错运算开销同时提升系统协同工作能力的优化方案,从而在提高 GPGPU 程序的可靠性的同时降低容错设计所带来的额外开销.通过对典型实例的测试验证了所提出的方案的可行性以及性能.

关键词: GPGPU; CUDA; 复算; 容错优化

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2014)02-0018-05

A Fault Tolerance Method Based on GPGPU

XU Dan-ni, HE Zhan-zhuang

(Xi'an Microelectronics Technology Institute, Xi'an 710054, China)

Abstract: This paper proposes a new fault-tolerant model realized by software method to ensure the reliability of general purpose computation on graphics hardware (GPGPU) on CPU-GPU heterogeneous platform. After analyzing the transient fault occurrence mode and error propagation of GPGPU, fault-tolerant designed both in CPU side and GPU side. An optimal scheme of the fault-tolerant which can reduce the computational overhead and enhance the ability of system interoperability is raised according to the feature of GPGPU. In addition, overhead from the design of fault-tolerance will decline when improving the reliability of GPGPU program. Finally, the feasibility and performance of the model proposed is tested and verified on typical examples.

Key words: GPGPU; CUDA; re-execution; fault tolerance optimization

1 引言

特殊的并行化处理架构使 GPU 拥有普通 CPU 无法比拟的数据运算能力^[1],这也使它在通用计算领域被寄予厚望,为此我们将通用计算程序重新设计后运行到 CPU-GPU 架构上以期获得更高的运算速度,即 GPGPU. GPGPU 中偶发的错误常会对最终的计算结果造成巨大的影响,而之前由于 GPU 仅作为图形处理使用很少考虑容错设计,因而对于此类新型架构的容错设计刚刚起步.如今,随着 GPGPU 的大规模推广及使用, GPGPU 的容错设计日益重要.在此领域, sheaffer 曾提出过针对 GPG-

PU 的基于硬件的容错机制^[2]. Ameya V. Abhyankar 则将 CPU 上传统的容错设计的方案移植运用到 GPGPU 上,并且加以对比讨论^[3]. 2010 年 NVIDIA 在其 fermi 架构 GPU 中首次使用增加 ECC 校验码的方式来保护存储部件数据的可靠性^[4]. 本文在之前的研究基础上进一步讨论 GPGPU 软件容错的设计方法及优化方案.

2 相关技术研究

2.1 CUDA 编程模型

2006 年 NVIDIA 公司发布了统一架构 GPU 和统一架构计算平台(CUDA),CUDA 可将一种简

收稿日期: 2013-04-13; 修回日期: 2013-05-22

基金项目: 国产化指挥平台应用技术研究(204030202)

单类C语言程序运行到CPU-GPU异构体系上,这使GPU摆脱了需要复杂的图形API才能进行操控的困境.通用计算程序可以被方便地移植到CPU-GPU异构体系上,利用GPU强大的并行计算能力进行运算,CUDA的编程模型如图1所示.

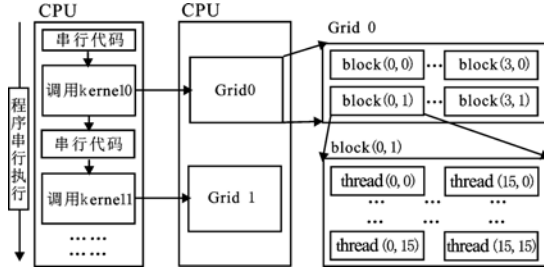


图1 CUDA编程模型^[1]

完整的CUDA程序由device端和host端两部分构成^[1].device端程序在GPU上执行,它们被称为kernel.host端程序在CPU上执行,它们负责调用各个kernel,并在前后调用间进行少量的串行计算,kernel的调用使CPU和GPU端的数据呈相互依赖关系.如图1所示,kernel在device端通常以一种分层形式组织线程.block是kernel的执行单位,在CUDA编程模型中,block以一定的方式被分发到GPU运算阵列的各个流多处理器(SM)上执行^[5].CPU和GPU二者协同工作.

