高技术通讯 2014 年 第 24 卷 第 8 期:791 ~ 799

doi:10.3772/j.issn.1002-0470.2014.08.003

面向低能耗的非精确异构多核上的运行时技术①

房双德②[*,**,***] 杜子东[*,**,***] 方运潭[*,**,***] 黄元杰[*,**,***] 李华伟[*,**] 陈云霁[*,**] 吴承勇③[*,**]

([*]中国科学院计算机体系结构国家重点实验室 北京 100190)

([**]中国科学院计算技术研究所 北京 100190)

([***]中国科学院大学 北京 100049)

摘 要 为降低异构多核处理器芯片的能耗,为非精确异构多核平台提出了一种基于分层调节器的硬件抽象和搜索方法。该方法首先将异构多核硬件及其非精确参数抽象为树状结构,其次使用能效分数标定调节器树,最后在线搜索其路径,为程序的每个算法获得最佳的硬件及其参数配置。实验表明,该方法能够在满足用户精确度需求的前提下,相比于精确 CPU 核,平均降低 40% 的能耗,且能够很好地适应精确度需求的变化。

关键词 非精确计算,异构多核,分层调节器,运行时,能耗减少

0 引 言

近年来为降低芯片能耗,处理器芯片逐渐从同构多核向异构多核发展[1]。即使如此,随着数据中心和移动设备等对能源的需求飞速增长,人们对降低能耗的要求仍是迫在眉睫,能耗问题已成为计算机体系结构领域关注的热点[2,3]。非精确/近似计算正是在这样的背景下得到了迅速发展,其基本想法是通过在可容忍的范围内牺牲计算的精确性来换取能耗的大幅减少。如果允许芯片中每个核(或部分核)产生非精确/近似的计算结果,则可进一步降低芯片的能耗。已有大量研究工作显示,非精确计算在很多重要领域具有巨大的应用潜力,例如图像、视频、搜索引擎、机器学习、数值计算等[2-5]。

在软件方面,研究者通过省略非关键计算(循环、任务等)的方式来换取能耗的减少[4]。硬件方面则通过屏蔽芯片中非关键计算单元和降低存储单元的电压和刷新率来减少能耗,但同时会带来一定的计算错误[6,7]。这些研究虽然相对成熟,但侧重于程序如何利用某一种特定的非精确软、硬件设计。而非精确和异构多核相结合,因其可以更多地减少能耗而逐渐受到重视,并成为一种新的研究趋势[8-11]。但是非精确异构多核平台面临新的挑战:运行时如何对异构的多个核及其非精确硬件设计进行抽象。抽象的目的是给运行时的能耗优化策略提供统一的、可搜索的空间,从而更好地发掘该复杂平台的潜力。本文提出了一种基于分层调节器的树状抽象结构,并使用离线标定调节器树和在线搜索的方法为程序在一定的精确度要求下最大限度地减少能耗,且自适应优化需求的变化。此外,为了提高该复杂平台的编程效率,本文沿用了 Ansel[12] 等提出的面向算法的编程,且为其增加了非精确编程的支持。本文搭建了一个包括 6 个异构核的非精确片上多核模拟器,并使用 5 个基准程序(每个程序 20000 个数据集)对该方法进行了评估。结果显示,该方法在满足精确度需求的前提下,相比于精确 CPU 核,可以平均减少能耗 40%,最高达到 50%。

1 非精确异构多核

针对功耗,散热和工艺等对体系结构发展的影响,工业界和学术界较为认可的一个解决方案是:异构多核,即把增加的晶体管用来实现对特定应用进行加速的低功耗、高效率的电路。异构多核结构作为低功耗的代表常被用于移动设备等领域,例如苹

① 核高基重大专项(2009ZX01028-002-003,2009ZX01029-001-003,2010ZX01036-001-002),863 计划(2012AA012202, 2012AA010901),国家自然科学基金(61003064, 61100163, 61133004, 61222204, 61221062, 61303158)和中科院先导专项(XDA06010403)资助项目。

② 男,1986 年生,博士生;研究方向:计算机体系结构;E-mail: fangshuangde@ict.ac.cn

③ 通讯作者,E-mail: cwu@ict.ac.cn

(收稿日期:2014-01-14)

果公司的 A 系列芯片。然而能耗依然是该类平台的瓶颈。图 1 展示了一个使用片上网络(network on chip,NoC)连接的异构多核结构,它是本文实验中所使用的模拟器平台,其包含了两个 CPU 核,一个图形处理单元(GPU),一个 CGRA 加速器(一种对循环做硬件展开的加速器,由多个块组成)[13],一个 H.264 解码加速器[14]和一个机器学习(machine learning, ML)加速器[10]。

