软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(8):1746−1760 [doi: 10.3724/SP.J.1001.2013.04361]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

静态程序切片的 GPU 通用计算功耗预测模型*

王海峰[2,3], 陈庆奎[1,2]

[1](上海理工大学 光电信息与计算机工程学院,上海 200093)

[2](上海理工大学 管理学院,上海 200093)

[3](临沂大学 信息学院,山东 临沂 276000)

通讯作者: 陈庆奎, E-mail: chenqingkui@gmail.com, http://oece.usst.edu.cn/

摘 要: 随着图形处理器通用计算的发展,GPU(graphics processing unit)通用计算程序功耗的度量与优化成为绿色计算领域中的一个基础问题.当前,GPU 计算能耗评测主要通过硬件来实现,而开发人员无法在编译之前了解应用程序能耗,难以实现能耗约束下的代码优化与重构.为了解决开发人员评估应用程序能耗的问题,提出了针对应用程序源代码的静态功耗预测模型,根据分支结构的疏密程度以及静态程序切片技术,分别建立分支稀疏和稠密两类应用程序的功耗预测模型.程序切片是介于指令与函数之间的度量粒度,在分析GPU应用程序时具有较强的理论支持和可行性.用非线性回归和小波神经网络建立两种切片功耗模型.针对特定 GPU 非线性回归模型的准确性较好.小波神经网络预测模型适合各种体系的 GPU,具有较好的通用性.对应用程序分支结构进行分析后,为分支稀疏程序提供加权功率统计模型,以保证功耗评估算法的效率.分支稠密程序则采用基于执行路径概率的功耗预测法,以提高预测模型的准确性.实验结果表明,两种预测模型及算法能够有效评估 GPU 通用计算程序的功耗,模型预测值与实际测量值的相对误差低于 6%.

关键词: 功耗模型;GPU 计算;非线性回归;程序切片;小波神经网络

中图法分类号: TP314 **文献标识码**: A

中文引用格式: 王海峰,陈庆奎.静态程序切片的 GPU 通用计算功耗预测模型.软件学报,2013,24(8):1746−1760. http://www.jos.org.cn/1000-9825/4361.htm

英文引用格式: Wang HF, Chen QK. Power consumption prediction model of general-purpose computing GPU with static program slicing. Ruan Jian Xue Bao/Journal of Software, 2013,24(8):1746−1760 (in Chinese). http://www.jos.org.cn/1000-9825/4361.htm

Power Consumption Prediction Model of General-Purpose Computing GPU with Static Program Slicing

WANG Hai-Feng[2,3], CHEN Qing-Kui[1,2]

[1](School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

[2](School of Management, University of Shanghai for Science and Technology, Shanghai 200093, China)

[3](Information School, Linyi University, Linyi 276000, China)

Corresponding author: CHEN Qing-Kui, E-mail: chenqingkui@gmail.com, http://oece.usst.edu.cn/

Abstract: With the development of general-purpose computing of GPUs (graphics processing units), power consumption measurements and optimization have become an essential issue in the green computing field. The current power consumption of GPUs is mainly measured by the hardware. However the programmers have had difficulty understanding the power consumption profile of the applications used to optimize and refactor before the compile phase. To solve this issue, power consumption models were proposed for

* 基金项目: 国家自然科学基金(60970012); 教育部博士点专项基金(20113120110008); 上海重点科技攻关项目(09511501000, 09220502800); 上海市一流学科建设项目(XTKX2012)

收稿时间: 2012-08-03; 修改时间: 2012-10-19; 定稿时间: 2012-12-27

GPU applications with regard to sparseness- branch and denseness-branch programs based on program slicing, respectively. The program slicing is fine-grained level that lies between the function and the instruction levels and has good feasibility and accuracy in the power consumption estimation. The power consumption prediction models for program slicing were proposed through no-linear regression and wavelet neural networks. To specific GPUs, the power prediction model based on no-linear regression is more precise than the prediction model based on wavelet neural networks. However the wavelet neural networks model has better generality to various kinds of GPUs. After analyzing the structure of the applications, the weighted power model for sparseness-branch programs was provided to achieve better effectiveness. The probability slicing power model for denseness-branch programs was also proposed to improve the accuracy that is based on the probability of the execution paths. The results indicate that the two different models can effectively predict the power consumption. And the average relative error between the predicted value and the measured value is less than 6%.

Key words: power consumption model; GPU computing; no-linear regression; program slicing; wavelet neural network

在保证目标性能约束下的能量管理是高性能计算领域中的重要研究方向[1].图形处理器通用计算(general-purpose computing graphics processing unit,简称GPGPU)作为一种性价比高的并行处理方案已广泛应用到科学计算和各种工程计算领域中.目前,GPU 功耗在不断提高,如高端显卡的供电功率已超过 300W,年耗电为 2 628KWh,还需要额外的 748KWh 用于冷却[2].能量作为一种重要计算资源逐渐引起研究人员的关注,GPGPU能耗优化的基础是分析应用程序的功耗、探索能量消耗的分布.开发人员需要了解应用程序计算过程中的能耗.可以通过仪器测量或者统计预测的方式获得计算能耗,然而硬件测量的方式可行性差、实施成本高.因此,GPU通用计算程序的能耗预测成为一个非常有意义的课题.

应用程序能耗预测的主要思路是,将程序分解为更小的度量单元,准确地获得度量单元的能耗值,并使用统计方法得到整个程序的能耗值.以NIVIDA公司的CUDA开发平台为应用背景,用静态程序切片法把核函数分解成粒度更小的代码切片,然后用非线性回归和小波神经网络两种方法建立切片功耗模型.根据应用程序中分支结构的分布特点,对分支稀疏和分支稠密的应用程序分别建立功耗预测模型.通过功耗预测模型为开发者提供一种量化分析源代码能耗的静态方法.在此采用切片作为程序分解粒度,预测精度比函数级分解粒度高;与指令级分析相比,摆脱了对编译器和仿真器的依赖性,降低了分析的难度和计算复杂度,并且具有较好的通用性.

1 研究现状

目前,单节点单任务的GPU能耗优化管理分为两个研究重点.

- 一个重点是功耗优化研究.林一松等人使用GPU仿真器深入分析线程Warp调度机制,建立了访存并行度与计算频率的关系模型,提出了有效的单任务功耗优化策略[3];另外,综合考虑GPU通用计算的异构体系,用 AOV 网分析单任务中各子任务的依赖关系,降低非关键路径上的冗余能耗,达到节能的目的[4].Gebhart和NVIDIA研究员深入GPU芯片体系,通过增加寄存器文件缓存和修改线程调度机制来优化功耗,并取得了理想效果[5].王桂彬等人提出了合并多个独立核函数的方法来实现节能,把核函数聚合转化为一个动态规划问题,函数聚合法可嵌入到编译器的能耗优化设计中[6].
- 另一个重点是GPU能耗的测量与评估.研究硬件功耗的测量方法,并测量NVIDIA GT200和GF100两种体系的指令级功耗,比如数据通信、单精度浮点算术指令和访存指令等[7].Collange测量了3种典型GPU体系中计算和访存指令的功耗,分析了测量结果,为能耗优化提供了基础数据[8].GPU能耗评估的研究较少,文献[9]尝试分析不同计算核的功耗特征,建立各种运算模式与功耗之间的关系模型;Hong等人建立了一个计算和访存并行度的性能模型,并分析了流多处理器 SM(stream multiprocessor)的功耗,通过统计SM执行的指令数来预测SM功耗[10].本文重点是单节点、单任务的GPU计算功耗的预测研究.由于现存的研究成果较少,对GPU功耗预测的研究方法可参考嵌入式软件能耗的预测.