2.2 容错方法的研究

可容错系统,即在系统发生逻辑故障的情况下,系统中的程序依然可以正确地执行^[6].对系统进行容错的基本策略即进行冗余设计,冗余的基本途径有以下几种:

(1)时间冗余.复算是一种典型的基于时间冗余的容错技术,简单来说即将同一段代码执行多次,通过比较结果来检错并判断是否回滚^[3],通过复算可以有效的检测出运算过程中发生的瞬时错误.这种容错方式不需要额外的硬件,软件改动也很少,使用方便,但它需要硬件有较强大的计算能力,故适合于拥有强大运算能力的GPU.

(2)空间冗余.空间冗余往往使用两套不同的硬件来执行同一段程序的两个不同版本,并比较结果检错.它有较强的执行效率,但是需要额外增加硬件设备,在具体实现时困难较大.

(3)基于算法多样性的冗余.例如,对特定的一些算法,设计与之最为匹配的检错算法.这种方式往往可以提高容错的效率,但针对特定算法进行特定设计会增加程序设计难度^[3].

3 GPGPU容错实现

GPGPU容错设计即对整个GPGPU体系进行容错设计.GPU的运算单元由于其高密度的集成方式导致其运行时高温^[2],这种状态很容易造成运算单元发生瞬时的运算故障,如何对运算单元的瞬时故障导致的错误进行容错处理是本节讨论的内容.另外,CPU运算单元虽然瞬时故障率不高,但作为GPGPU体系不可或缺的一部分,本文也会对此做相应的讨论.

3.1 故障模型

图2为模拟CUDA框架下的GPGPU程序在运行过程中瞬时故障发生及错误传播路径的示意图,其中故障点及错误数据均由下划线及粗体标出.

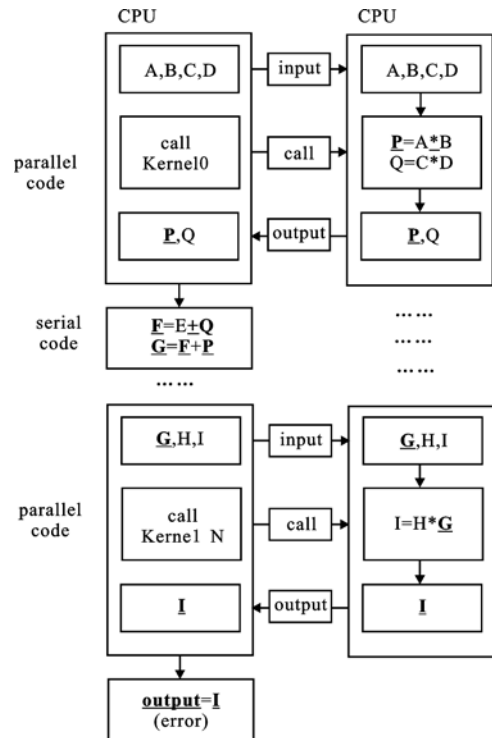


图2 GPGPU故障模型

GPGPU运算过程中瞬时故障一般可以分为两类,GPU端故障和CPU端故障^[7].如图2所示,GPU端故障,如kernel0运算时发生的瞬时故障,它导致结果P出错;CPU端故障,如CPU端的串行程序发生瞬时计算故障,导致结果F出错.

GPGPU中由瞬时故障导致的错误通常被分为GPU端原生错误,传播错误,CPU错误等^[8].此处将其归结为两类,原生错误,由于瞬时故障发生从而直接导致的运算结果错误,如模型中{P, F};传播错误,由于kernel的调用和数据的相关性、活跃性

导致错误沿数据通路在系统中不断传播扩散,继而不断引发的新错误,如模型中 $\{G, \dots, I, \text{output}\}$.

3.2 容错机制的实现

加入容错设计的目的是检测纠正可能发生的错误,使通用计算程序在有瞬时故障发生的情况下依然可以得到正确的运算结果. 根据前文中错误的分类,基于 GPGPU 的容错设计可分两类.