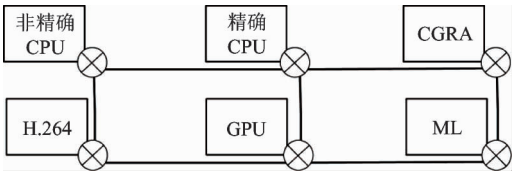

图 1 基于片上网络的异构多核平台

与此同时,另一类新兴的面向低能耗的方法——非精确计算正迅速发展并被广泛接受。本文首先将现有的多种非精确硬件设计[7,10,15]引入到异构多核平台上的每个核,从而构建一个非精确异构多核平台,并在此基础上研究运行时关键技术。

非精确硬件设计对外提供可调的硬件参数,不同的取值对计算的精确度和能耗有不同的影响。如表 1 所示,我们使用了 Lingamneni[7]等提出的方法,裁剪/屏蔽加法器和乘法器低 N 位运算。这种设计对应的两个硬件参数分别控制加法器和乘法器的位数,在 32 位算术逻辑单元(arithmetic logic unit, ALU)上,其可调范围(即 N 的取值范围)是 0 ~ 31。这种设计的原理是二进制存储中高位到低位的重要度是以 2 的指数级递减。我们将这种非精确的方式使用到了 CPU 的 ALU、GPU 的流式多处理器以及 CGRA 的每个块结构的计算单元之中。对于 ML 加速器,使用了概率逻辑最小化(probabilistic logic minimization, PLM)的方式创建其中的乘法器部件,从而影响其多层感知器的能耗和精确度[10],采用了 4 种不同的精确度设计。对 H.264 加速器,按位调节帧存储器(4 × 8 位的块)的电压来达到节省能耗的目的[15],而低电压会对存储单元的稳定性产生影响,从而对数据引入一定的错误。

表 1 核的非精确硬件设计

核	非精确设计	参数个数	取值范围
CGRA	裁剪块(tiles)中运算单元低 N 位	2	[0 ~ 31]
ML	选择 4 中不同精度的 MLP 配置	1	[0 ~ 3]
H.264	调整解码器帧存储器的电压	8	[0.4 ~ 1]
CPU	裁剪加法器和乘法器的低 N 位	2	[0 ~ 31]
GPU	裁剪流式多处理器的低 N 位	2	[0 ~ 31]

图 2 显示了在执行 JPEG 应用程序时,将其离散余弦变换(discrete cosine transform, DCT)算法运行在非精确 CPU 核上,且屏蔽其加法器和乘法器的低 26 位运算逻辑产生的结果。我们使用典型的图像质量评估方法——结构相似度指标测量(structural similarity index measurement, SSIM)[16],对得到的图像与精确的原重建图像进行比较。结果显示,仅引入了 7% 的错误,即:重建图像(左)和原重建图像(右)的 SSIM 值分别是0.93和1.0,且主观视觉质量几乎无差别,但是最终获得了 25% 的能耗节省。

图 2 非精确 CPU 对 JPEG 编码的影响

2 编程支持和运行时技术

在以往单一处理单元的结构上,程序员习惯于使用语句加上库调用的方式进行编程。然而随着硬件体系结构的复杂性、异构性和集成度日趋增加,编程抽象也亟需提升。更高的编程抽象使程序员更关注于应用程序的逻辑;而底层实现细节,以及能耗在不同执行环境和平台上的调优问题应该交给相应的运行时系统自动管理[5,12]。

2.1 面向算法的编程支持

在本研究中,采用了 Ansel[12]等提出的面向算

法的编程来提升异构多核平台上的编程抽象,并在此基础上增加了对非精确编程的支持。面向算法的编程是一种以算法(而非语句或操作)为粒度的编程方式。算法的多个版本的实现在编译时或者运行时可以进行自由替换[17]。算法的具体实现可以直接调用库或由程序员自己提供。和直接调用库不同,程序员不必关注在何种执行环境和平台下选择何种算法的实现才能达到最好的性能和能耗,运行时将自动完成这项工作,这大大减轻了其在异构多核复杂平台上进行自动能耗和性能调优的难度。