嵌入式软件能耗预测的研究方法可分为 3 个层次:(1) 指令层.从指令角度分析能耗,建立嵌入式软件的能耗预测模型[11,12].针对特定处理器预测较准确,缺乏通用性,而且对仿真器有较强的依赖性.(2) 函数(或子过程)层.函数是嵌入式软件中最小的功能单元.Tan 等人从函数角度提出了一种预测软件能耗的宏模型,用函数级的能耗来刻画整个软件的宏模型,从而对软件能耗进行预测[13].函数是比指令大的分析粒度,从函数级分析软件功

耗能够提高预测模型的通用性.(3) 软件体系层.软件体系层研究的主要工作是描述嵌入式软件的能耗和具体建模方法.Senn 等人提出了以体系结构建模语言 AADL 评估能耗的精化方法[14];张腾腾等人提出了基于进程代数的能耗模型,以接口为研究对象定义系统的最大、最小和平均能耗[15];刘啸滨等人通过提取嵌入式软件的特征量建立 BP 神经网络估算软件能耗的模型,具有一定的通用性,而且预测误差在可接受范围内[16].

选择能耗分析粒度需要考虑预测模型的通用性和准确性的均衡问题,粒度越小精确度越高,粒度越大通用性越好.在 GPGPU 能耗预测研究中,选择程序切片作为能耗分析粒度,程序切片介于指令与函数之间,在保证预测精度的前提下通用性较好.

下面介绍与相关研究方法的比较.本文从程序切片粒度度量功耗,建立功耗与程序特征量之间的非线性拟合模型.文献[17]采用线性回归方法,面对多种输入时回归模型不具有通用性,而且准确性不够稳定.非线性关系包括线性关系,提高了预测的准确性和稳定性.神经网络作为一种数值逼近方法,在数学上能够逼近任意的非线性函数,具有较高的拟合度.刘啸滨等人用传统 BP 神经网络拟合软件体系结构特征与能耗的非线性模型[16].本文第 3.2 节根据实际应用中切片计算密度的分布特点,采用小波神经网络构建回归模型,在样本切片密集的训练阶段高频率学习,而在样本切片稀疏的阶段低频率学习,具有更灵敏的逼近能力和更强的容错能力.

2 背景知识

2.1 功耗与测量

首先要明确能耗与功耗的概念.功耗是衡量系统消耗能量的一个重要概念,GPU 计算功耗是计算程序运行过程中单位时间内能量的消耗,反映 GPU 消耗能量的速率,单位是瓦特(W).GPU 能耗是其运行过程中总的能量消耗,单位是焦耳(J),由时间 t 和功耗共同决定.GPU 计算能耗主要由静态漏电能耗与动态能耗组成,静态漏电能耗与 GPU 硬件体系的设计有关,动态能耗与计算中参与晶体管能耗相关,在此只研究 GPU 动态能耗[18].

GPU 功耗测量是能耗预测的基础.高端 GPU 显卡的供电由两部分组成:主板 PCI-E 总线供电和外接电源.研究表明,运算能耗主要来自外接电源,PCI-E 总线只提供 10W~15W 的供电,在总体能耗中所占比例甚少,可忽略 PCI-E 总线供电产生的能耗[8].目前,较准确的测量方法是利用电流探头测量显卡外接电源的电流,电流探头把电流转换为电压信号,再由数字示波器测量电压信号来计算功耗[7,8].这种方法具有采样频率高和准确性好的优点,但是记录采集数据时存在困难.

为了解决实时记录功耗数据的问题,文中设计了功耗采集卡,测量外接电源的电流获得 GPU 能耗.功耗采集卡工作原理如下:先由电流传感器 ACS713-30T 将电流转换为电压值,采集卡上的微控制器 ATmega168 把电压模拟信号转化为数字信号;然后,USB 控制器 FTDI FT232RL 芯片将采集数据传输到计算机中保存.该测量方案不仅采样精度高,而且测量精度高.实验中,应用程序的执行功耗采用如下方式获得:由于在计算过程中功耗动态变化,因此以程序执行中各采样点的平均功耗作为执行功耗的近似值.设在程序执行过程中各采样点的测量功耗为 $\{P_1,P_2,...,P_s\}$,则平均采样功耗为该功耗序列的平均值.

2.2 静态切片法

静态切片(program slicing)是一种经典程序分解技术,广泛用于程序调试、维护和测试领域.切片是一个变量及影响变量值的程序语句集合.切片把程序转化为只包含与程序分析相关的语句集合,缩小了程序分析和理解的范围[19,20].

定义 1. GPU 通用计算程序是具有特定逻辑关系的核函数集合,形式化表示为 $P=\{K_1,K_2,\ldots,K_m\}$.

定义 2. 核函数(kernel function)是程序切片集合,形式化表示为 $K=\{C_1,C_2,C_i,\ldots\}$.其中,C_i 表示程序切片,C_i 为一个三元组(K,S,V),K 是切片所属的核函数;S 为与变量 V 相关的程序语句子集,即 $S=\{s^1,s^2,\ldots,s^k\}$;V 表示在 S 中引用的变量集合,$V=\{v_1,v_2,\ldots,v_m\}$.

下面以一种形式化的方式简要介绍静态切片法.

设核函数 $K=\{s^1_{v_1,v_2},s^2_{v_2},s^3_{v_3},s^4_{v_1},s^5_{v_3},s^6_{v_2},s^7_{v_1},s^8_{v_1,v_3},s^9_{v_2},s^{10}_{v_3},s^{11}_{v_1},s^{12},s^{13},s^{14}_{v_1},s^{15}_{v_3},s^{16}_{v_1}\}$,其中,$s$ 代表核函数中的指令,上标

表示指令序号,下标表示该指令涉及的变量;K 中的变量集合 $V_c=(v_1,v_2,v_3)$.于是,在核函数中,变量 v_1 的静态切片 $C_1=(K,S_1,v_1)$,$S_1=\{s_1,s_4,s_7,s_8,s_{11},s_{14},s_{16}\}$.需要注意的是,静态切片有多种划分形式,如根据变量集的幂集可分解为多种切片,如 $K=(C_1,C_2,C_3)$,$C_1=(K,S_1,v_1)$,$C_2=(K,S_2,v_2)$,$C_2=(K,S_3,v_3)$;或 $K=(C_1,C_2)$.其中,$C_1=(K,S_1,v_1)$,$C_2=(K,S_2,v_2,v_3)$.

总之,程序切片是一种介于函数与指令之间的单元,能够有效分解数据依赖性.切片比软件体系和函数这两种粒度的预测精度高.此外,开发人员可了解源代码的功耗分布情况,并重构功耗高的程序切片.

2.3 功耗影响因素

应用程序的计算密度和线程配置是影响GPU能耗的两个因素.计算密度是衡量计算性能的一种基本准则,其数学形式为应用程序中算术操作和存储器访问操作的比率[21].GPU通用程序中的变量分为4类,存储在全局显存(global memory)、共享显存(shared memory)、常量存储区(constant memory)和纹理存储区(texture memory),记为GM,SHM,CM,TM.研究表明,不同访存操作的功耗不同,利用静态切片法能够明确区分4种访存操作.由于读写不同存储区域的功耗不同,需要从功耗角度归一化访存操作数,这里引入功耗因子 r 来统一访存数量[22].由于全局显存访问操作占比例最高,因此用全局显存的功耗因子 r_0 作为基准,$r_0=1$.共享存储器、常量存储器和纹理存储器的比例系数分别为 r_1,r_2 和 r_3.若程序切片 C_i 中全局显存、共享存储器、常量存储器和纹理存储器的读写语句为 n_0,n_1,n_2 和 n_3,则归一化操作如公式(1)所示,得出切片 C_i 的访存操作数 N_{mem}.

$$N_{mem}=\sum_{i=0}^{3}r_i\times n_i \tag{1}$$

定义3. 基于功耗的计算密度是计算指令与归一化访存指令的比值:

$$\tilde{A}=N_a/N_{mem} \tag{2}$$

应用程序启动的线程规模是另一重要因素.例如,程序可启动512或者128个线程进行计算,不同线程配置使得活跃流多处理器数目不同,随着活跃流多处理器的增多,功耗相应增加[10].为了量化计算过程中活跃SM数,引入SM饱和度的概念.