(1)消除原生错误的影响. 首先, GPU 端容错, 即对 kernel 容错设计, 如此可检测纠正 GPU 中瞬时故障导致的运算错误, 确保 kernel 的最终结果正确. 其次, CPU 端容错, 检测纠正 CPU 串行运算时瞬时故障导致的错误, 保证 kernel 间串行运算的结果正确.

(2)消除传播错误的影响. kernel 是 CPU 和 GPU 间数据传播的桥梁, 故也是错误传播的媒介, 上文中的措施保证了 kernel 输入输出数据的正确, 避免错误在整个程序中传播扩散, 从而可以消除传播错误.

具体设计时将基于回滚恢复的 checkpointing 技术^[9]移植应用到 GPU 通用计算的容错设计中, 适当加以改进使其适应 CUDA 编程模型和 GPU+CPU 异构体系结构, 设计模型如图 3 所示.

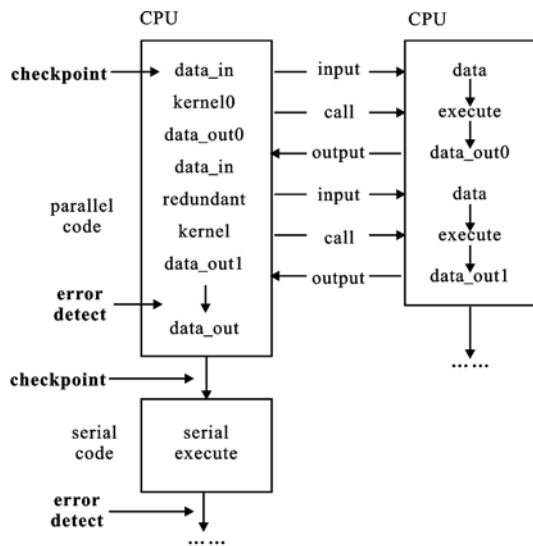


图3 GPGPU 容错模型

整个容错分为备份、冗余、检错纠错三部分.

(1)备份方式. 设计计算状态保存点 (checkpoint), 周期性保存恢复程序运算需要的全部信息^[9].

(2)冗余方式. 选择适合 GPU 通用计算的复算. 对 GPU 端进行冗余处理时, 在 CPU 端对同一个 kernel 连续调用两次, 使得 kernel 在 GPU 中连续执行两

次, 从而得到两个输出的结果. 而对于 CPU 中的串行运算, 同样采用简单的复算方式, 并同样在串行程序运行的前后设置 checkpoint 和 error_detect.

(3)检错纠错方式. 设置错误检测点 (error_detect), 在对复算结果执行检错程序进行比较, 判断结果是否正确, 如结果正确则更新 checkpoint, 有误则将程序回滚至最近的 checkpoint 并利用备份数据重新执行.

以上容错设计, 能够较为严格地检测出计算部件瞬时故障引起的错误, 并且有效阻止错误在程序中的传播和扩散.

3.3 容错机制的优化

前文中提出的容错模型, 可以提高系统的稳定性, 但同时也有一些不足之处:

(1)GPU 和 CPU 之间的数据的读写操作是需要时间开销的, 对同一个计算任务两次启动 kernel 并两次读写数据会增加额外开销.

(2)CPU 启动 kernel 后需要等待 GPU 运算并返回运算结果, 此时 CPU 处于空闲等待状态, 这浪费了 CPU 的资源.

(3)在 CPU 端进行备份, 不仅需要备份 kernel 执行需要的数据, 还需要保存 CPU 执行 kernel 调用时的相关上下文, 这些数据会增加备份的时间和空间开销, 但对于 GPU 中故障的检测和恢复并没有实际意义.

(4)CPU 的硬件特性使得它发生瞬时故障率没有 GPU 高, 且 CPU 不具备 GPU 所拥有的强大计算能力, 故采用复算冗余代价高.

因此, 可以以缩短执行时间, 提高程序执行效率为目标对此容错模型进行优化. 优化后的容错模型如图 4 所示.