如表 2 所示,本文采用基于编译制导的方式,提供了一系列轻量级的语言制导来辅助程序员管理算法的各个实现变种和构建主程序。主要分为两大类:一类是用来组织算法及其多个版本的实现,调用算法构建应用程序。另一类是用来支持非精确计算的编译制导。为支持使用这些编译制导编写程序,本文实现了其相应的预编译器。由于编译制导的编程方式较为简单且广泛被使用(例如 OpenMP),本文就不再使用具体程序举例。

表 2　面向算法编程的编译制导

功能	编译制导
组织算法及其多个版本实现	@ ALG _ BEGIN @ ALG _ NAME @ ALG _ INTERFACE @ ALG _ VERSION:[CGRA | GPU | …] @ ALG _ END
算法调用	@ ALG _ CALL
指定精确度计算公式	@ APP _ ACCURACY _ METRIC
指定精确度需求	@ APP _ ACCURACY _ BOUND

2.2　运行时的硬件抽象和能耗调优

由于运行时在整个方法中处于编程模型和硬件之间,其主要作用是:(1)对复杂的、异构的非精确多核进行抽象,为优化策略提供统一的可搜索的优化空间;(2)进行能耗调优,即把所有的算法调度到适合的核上,并把核的硬件参数调整到最优,以保证满足用户精确度需求的情况下,使能耗降低最大化。我们按照这两个方面对运行时技术进行介绍。

(1)基于分层调节器的硬件抽象

运行时要搜索的巨大的优化空间是由两部分组合而成:为整个程序中的每个算法搜索合适的核和为所有核的硬件参数取合适的值。而这两个空间是以乘法的形式组合起来。但是异构平台上的核结构复杂,且特点各异,而需要一种方式把整个优化空间以统一的、可搜索的方式展现给运行时的搜索策略,即需要在运行时中提供一个软件抽象层。

本文采用了分级调节器的方式进行抽象。其主要思想是:加速器及其参数的选择都将被统一抽象为调节器。搜索策略所看到的一切具体的硬件及其配置都被视为调节器,所进行的一切操作都视为扭动调节器。如图 3 所示,调节器是分两级组织的,即核级调节器和参数级调节器。核级调节器用来选择使用哪个核来执行该算法。而参数级调节器则用来选择该核的非精确硬件参数的取值。例如,JPEG 编码程序将一幅图像按照 8 ×8 的块依次进行颜色空间转换(colorspace conversion, CC)、离散余弦变换(discrete cosine transform, DCT)、量化(quantization, QT)和哈夫曼编码(Huffman encoding , HE)四个步骤处理,则我们抽象出来解空间的形式为一个向量:<核调节器_ CC,核调节器_ DCT,核调节器_ QT,核调节器_ HE >。该程序中的每一个算法都对应一个核级调节器,其组织如图 3 所示,是一个分层的树形结构,称之为调节器树。而优化策略通过扭动/调节两级的调节器从调节器树中选取路径,即选取核