定义4. SM饱和度是活跃SM数与GPU中全部SM的比值,记为 Sa.通过调节线程块Block的数量来控制活跃SM的数量,重点考虑30%,60%,80%,100%这4种情况.

3 切片功耗模型

本节建立程序切片功耗模型来实现各类切片功耗的预测,为下一步程序功耗预测奠定基础.在大量切片功耗数据的基础上,以统计方法或离线学习的方式建立回归分析和小波神经网络两种预测模型.切片功耗测量的实验要求如下:选取NIVIDA的Tesla C870,GeForce GTX 260,GTX280及Fermi体系的GTX480作为测试对象.程序切片功耗具体的测量方法如下:由于程序切片的本质是指令序列,将待测量的切片设计成可执行的GPU核函数来仿真切片,此后称这类核函数为仿真切片.为了保证功耗采集的准确性,以循环方式重复运行仿真切片来增加执行时间,以仿真切片的平均采样功耗作为程序切片的功耗.在此,设计20种特定计算密度的核函数作为测试的仿真切片,计算密度分布在[7.1,8200]区间中,SM饱和度控制在[0.1,1]范围内;为了保证采样的正确性,仿真切片的连续运行时间必须大于500ms,然后,20种仿真切片分别在4种典型的GPU上执行,测试其功耗并统计规律.

3.1 回归预测模型

传统的嵌入式软件能耗预测模型主要使用线性回归分析方法,建立软件功耗与程序特征量之间的 n 元线性关系[17]:

$$E=C_1+C_2S_1+C_3S_2+\ldots+C_{n+1}S_n \tag{3}$$

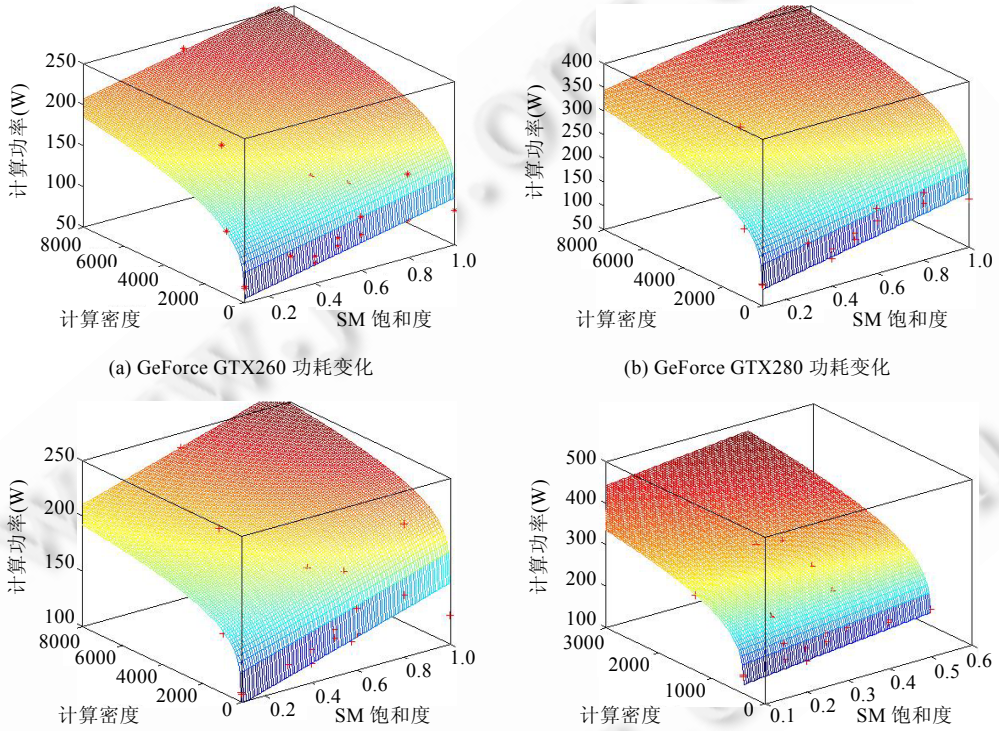
其中,E 表示程序能耗,C_1 表示初始化时的能耗,S_n 表示影响程序能耗的特征量,C_n 表示各个特征量的参数.线性回归分析存在如下缺陷:(1) 程序特征量与能耗存在线性关系的依据不充分;(2) 线性回归模型的预测误差较大,不能满足实际需求.在此,认为GPU通用计算程序的能耗与计算密度、SM饱和度存在非线性函数关系,通过

二元非线性回归分析建立切片功耗预测模型.计算密度和 SM 饱和度对功耗的影响呈现如下规律:开始影响效果不明显,随着计算密度和 SM 饱和度的增加,影响效果显著增加,随后又变缓慢,直至稳定.依据上述变化规律,选择公式(4)的渐近型非线性回归函数:

$$P(x_1,x_2)=\beta_0 x_1+\beta_1 x_2^{\beta_2}+\varepsilon \tag{4}$$

再用最小二乘法拟合预测模型,对于测量结果序列$(x_{1i},x_{2i})(i=0,1,\ldots,n)$,在取定的函数类$\Phi$中求$P(x_1,x_2)\in\Phi$,使误差$r_i=p(x_1,x_2)-y(x_1,x_2)$的平方和最小.从几何意义上讲,就是寻求与给定点集合$(x_{1i},x_{2i})(i=0,1,\ldots,n)$的距离平方和为最小的曲面$z=f(x,y)$.函数$P(x_1,x_2)$成为拟合函数,求拟合函数的方法为曲面拟合的最小二乘法[23].这里,以 20 个仿真切片功耗测量值为原始数据,用 Matlab 中的 nlinfit 函数进行各种 GPU 的功耗与计算密度、SM 饱和度的非线性最小二乘拟合,nlinfit 函数用高斯-牛顿算法进行最小二乘拟合.

图 1 所示为 4 种 GPU 功耗拟合曲面,其中,x 轴表示计算密度,y 轴表示 SM 饱和度,z 轴为计算功耗.图 1(a)~图 1(d)的拟合函数见表 1.



(a) GeForce GTX260 功耗变化　(b) GeForce GTX280 功耗变化

(c) Tesla C870 功耗变化　(d) GeForce GTX480 功耗变化

Fig.1 Variation of power consumption of different GPUs

图 1 不同 GPU 的功耗变化趋势

Table 1 Regression functions of different GPUs power consumption

表 1 各类 GPU 功耗回归函数

GPU 类型	功耗预测模型
GTX260	$p(x_1,x_2)=65.6x_1+29.4x_2^{0.2}$
GTX280	$p(x_1,x_2)=95x_1+46.7x_2^{0.2}$
C870	$p(x_1,x_2)=62.4x_1+75.8x_2^{0.1}$
GTX480	$p(x_1,x_2)=98.7x_1+102.3x_2^{0.15}$

GeForce与Tesla是NIVIDA的不同产品系列.Tesla C870是面向专业计算的GPU,GeForce系列的GTX260和GTX280属于第二代体系架构GT200的典型产品,GTX480是采用最新Fermi体系的处理器.在表1的预测模型中,x_1为SM饱和度,x_2为计算密度.比较可见,拟合参数β_0,β_1与GPU类型有关,而参数β_2与GPU体系有关.为了验证拟合函数的有效性,再设计40个不同计算密度和SM饱和度的仿真切片,用预测值与实际测量值的残差$\varepsilon = P - \hat{P}$对二元非线性拟合模型进行评估.GeForce GTX280的残差曲线如图2所示.