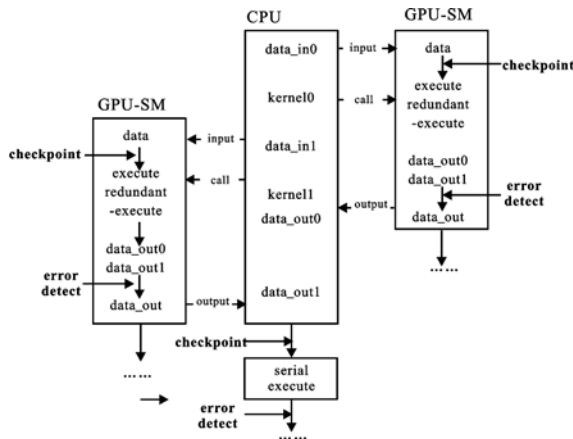


图4 优化后容错模型

如图 4 所示, 对容错模型进行改进, 其使用的策

略和达到的目标如下:

(1)降低 kernel 即并行运算容错开销.首先,重新设计 kernel,使原始运算和冗余运算能在同一个 kernel 中进行,此方式只需进行一次 kernel 的调用和一次读写数据,可以减少 CPU 与 GPU 间数据传输量和并降低通信开销.其次,在 GPU 中设置 checkpoint 和 error_detect,这样可以减少备份的数据量,如调用 kernel 时涉及的上下文等对恢复无用的数据就可以省略.

(2)降低 CPU 端串行运算容错开销.由于 CPU 端串行运算量小,针对不同的运算进行特定的容错设计所增加的时间开销可以被接受,而算法多样性的冗余设计却能够大幅度降低复算给 CPU 造成的计算压力.

(3)提高 GPU 和 CPU 协同工作能力.对原始数据进行相关性分类,使 CPU 在空闲时转而执行无数据依赖关系的新 kernel,实现计算和读写的重叠.在此基础上,利用延迟写入技术^[10],在检错程序检测到的某个 kernel 数据有误并进行恢复处理时,其余执行过程不停,只是将写新备份点的时间推迟到得到容错结束得到正确结果后.采用以上的方法,隐藏了 CPU 空闲等待时间,同时更好地利用 GPU 的并行运算能力.

综上,此优化方案,能够在保证系统可靠性的前提下提升系统整体性能.

4 实验与评价

为检测所建立的模型实际运行时的效果,在典型的矩阵运算上进行容错模型的验证,矩阵规模为 $(320,320) \times (640,320)$,运算方式为常见的乘加混合运算.

实验所用的软硬件平台为 CPU: Intel i5-2410M,主频 2.30 GHz;显卡: Nvidia GT520 MB;开发工具, VS2008+ CUDA5.0.

假设系统的平均无故障时间为 750 ms,测得在容错以及容错优化的方案中,错误都被检出并排除,故在可靠性上方案均可以达到设计要求.对比原始程序,容错方案,以及优化后方案的运算性能,结果如表 1 所示.

表 1 总开销对比 ms

原始程序	容错方案	优化方案
4 970	11 688	6 757

本文设计的容错方案以增加程序运行的时间开

销为代价,相比原始程序加入容错设计后时间开销大幅度增加,然而进行优化设计后整个程序时间开销降低了约 40%,执行的效率得到提高.

本文中设计了对容错优化的三项策略,经实验测试得这三方面对于最终优化结果的贡献如图 5 所示.

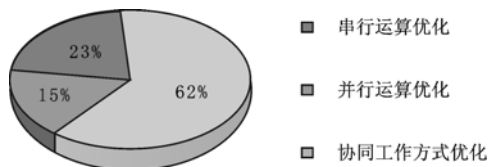


图 5 各部分优化占总优化百分比

并行运算即 kernel 的容错优化设计对最终结果贡献最大,约占 62%;其次为 CPU 串行运算容错的优化,约占 23%;最后是协同工作方式的优化,约占 15%。故对于前两部分,再次进行详细测试分析,测试并行运算及串行运算在故障和正常两种情况下的平均容错开销,结果如表 2 所示.