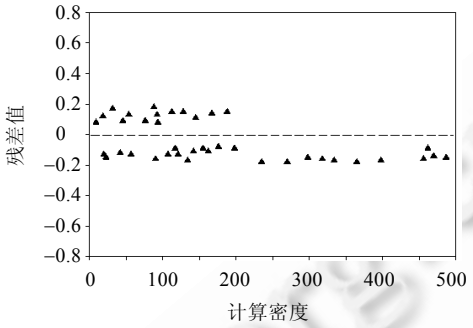


Fig.2 Residual difference of fitting power consumption prediction value

图2 拟合功耗预测值的残差

由图2可见,二元非线性拟合模型的预测值与实测值接近,最大残差为0.18,最大误差为3.98%.图中可观察到残差分布较均匀,尤其是在计算密度小于200时的分布更合理,并且整个图中无突变的较大误差出现.再进行统计检验,统计检验的结果表明,残差满足以下两个条件:(1) 独立同分布;(2) 在独立同分布的基础上,残差满足标准正态分布.统计检验的证明略.

上述结果说明:非线性拟合模型是正确的;然而面对各种GPU时,回归模型缺乏通用性,需要为不同GPU确定模型的参数.此外,非线性回归模型建立在假设基础上,用于回归分析的数据集尚不充分、完备,因此模型的准确性不够稳定.

3.2 小波神经网络预测模型

为了提高切片功耗预测的通用性,本节提出利用小波神经网络在程序切片级估算能耗的方法,能耗预测模型对4个程序特征量进行度量,拟合出程序特征量与能耗的非线性函数关系.小波神经网络(wavelet neural networks,简称WNN)集中了小波分析和人工神经网络的优点,具有神经网络的学习、泛化及非线性映射能力.同时,小波函数具有多尺度时频分析手段和良好的逼近特性,WNN比BP神经网络有更强的自适应能力和更高的预测精度.

首先确定切片的特征量,分别为计算密度、SM饱和度、代码行数L_i、GPU类型.基于能耗的计算密度是一个重要的特征.研究表明,应用程序的计算密度具有一定的分布规律,存在明显的稠密区和稀疏区.选择3个基准程序集CUDASDK[24],Parboil[25]和Rodinia[26],统计35个应用程序的计算密度分布,如图3所示.

由图3可见,应用程序的计算密度存在稠密和稀疏区间,如稠密区间[0,90]和[400,500],稀疏区间[200,400].小波神经网络继承了小波分析多尺度分辨率的特点,在训练数据分布的稠密区域以高分辨率学习,而在稀疏区以低分辨率学习,这是选择小波神经网络拟合非线性模型的主要原因.SM饱和度量化了GPU中的活跃SM,取值范围为[0,1],在此作为第2个特征量.大量研究结果表明,每执行一条二进制指令都有一个固定能耗值,指令数越多,产生的能耗就越大.代码行数在一定程度上反映切片的规模,同时与能耗也有密切关联,因此可作为第3个能耗特征量.最后一个特征量是GPU类型,不同GPU体系能耗存在差异,把参与训练的GPU类型按顺序进行编号:(0,1,…),用编号作为输入值.

小波神经网络采用紧致型拓扑结构,如图4所示.

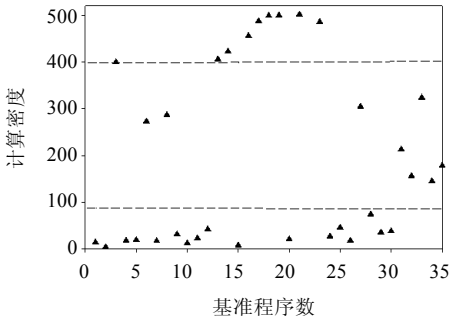


Fig.3 Arithmetic density distribution of benchmark

图 3 基准程序集计算密度分布

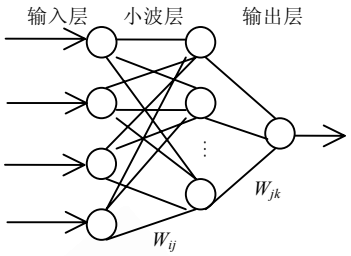


Fig.4 Framework of wavelet neural networks

图 4 小波神经网络的结构

紧致型结构不改变传统神经网络的拓扑结构,只是把神经元(neuron)替换为小波元(waveron),将小波函数融合在神经网络中.输入层 4 个节点,负责输入 4 种特征量;小波层决定网络误差,本质是神经网络的隐含层.由于单隐含层就能逼近任何闭区间的连续函数,实现任意 n 维到 m 维向量的映射.因此,拓扑结构的小波层为 1[27].采用 BP 网络中的经验公式确定小波层节点数为 11;输出层节点数为 1,神经元数量为 16.小波函数为 Morlet 余弦调制小波,因为 Morlet 连续可导,具有时频局部化好的特性,误差小而且抗干扰能力强,如公式(5)所示:

$$y=\cos(1.67x)\mathrm{e}^{-x^2/2} \tag{5}$$

输出层神经元函数为双曲正切函数,用该激励函数时收敛速度较快.

$$s(x)=a\frac{1-\mathrm{e}^{-bx}}{1+\mathrm{e}^{-bx}}=\frac{2a}{1+\mathrm{e}^{-bx}}-a \tag{6}$$

其中,公式(6)中的参数取值一般为 a=1.716,b=0.667.小波神经网络的值为

$$y(k)=\sum_{j=1}^{l}w_{jk}h(j),\ k=1,2,...,m \tag{7}$$

其中,l 为小波层节点数,m 为输出层节点数,$h(j)$为第 j 个小波层节点的输出.网络权值、小波函数伸缩因子和平移因子的调整规则与 BP 神经网络权值修正法相似.为达到误差最小的目的,采用梯度修正法来调整网络的权值和小波基函数参数.其修正过程如下:先计算网络预测误差为

$$e=\sum_{k=1}^{m}Y(k)-y(k) \tag{8}$$

其中,$Y(k)$为实际输出值,$y(k)$为预测输出值.为使误差 e 最小,采用文献[27]中的方法调整网络权值和小波基函数的系数.

WNN 构造完成后,非线性拟合的具体步骤如下:

(1) 提取大量仿真切片的计算密度、SM 饱和度、代码行、GPU 类这 4 个程序特征量,并测量仿真切片的平均功耗.

(2) 对 4 个特征量进行预处理,处理后的值作为 WNN 的输入值,测量功耗为训练阶段的输出值.

(3) 初始化 WNN,例如伸缩因子、平移因子以及网络权值 w_{ij} 和 w_{jk}.

(4) 输入仿真切片的输入值和期望输出值,根据文献[27]中的方式修正权值.若误差函数小于预先设定值,则停止网络学习;否则继续学习,直到收敛为止.

(5) 训练阶段结束后,输入验证程序切片的特征量到 WNN,得到功耗预测值,再与测量功耗进行对比,验证拟合的有效性和精度.

最后,用第 3.1 节的 40 个验证切片来测试 WNN 的预测效果,实际测量值与预测值的最大残差是 0.58,最大误差为 5.87%.实验结果表明,WNN 的预测精度低于非线性回归模型,但是通用性明显比回归模型要好,适合处理各种类型的 GPU.

4 程序功耗预测模型

应用程序经过分解后变成切片集合,在切片功耗模型基础上可进一步研究程序的功耗评估模型.下面以Nvidia公司的CUDA平台程序为例来分析GPGPU程序的特点.GPGPU程序主要采用C语言开发,以结构化设计为核心编程模式.由于一个重要的设计原则是尽量提高数据并行处理的吞吐量,因此在该原则的约束下,GPGPU程序与普通程序有两个不同的结构特点:(1) 为了提高线程并发执行,需要杜绝线程调度中出现Warp串行执行的情况,尽量减少分支逻辑.(2) 为了发挥海量线程的并发处理能力,需要通过多线程的方式展开循环体,以减少循环结构;对于无法展开的循环体,尽量设计成简单循环,即循环次数和步长已知的循环结构.总之,GPGPU程序分支结构相对较少,并且基本上只有简单循环结构.