表 2 运算部分开销对比 ms

		原始程序	容错方案	优化方案
并行运算	正常	9	18	12
	故障	9	38	17
串行运算	正常	15	62	16
	故障	15	63	47

经优化设计后,两部分计算的容错时间开销均有显著降低,二者的优化收益比均超过 20%,优化有明显的效果.

5 结束语

为提高 GPGPU 可靠性,本文针对其运算过程中瞬时故障导致的错误提出了一种容错方案,即在不改动硬件结构的情况下进行软件容错.设计容错模型是将 CPU 端和 GPU 端均纳入容错的范畴,以保障整个程序执行过程中数据的正确性和系统的稳定性.最后通过的实验可以看到实际的容错和优化效果.

当然,在 GPGPU 的容错设计中,还有很多值得探讨和改进的,例如备份点的数据优化策略,即如何最小化备份数据量进而提高性能,这些将在今后的研究中重点关注.

参考文献:

[1] 张舒,褚艳利. GPU 高性能计算之 CUDA [M]. 北

- 京:中国水利水电出版社,2009.
- [2] Sheaffer J W, Luebke D P. A hardware redundancy and recovery mechanism for reliable scientific computation on graphics processors [C]// Timo Aila, Mark Segal. GraphicsHardware, California;Siggraph, 2007: 55-64.
- [3] Ameya V A. Performance cost analysis of software implement hardware fault tolerance technique[D]. Madison: University of Wisconsin Madison, 2010.
- [4] Nvidia C. NVIDIA's next generation CUDA compute architecture:FERMI[J]. Computer system, 2009(26): 63-72.
- [5] Kanter D. NVIDIA's GT200:Inside a parallel processor [J]. Real world technologies, 2008(1):1-12.
- [6] Avizenis A. Design of fault tolerant computers [C]//Fall joint computer conference. ACM, 1967: 733-743.
- [7] 贾佳,杨学军. 异构系统硬件故障传播行为分析及容错优化[J]. Journal of Software, 2011, 22(12): 2854-2864.
- [8] 徐新海,杨学军. 一种面向 CPU-GPU 异构系统的容错方法[J]. 软件学报, 2011, 22(10):2538-2552.
- [9] 王攀峰. 应用级 checkpointing 技术研究 with 实现[D]. 长沙:国防科学技术大学, 2008.
- [10] 马安国. GPGPU 体系结构关键技术研究[D]. 长沙:国防科学技术大学, 2011.

作者简介:



徐丹妮 女, (1989—), 硕士. 研究方向为计算机控制、GPU 架构.

贾占庄 男, (1962—), 总工程师, 博士生导师, 研究员. 研究方向为计算机控制技术、IP 核设计等.

(上接第 17 页)

- [2] Frank O' Mahony, Jaussi, Kennedy, et al. A 47 * 10 Gb/s 1.4 mW/Gb/s parallel interface in 45nm CMOS [J]. IEEE J. Solid-State Circuits, 2010, 45(12):2828-2837.
- [3] Kurd N, Mosalikanti, Neidengard, et al. Next generation Intelore micro-architecture (Ne-halem) clocking [J]. IEEE J. Solid-State Circuits, 2009, 44 (4): 1121-1129.
- [4] Beukema, Sorna, Selander, et al. A 6. 4-Gb/s CMOS SerDes core with feed-forward and decision-feedback equalization[J]. IEEE J. Solid-State Circuits, 2005, 40 (12):2633- 2645.
- [5] Chih Hsien Lin, Chih-Ning Chen, Shyh-Jye Jou. Adaptive on-die termination resistors for high-speed transceiver[C]//VLSI Design, Automation and Test. Taiwan:Chungli, 2005.

作者简介:

马 轩 男, (1989—), 硕士研究生. 研究方向为高速串行接口芯片设计.

王自强 男, (1975—), 博士研究生, 讲师. 研究方向为模拟集成电路设计、高速串行接口芯片设计.