经过以上分析后,本文将GPGPU程序划分成分支稀疏和分支稠密两大类.由于防止出现串行化线程调度,因此在设计原则的约束下,分支稀疏的程序所占比例较大.然而,许多复杂算法移植到GPU后很难避免出现复杂分支逻辑,例如图论中的典型算法、人工智能算法等,因此也存在一定比例的分支稠密程序.为了提高功耗预测模型的针对性,分别为分支稀疏和稠密的两类程序建立功耗预测模型.

4.1 分支稀疏程序的功耗预测模型

GPU通用计算中,大部分的程序具备分支稀疏的特点,在切片功耗模型的基础上,用统计的方法建立应用程序的功耗预测模型.

定义 5. 程序切片长度是切片内有效执行语句数,$|C_i|$表示切片 C_i 的长度.有效代码是指通过编译器能够转化可执行机器指令的代码,比如变量计算、变量赋值、函数初始化及调用等.

定义 6. 程序能耗是所有切片能耗之和,如公式(9)所示:

$$E=\sum_{i=1}^{m}(P_i\times t_i) \tag{9}$$

其中,P_i 是切片 C_i 的计算功率,t_i 是切片的执行时间.

在实际应用中,确定各切片的执行时间非常困难,定义6只是一种理想能耗模型.可方便地获得应用程序的执行时间 t,因此用切片长度来估算切片执行时间,如公式(10)所示:

$$\overline{t_i}=\frac{|C_i|}{L}\times t,\ L=\sum_{i=1}^{m}|C_i| \tag{10}$$

其中,$\overline{t_i}$ 为切片 C_i 的估算执行时间,t 为程序执行时间,L 为程序有效代码行数,m 为程序中切片数.应用程序的估算能耗为

$$\overline{E}=\sum_{i=1}^{m}(P_i\times\overline{t_i}) \tag{11}$$

预测能耗与实际测量能耗的误差为 $\varepsilon=|\overline{E}-\overline{P}\times t|$.为方便对比,只关注应用程序的功耗.应用程序的实际测量功率取采样点的平均功率 $\overline{P}$,预测功耗取各切片功耗的加权求和值,如公式(12)所示:

$$\hat{P}=\sum_{i=1}^{m}\left(P_i\times\frac{|C_i|}{L}\right),\ \overline{E}=\hat{P}\times t \tag{12}$$

预测误差 $\varepsilon=|\hat{P}-\overline{P}|$.这种分支稀疏程序的功耗预测法简记BSPM(branch-sparseness prediction method).

4.2 BSPM关键技术

BSPM的关键技术是如何求解基于功耗的计算密度 $\tilde{A}$.在扫描源代码的过程中,由变量存储类型实施程序分解,统计切片中计算与访存的数量,求得计算密度和切片长度.在基于功耗的计算密度求解中,需要解决以下3个问题:

(1) 正确区分访存操作的类型.因为读写不同存储区造成的功耗不同,所以要正确区分访存操作的类型.全局显存操作最为普遍,因此作为功耗基准访存操作.NIVIDA的CUDA中,用编译限定符定义存储区[24],用解析源程序中的限定符来区分访存类型.然而,全局存储区没有特定编译指示符,可用扫描存

储分配函数名的方式实现区分,如 cudaMalloc 等函数.

(2) 正确区分计算与访存语句.在源代码扫描的过程中,用操作符为判断依据进行词法分析.当解析到一个计算操作符时,标记一次计算语句;当检测到赋值操作符时,则标记为一次访存语句.由于普通 C 语句中计算与访存操作紧密结合,可把普通语句分解成多条逻辑语句.如语句(*Vect0)=m0+m3,将该语句分解为一条算术逻辑语句 m0+m3 和一条访存逻辑语句.此外,因为加减乘除指令功耗相似[10],计算操作不再细分.

(3) 循环语句展开.由于算法逻辑的复杂性,不可能完全避免循环结构,应用程序中仍然有较多简单循环结构.提取关键循环体是计算体系优化和编译研究领域的经典问题.采用文献[28]中的方法将程序切片划分为基本块,每个基本块由普通语句或为复合语句组成;然后,用 For,While 等关键词抽取基本块中的循环体,扫描结束后,为每个逻辑语句增加一个计数值,计数值为这条语句循环执行的次数.

算法 1. 切片中循环体展开算法.

输入:切片 $C\{B_1,B_2,...,B_k\}$.

输出:展开后的切片 $C\{S_1,S_2,...,S_n\}$.

1. For B_i=B_1 to B_k do
2. For each S_j in B_i do
3. S_j.*count*=1;
4. Endfor
5. Endfor
6. For B_i=B_1 to B_k do
7. If B_i is LoopBlock then
8. Search the *iteration_number* in B_i;
9. For each S_j in B_i do
10. S_j.*count*=*iteration_number*;
11. Endfor
12. Endfor
13. Return $C\{S_1,S_2,...,S_n\}$

首先,初始化切片中各语句的计数域 *count* 为 1(第 1 行~第 5 行),再扫描源程序中的 For,While 等循环标记来提取循环体的基本块(第 7 行),当一个循环基本块确定后,再解析循环语句来寻找循环次数(第 8 行);然后,为循环基本块中的每个逻辑语句的计数域 *count* 赋值(第 9 行~第 11 行);最后,切片 *C* 是循环展开后的逻辑语句集(第 13 行).

在解决这 3 个关键问题后,介绍 BSPM 算法.首先统计切片中基于功耗的计算密度.设根据 4 类存储区域将程序划分为 4 个切片集$\{C_{gm},C_{shm},C_{cm},C_{tm}\}$.每个切片子集中的切片之间不存在语句交集,$C_{gm}^i \cap C_{gm}^j = \varnothing$,计算操作符号集和访存符集分别记为 OP^c,OP^m,BSPM 算法如下:

算法 2. 基于功耗的计算密度求解.

输入:切片 $C_i\{S_1,S_2,\ldots,S_n\}$,操作符集 OP^c,OP^m.

输出:切片基于功耗的计算密度 $\overline{A}$.

1. For C_j=C_{gm} to C_{tm} do
2. If $C_i \in C_j$
3. r=r_i
4. Endfor
5. For each C_{ij} in C_i do
6. For each S_i in C_{ij} do

7. If $S_i.op \in OP^c$ then
8. $\#A=\#A+S_i.count$;
9. Else $S_i.op \in OP^m$
10. $\#M=r\times(\#M+S_i.count)$;
11. Endfor
12. Endfor
13. $\overline{A}=\#A/\#M$;
14. Return $\overline{A}$.

首先,根据变量的存储类型来确定归一化因子 r(第 1 行~第 3 行);再扫描切片中有效代码行,若语句中存在计算操作符,则计算统计量$\#A$增加(第 7 行、第 8 行);若存在访存操作,则访存统计量$\#M$增加(第 9 行、第 10 行)并且乘以归一化因子;最后,基于功耗的计算密度是运算统计量$\#A$与访存统计量$\#M$的比值(第 13 行).

算法 3. BSPM 算法.

输入:应用程序$\{C_1,C_2,\ldots,C_m\}$.

输出:预测功耗 $\hat{P}$.

1. For $C_i=C_1$ to C_m
2. For each S_i in C_i
3. $L_i=L_i+1$;
4. Endfor
5. For i =1 to m
6. $\hat{P}$ +=$P_i\times L_i/L$
7. Endfor
8. Return $\hat{P}$

在算法 3 中,首先统计程序各切片的有效代码行数(第 1 行~第 3 行),然后遍历切片集,由公式(12)求出加权后的预测功率(第 5 行、第 6 行).

4.3 分支稠密程序的功耗预测模型

在实际应用中,还有一些计算逻辑复杂的程序移植到 GPU 中,第 4.1 节的 BSPM 在处理这些分支稠密的程序时存在缺陷.本节提出一种针对分支稠密程序的功耗评估方法.首先以具体实例来介绍 BSPM 的局限性.

设一个分支稠密应用程序 $P=\{s^1,s^2,s^3,s^4,i^5,i^6,s^7,s^8,i^9,s^{10},\ldots\}$,其中,$i$ 表示 if 分支指令,s 表示普通指令.假设语句 s^8 计算密度比较大,在实际运行中,其执行概率仅为 0.001.由于语句 8 的计算密度值比较大,在 BSPM 中,语句 8 对切片功耗的影响最大.然而实际执行过程中,由于其执行概率小,可忽略对功耗的影响.由此可见,预测分支稠密程序时 BSPM 会出现误差,预测准确性不稳定.

为了提高分支稠密程序功耗预测的准确性和稳定性,用概率切片的方法解决程序中分支执行路径对计算密度的影响.概率切片的主要思想是,以一定概率阈值来限制执行次数少的切片对程序计算密度的影响.例如,程序切片 C_i 包含子切片 C_{i1} 和 C_{i2},若 C_{i2} 执行的概率低于某阈值,则忽略 C_{i2} 对计算密度的贡献,以 C_{i1} 的计算密度作为 C_i 的计算密度.

定义 7(有效计算密度). 设切片 C_i 中的子切片为$\{C_{i1},C_{i2},\ldots,C_{ik}\}$,若 C_i 中大于执行概率阈值的子切片集为 $\hat{C}_i=\{C_{i1},C_{i2},\ldots,C_{im}\}$,则切片有效计算密度为 $\hat{C}_i$ 中计算与访存指令的比值.

对于分支结构稠密的应用程序,用有效计算密度替代第 3.1 节和第 3.2 节中的计算密度,修正回归拟合模型和小波神经网络的切片预测模型,应用程序的预测功率 $\hat{P}$ 如公式(13)所示:

$$\hat{P}=\sum_{i=1}^{m}\left(P_i\times\frac{|\hat{C}_i|}{\hat{L}}\right),\ \hat{L}=\sum_{i=1}^{m}|\hat{C}_i| \tag{13}$$

其中,P_i 是切片的预测功率,$\hat{C}_i$ 为程序中有效切片集,$\hat{L}$ 为有效代码行数.分支稠密程序的功耗预测方法简记为BDPM(branch-density prediction method).

4.4 BDPM关键技术

BDPM 的关键技术是切片执行概率的判断和计算.现有 3 种方法:(1) 静态评估法.依据领域专家对程序执行流程的认知和判断,对分支进行预测[29,30].静态法简单,然而准确性差.(2) 动态分析法.在运行环境中,根据动态变化的输入数据,不断更新分支结构执行的概率.动态分析法准确性最高,但是应用中很难实施.(3) 拟动态分析方法.该方法基于一种假设,即历史数据对未来数据有正确指导意义.选择典型的测试数据集,测试并统计程序中各分支的执行概率,用历史数据的执行概率作为未知数据的分支预测值[31].由于准确性高于静态法,而且可实施性强,本文选取该方法获取切片的分支执行概率.

设切片 $C_i=(C_{i1},C_{i2},...,C_{im})$的子切片的执行概率值为 $p_i=\langle p_{i1},p_{i2},...,p_{im}\rangle$,用拟动态分析法作路径剖析(path profiling)求解概率向量 p_i.具体过程如下:(1) 首先,向各切片中插入探针变量(probe)及操作代码;(2) 其次,在程序执行过程中计算探针值并收集相关信息;(3) 最后,依据相关信息获取各分支路径的执行次数,并求出执行概率[32].

设切片 C_i 的测试数据集 $D=\{d_1,d_2,\ldots,d_s\}$,子切片执行概率 p_{ii} 由公式(14)得出.

$$p_{ii}=\sum_{i=1}^{s}(\alpha_i^{di}/\alpha_{di})\Big/s \tag{14}$$

其中,α_i^{di} 表示子切片 C_{ii} 在测试数据集 d_i 时的执行次数,α_{di} 则是数据集 d_i 全部子切片的执行次数.

求解了分支的执行概率以后,对子切片可用定义 7 求得有效计算密度为$\langle a_{i1},a_{i2},...,a_{im}\rangle$,分支稠密程序的预测功耗算法如下:

算法 4. BDPM 算法.

输入:各切片功率$\langle P_1,P_2,\ldots,P_n\rangle$,各切片执行概率向量$\langle p_1,p_2,...,p_i,...,p_n\rangle$,$\delta$.

输出:预测功耗 $\hat{P}$.

1. For $C_i=C_1$ to C_n do
2. For $C_{ij}=C_{i1}$ to C_{im} do
3. If $p_{ij}>\delta$ then
4. $C_{ij}\subset\hat{C}_i$
5. End for
6. $L+=|\hat{C}_i|$
7. End for
8. For $C_i=C_1$ to C_n do
9. $\hat{P}+=P_i\cdot|\hat{C}_i|/L$
10. Endfor

算法 4 先统计各切片中的有效代码行数(第 2 行~第 5 行)和程序的有效代码行数(第 6 行),再由公式(13)得出程序的预测功耗(第 8 行~第 10 行).

5 实验分析

实验的目的是验证功耗预测的有效性和准确性.测试大量分支稀疏和分支稠密程序,比较测量值与预测值之间的相对误差,以此评价功耗预测方法.实验中采用第 2.2 节介绍的方法测量应用程序的执行功耗.为了提高实验的准确性,先不考虑对各种 GPU 的适应性,选取第 3.1 节中的非线性回归作为切片功耗预测模型.针对NVIDIA GT200 和 Fermi 两种体系,选择 GT200 体系的 GeForce GTX 280 和 Fermi 体系的 GeForce GTX 480 作为实验对象.具体实验环境是 Intel Core2 2.33GHZ,2GDDR RAM,320G 硬盘,NVIDIA GeForce GTX 280,核心频

率为602MHZ,存储频率1107HZ;NVIDIA GeForce GTX 480,核心频率为700MHZ,存储频率3696HZ.操作系统为WindowsXP Profession,CUDA toolkit 2.3,驱动程序版本为190.29.由于Perl语言具有文本和字符串处理的强大功能,可降低编程难度和提高开发效率,用Perl语言实现功耗分析预测原型系统.

5.1 BSPM的验证

为了验证BSPM的有效性,表2列出了8种分支稀疏的实验程序,选自文献[25,26]和CUDA SDK开发包[24].实验中,通过调整程序的线程配置方案来改变活跃SM的数量,而以此调整不同的SM饱和度,*Sa*分布在0.3~0.8之间;计算密度分布在5.7~287区间,覆盖了基准程序集85%的计算密度.由表2可见,预测值小于和大于实际测量值的情况均有出现,并无规律可循,但是预测值与测量值的相对误差全部低于 6%.实验结果表明,BSPM 能够有效地预测分支稀疏程序的计算功耗.

Table 2 Comparison of power consumption prediction value of GTX280

表2 GTX280功耗预测数据对比

测试程序	功能描述	*Sa*	$\overline{A}$	预测值(W)	实测功耗(W)
Dotp	矩阵点积	0.5	20.5	114.6	119.5
Madd	矩阵乘加	0.3	23.3	101.2	105.6
Dmadd	双精度矩阵乘加	0.3	29.1	102.1	108.2
Mtrans	矩阵转置	0.4	5.7	86.8	92.3
scalarProd	矢量缩放	0.55	12.1	102.2	106.5
fwtBatch1	快速瓦尔希变换	0.8	37.8	123.7	119.4
scan	扫描	0.6	35.5	117.6	122.2
256histogram	直方图	0.45	287	131.8	126.5

5.2 BDPM的验证

在验证处理分支稠密程序时,BDPM优于BSPM.首先选择5个分支稠密的应用程序,见表3.离散余弦变换DCT(discrete cosine transform)分支结构较少,而字符串匹配算法和H.264解码预测算法存在大量分支结构,特别是H.264预测算法则是典型稠密分支程序[33].其次,为了简化BDPM实验,5个应用程序的线程配置全部使用SM满载,即*Sa*=1.由于离散余弦变换中的分支语句相对较少,BDPM得出的有效计算密度小于BSPM中的计算密度,因此,由公式(13)计算得出的功耗预测值小于公式(12)的预测值,而且更接近实际测量值;在字符串匹配算法中,随着分支结构的增多,BDPM的有效计算密度与BSPM的计算密度差距增大,因此,BSPM的预测值与实测值的相对误差增加;在H.264解码预测算法中,BSPM的预测值与实测值的误差最大,已经达到13.8W.实验结果表明,BDPM在预测分支稠密程序时准确性优于BSPM.

Table 3 Comparison of two different power consumption prediction methods

表3 两种方法功耗预测对比

测试程序	功能描述	$\overline{A}$	*Sa*	BSPM (W)	BDPM (W)	实测功耗(W)
DCT	离散余弦变换	186.5	0.6	125.4	118.5	120.6
cugrep	字符串匹配	11.3	0.7	110.5	121.1	116.2
binomialOptions	金融分析示例	273	0.3	120.7	124.8	127.3
BlackScholes	金融分析示例	18.1	0.8	116.6	120.7	124.5
cuh264	H.264解码	27.6	0.8	117.5	135.7	131.3

5.3 不同体系的对比

NIVIDA的第二代芯片体系GT200与第一代GT80比有巨大变化,而第三代体系Fermi又完成了部分重要改变,大幅度增加了 CUDA 核心数量,提供二级缓存体系,因此,不同芯片体系功耗必然存在差异.为了进一步验证功耗预测方法对不同芯片体系的适应性和鲁棒性,对GeForce GTX 480显卡重复上述实验,比较GT200体系的GTX280和Fermi体系的GTX480两种GPU产品功耗.另外,由于SM饱和度影响计算功耗,因此配置不同的线程规模来重复实验,考虑活跃SM分别为饱和、近似饱和、半饱和状态,即SM饱和度为*Sa*=1,0.8,0.6这3种

情况.两种体系 GPU 计算时的实测值与预测值的相对误差比较见表 4.

Table 4 Comparison of relative error of two GPU architectures with different SM saturations

表 4 两种 GPU 体系在不同 SM 饱和度下的相对误差对比

测试程序	功能描述	相对误差(*Sa*=1)		相对误差(*Sa*=0.8)		相对误差(*Sa*=0.6)	
		GTX200	Fermi	GTX200	Fermi	GTX200	Fermi
分支稀疏	Dotp	4.22	4.26	4.14	4.32	4.31	4.33
	Madd	4.25	4.28	4.23	4.08	4.29	4.32
	Dmadd	5.82	5.67	5.61	4.55	5.79	5.62
	Mtrans	5.93	5.88	5.90	5.34	5.87	5.65
	scalarProd	4.17	4.33	4.07	4.26	4.21	4.42
	fwtBatch1	3.83	4.12	3.60	3.95	3.69	3.97
	scan	4.12	4.23	4.07	4.20	4.36	4.34
	256histogram	4.42	4.82	4.24	4.57	4.28	4.63
分支稠密	DCT	1.85	2.97	1.74	2.85	1.78	2.96
	cugrep	4.47	4.22	4.28	3.96	4.39	4.14
	binomialOptions	2.35	3.08	2.11	2.93	2.32	3.13
	BlackScholes	3.46	3.91	3.13	3.69	3.27	3.81
	cuh264	3.62	4.41	3.44	4.27	3.56	4.49

由表 4 可见:对于分支稀疏应用程序,BSPM 在两种体系下的相对误差均小于 6%,大部分处于 3.9~4.5 之间;在两种 GPU 体系下,BDPM 的相对误差均小于 4.5%,主要处于 2.0~3.6 之间.通过两种 GPU 体系下相对误差的对比,说明两种功耗预测法针对性强,对于GPU硬件体系的变化具有较好的鲁棒性.此外,对比不同SM饱和度的实验结果可见,当 GPU 中活跃 SM 处于渐进饱和状态时(*Sa*=0.8),预测模型的相对误差低于饱和及半饱和状态.该现象说明,功耗预测模型最适合活跃 SM 渐进饱和的应用场景.GPU 应用程序设计中配置线程规模太小,比如活跃 SM 半饱和状态时,GPU 并行计算性能未得到充分发挥;而当线程规模较大,GPU 满负载工作时能耗及稳定性无法保证.因此,线程规模达到活跃 SM 渐进饱和状态是一种理想的设计目标[10]。该功耗预测模型符合实际需求,具有良好的应用价值.

6 总结与展望

选择程序切片作为程序分解的合理粒度,提取切片中与能耗相关的特征量,如计算密度和 SM 饱和度等.用非线性回归拟合建立针对各种 GPU 的切片能耗模型,预测准确性较高;为了建立适合各种 GPU 体系的通用能耗预测模型,采用小波神经网络拟合特征量与能耗之间的非线性关系,因为小波函数的时频多尺度特性提高了神经网络的逼近和容错能力.可根据实际应用的需求选择这两种切片能耗模型,对预测精度要求低而对 GPU 适应性要求高的场景选择小波神经网络预测;若针对特定 GPU 芯片,可选择预测精度高的非线性回归模型.在切片功耗模型基础上充分考虑应用程序的分支逻辑,为分支稀疏的应用程序建立根据切片长度加权的功耗模型,可预测应用程序的平均计算功率;对于分支稠密的应用程序,用概率切片去掉执行概率小的代码片,提高功耗预测的准确性.实验结果表明,应用程序功耗预测模型有效,误差在可接受范围内,对各种 GPU 体系的适应性强.在研究方法上,本文在相近研究基础上做了重要改进,从切片粒度分解程序,在保证预测误差的情况下提高了预测模型的通用性.非线性回归比线性回归法提高了预测模型的稳定性;小波神经网络可根据样本数据的分布特征进行学习,与 BP 神经网络相比,具有更好的抗干扰能力.

GPU 通用计算功耗预测研究实现了普通开发人员通过扫描源代码就能了解程序的功耗分布,可判断能耗瓶颈代码段,方便开发人员进行代码重构和优化.下一步的研究难点是使用模糊技术处理程序分析中的不确定因素,提高功耗预测模型的准确性.针对分支稠密程序执行路径的概率确定问题,利用软件自动测试技术来降低人工干预成本,实现 BDPM 模型的预测精度与代价的均衡.此外,提高功耗预测模型的应用性也应引起关注,例如提取应用程序中功耗密集的代码区、可视化应用程序的能耗分布等.

致谢 感谢匿名审稿人和编辑提出的宝贵意见.

References:

[1] Kurp P. Green computing. Communications of the ACM, 2008,51(10):11−13. [doi: 10.1145/1400181.1400186]

[2] Wang XR, Chen M, Fu X. MIMI power control for high-density servers in an enclosure. IEEE Trans. on Parallel and Distributed System, 2010,21(10):1412−1426. [doi: 10.1109/TPDS.2010.31]

[3] Lin YS, Yang XJ, Tang T, Wang GB, Xu XH. A GPU low-power optimization based on parallelism analysis model. Chinese Journal of Computers, 2011,34(4):705−716 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.00705]

[4] Lin YS, Yang XJ, Tang T, Wang GB, Xu XH. An integrated energy optimization approach for CPU-GPU heterogeneous system based on critical path analysis. Chinese Journal of Computers, 2012,35(1):123−133 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2012.00123]

[5] Gebhart M, Johnson DR, Tarjan D, Keckler SW, Dally WJ, Lindholm E, Skadron K. Energy-Efficient mechanisms for managing thread context in throughput processors. Computer Architecture News, 2011,39(3):235−246. [doi: 10.1145/2024723.2000093]

[6] Wang GB, Lin YS, Yi W. Kernel fusion: An effective method for better power efficiency on multithreaded GPU. In: Proc. of the 2010 IEEE/ACM Int'l Conf. on Green Computing and Communications & Int'l Conf. on Cyber, Physical and Social Computing. 2010. 344−350. [doi: 10.1109/GreenCom-CPSCom.2010.102]

[7] Shaikh MZ, Gregoire M, Li W, Wroblewski M, Simon S. In situ power analysis of general purpose graphical processing unit. In: Proc. of the 19th Euromicro Int'l Conf. on Parallel, Distributed and Network-Based Processing. 2011. 40−44. [doi: 10.1109/PDP.2011.67]

[8] Collange S, Defour D, Tisserand A. Power consumption of GPUs from a software perspective. In: Proc. of the Computational Science (ICCS 2009). LNCS, Heidelberg: Springer-Verlag, 2009. 914−923.

[9] Jiao Y, Lin H, Balarji P, Feng W. Power and performance characterization of computational kernel on the GPU. In: Proc. of the IEEE/ACM Int'l Conf. on Green Computing and Communications & Int'l Conf. on Cyber, Physical and Social Computing. 2010. 221−228. [doi: 10.1109/GreenCom-CPSCom.2010.143]

[10] Hong S, Kim H. An integrated GPU power and performance model. Computer Architecture News, 2010,38(3):280−289. [doi: 10.1145/1816038.1815998]

[11] Lee D, Ishihara T, Muroyama M, Yasuura H, Fallah F. An energy characterization framework for software-based embedded systems. In: Proc. of the 2006 IEEE/ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia. 2006. 59−64. [doi: 10.1109/ESTMED.2006.321275]

[12] Sinha A, Ickes N, Chandrakasn AP. Instruction level and operating system profiling for energy exposed software. IEEE Trans. on Very Large Scale Integration (VLSI) Systems, 2003,11(6):1044−1057. [doi: 10.1109/TVLSI.2003.819569]

[13] Tan TK, Raghunathan A, Lakishminarayana G, Jha NK. High-Level software energy macro-modeling. In: Proc. of the 38th Design Automation Conf. 2001. 605−610. [doi: 10.1109/DAC.2001.935580]

[14] Senn E, Laurent J, Juin E, Diguet JP. Refining power consumption estimations in the component based AADL design flow. In: Proc. of the IEEE Conf. on Specification, Verification and Design Language. 2008. 173−178. [doi: 10.1109/FDL.2008.4641441]

[15] Zhang TT, Wu X, Li CD, Dong YW. On energy-consumption analysis and evaluation for component-based embedded system with CSP. Chinese Journal of Computers, 2009,32(9):1−8 (in Chinese with English abstract).

[16] Liu XB, Guo B, Shen Y, Xiong B, Wang JH, Wu YS, Liu YB. Embedded software energy modeling method at architecture level. Ruan Jian Xue Bao/Journal of Software, 2012,23(2):230−239 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4145.htm [doi: 10.3724/SP.J.1001.2012.04145]

[17] Tan TK, Raghunathan AK, Jha NK. Software architectural transformations: A new approach to low energy embedded software. In: Proc. of the Design, Automation and Test in Europe Conf. and Exhibition, Processing. 2003. 1046−1051. [doi: 10.1109/DATE.2003.1253742]

[18] Leon AS, Langley B, Shin JL. The UltraSPARC T1 processor: CMT reliability. In: Proc. of the IEEE 2006 Custom Integrated Circuits Conf. 2006. 555−562. [doi: 10.1109/CICC.2006.320989]

[19] Amtoft T. Slicing for modern program structures: A theory for eliminating irrelevant loops. Information Processing Letters, 2008, 106(2):45−51. [doi: 10.1016/j.ipl.2007.10.002]

[20] Martin PW, Hussein Z. Deriving a slicing algorithm via FermaT transformations. IEEE Trans. on Software Engineering, 2011, 37(1):24−47. [doi: 10.1109/TSE.2010.13]

[21] Pharr M. GPU Gems2. 3rd ed., Boston: Addison Wesley, 2005. 493−495.

[22] Wang HF, Chen QK. Power estimating model and analysis of general programming on GPU. JOURNAL OF SOFTWARE, 2012, 7(5):1164−1170. [doi: 10.4304/jsw.7.5.1164-1170]

[23] Bates DM, Watts DG. Nonlinear Regression Analysis and Its Applications. New York: Wiley, 1997. 36−37.

[24] NVIDIA_Corporation.CUDA c programming guide. 2012. http://www.nvidia.com/

[25] Parboil benchmark suite. 2012. http://impact.crhc.illinois.edu/parboil.php

[26] Che S, Boyer M, Meng J, Tarjan D, Sheaffer JW, Lee SH, Skadro K. Rodinia: A benchmark suite for heterogeneous computing. In: Proc. of the 2009 IEEE Int'l Symp. on Workload Characterization. 2009. 44−54. [doi: 10.1109/IISWC.2009.5306797]

[27] Zhang QH, Benveniste A. Wavelet networks. IEEE Trans. on Neural Networks, 1992,3(6):889−898. [doi: 10.1109/72.165591]

[28] Wang DW, Dou Y, Li SK. Loop kernel pipelining mapping onto coarse-grained reconfigurable architectures. Chinese Journal of Computers, 2009,32(6):1089−1098 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2009.01089]

[29] Henessy JL, Patterson DA. Computer Architecture: A Quantitative Approach. 5th ed., San Francisco: Morgan Kaufmann Publishers, 2003. 148−156.

[30] Clark D, Hunt S, Malacaria P. Quantified interference for a while language. Electronic Notes in Theoretical Computer Science, 2005,112(2):149−166. [doi: 10.1016/j.entcs.2004.01.018]

[31] Singer J. Towards probabilistic program slicing. In: Proc. of the Dagstuhl Seminar05451. Wadern: Dagstuhl Research, 2006. 1−14.

[32] Wang LL, Li BX, Zhou XY. Profiling all paths. Ruan Jian Xue Bao/Journal of Software, 2012,23(6):1413−1428 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4102.htm [doi: 10.3724/SP.J.1001.2012.04102]

[33] Moecke M, Seara R. Sorting rates in video encoding process for complexity reduction. IEEE Trans. on Circuits and Systems for Video Technology, 2010,20(1):88−101. [doi: 10.1109/TCSVT.2009.2029022]

附中文参考文献:

[3] 林一松,杨学军,唐滔,王桂彬,徐新海.一种基于并行度分析模型的 GPU 功耗优化技术.计算机学报,2011,34(4):705−716. [doi: 10.3724/SP.J.1016.2011.00705]

[4] 林一松,杨学军,唐滔,王桂彬,徐新海.一种基于关键路径分析的 CPU-GPU 异构系统综合能耗优化方法.计算机学报,2012, 35(1):123−133. [doi: 10.3724/SP.J.1016.2012.00123]

[15] 张腾腾,吴晓,李长德,董云卫.基于 CSP 的构件化嵌入式软件能耗分析与评估方法研究.计算机学报,2009,32(9):1−8.

[16] 刘啸滨,郭兵,沈艳,熊冰,王继禾,伍元胜,刘云本.嵌入式软件体系结构级能耗建模方法.软件学报,2012,23(2):230−239. http://www.jos.org.cn/1000-9825/4145.htm [doi: 10.3724/SP.J.1001.2012.04145]

[28] 王大伟,窦勇,李思昆.核心循环到粗粒度可重构体系结构的流水化映射.计算机学报,2009,32(6):1089−1098. [doi: 10.3724/SP.J.1016.2009.01089]

[32] 王璐璐,李必信,周晓宇.全路径剖析方法.软件学报,2012,23(6):1413−1428. http://www.jos.org.cn/1000-9825/4102.htm [doi: 10.3724/SP.J.1001.2012.04102]

王海峰(1976－),男,山东临沂人,博士,副教授,CCF 会员,主要研究领域为高性能计算,低功耗优化.
E-mail: gadfly7@126.com

陈庆奎(1966－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络计算,并行计算,物联网技术.
E-mail: chenqingkui@gmail.com