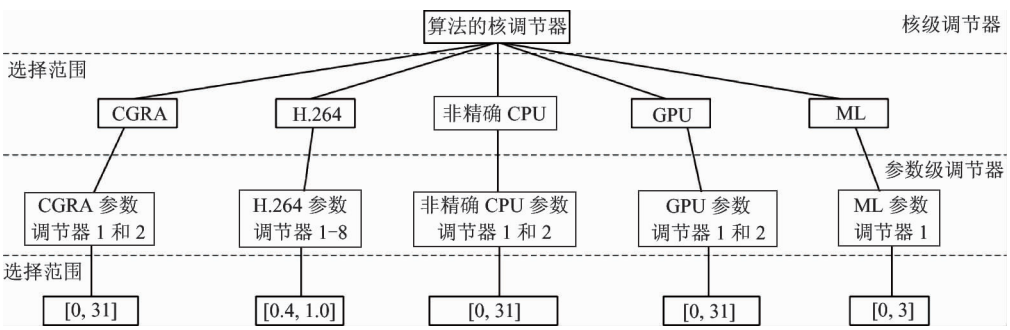

图 3　算法的分层树状调节器

及其硬件参数。此外,该树的边上经过训练后会带有一定的权重,暗示了选择该边的概率。下一节,我们将会详细讲述搜索策略如何利用分层调节器的结构。

(2)能耗调优

能耗调优的本质是在一定精确度的底线上为程序的每个算法寻找最节省能耗的解决方案,即一组核及其非精确配置参数。这是典型的组合优化难题。

调节器树的标定和搜索:由于采用了分层调节器机制,程序中的每个算法将对应一棵调节器树。运行时要解决的问题就统一为如何从核级和参数级分别选择一条路径,在满足精确度需求的前提下,最大化能耗节省。为高效地搜索,首先为每个算法标定其调节器树。其关键思想是为该算法调节器树的核级上的各个边添加权重,权重暗示着该边被选中的概率。将权重定义为能效分数(energy-efficiency score, ES),其主要衡量了一个核的两方面的能力:(1)核自身的硬件特性是否适合高效运行该算法,即核在精确状态下对该算法的能耗降低能力,定义为 P_e;(2)核的非精确设计是否对该算法的能耗影响显著,即核在非精确状态下对该算法的能耗节省的潜力,定义为 P_a。为获得 P_e 和 P_a,将在训练数据集上进行离线训练,具体步骤如下:(1)首先在训练数据集上,使用精确的 CPU 核运行该算法作为能耗的比较基准,记作 E_{baseline};(2)依次使用每个核的精确模式运行该算法获得能耗数据,记作 $E_{\text{exact\_core\_}i}$(i 是核的 ID);(3)依次在每个核的非精确,且调节器值最大的模式下,获得该算法的能耗,记作 $E_{\text{inexact\_core\_}i}$。那么,对于该算法,每个核 i,其 P_e、P_a 和 ES 分别使用以下公式(1)、(2)和(3)计算:

$$P_e = (E_{\text{baseline}} - E_{\text{exact\_core\_}i})/E_{\text{baseline}} \quad (1)$$

$$P_a = (E_{\text{baseline}} - E_{\text{inexact\_core\_}i})/E_{\text{baseline}} \quad (2)$$

$$ES = (P_e + P_a)/2 \quad (3)$$

最后将所有核的 ES 归一化,作为核级调节器的边权重。

调节器树标定完成之后,运行时的搜索算法将更关注于如何从参数级调节器选择一条路径。本文采用了模拟退火算法,解的形式如 2.2 节的部分(1)中所述,是一个由该程序各个算法的核调节器组成的向量,每个核调节器对应着一个带权重的双层树状结构。对于调节器树的参数级,该算法随机搜索当前解的邻域生成新的解决方案;对于核级,则采用带能效权重的随机方法。我们使用了经典的 Metropolis 接受准则;同时,解决方案导致的错误如果超过了用户要求的精确度底线,该解决方案也不会被接受。由于模拟退火算法较为经典,本文不再赘述其细节。

应对变化的需求:搜索过程虽然可以在离线的情况下进行[2,5],然而在实际的应用场景中,精确度、能耗和性能需求是可以改变的,原因是用户/执行环境往往需要根据情况决定是否需要更关注能耗,还是精确度/服务质量[11]。因此,我们的运行时使用在线搜索的方式,每当用户精确度或能耗的需求发生改变时,则进行重新搜索。而为了减少重新搜索的开销,在运行时的数据库中为每个程序维护了历史搜索记录,即解决方案的能耗和精确度数据。每次搜索开始之前,运行时将访问数据库,从中选择一个跟当前需求相似的曾经找到过的最好的解决方案(配置组合),并作为初始解,继续新的搜索。在线搜索的问题是开销和收益的管理,搜索过程被认为是额外的(非用户产生的)程序运行,即开销。另一方面,当一个较好的解决方案找到并应用于以后的每次程序执行时,我们获得更多的能耗和时间的节省,即搜索带来的收益。鉴于程序在计算系统中(例如数据中心)往往会运行很多次,越是重要的程序越是如此,本文借鉴 Chen 等[18]的思想,即使用基于预算的策略来管理该程序长期运行的能耗收益和开销,具体使用如下两条准则:(1)仅使用收益的一定比例来做搜索,在没有收益的情况下,仅仅使用当前整体能耗的很小的比例来触发搜索过程;(2)如果搜索过程持续失败(未找到更好的方案),则降低搜索频率,一旦成功则恢复原来的频率。

2.3 框架实现

如图 4 所示,给出了整个方法的实现框架。自左向右,用户使用 2.1 节中的方法编写的程序经过预编译后将会被分成粘合代码和算法的多版本实现代码,同时会为每个算法生成一棵调节器树。粘合代码和算法的多版本经过本地编译器(例如 GCC, NVCC 等)编译后获得二进制文件。算法的调节器树在训练数据集上进行离线标定,而后和算法的多版本一同存入数据库。程序执行时,运行时系统将通过粘合代码中的算法调用接口来查询数据库中的算法,并使用标定后的调节器树和搜索策略来选择合适的算法实现版本和相应的硬件参数,同时使用 2.2 节中的两条准则管理搜索开销和收益,最终为该程序搜索最佳的能耗解决方案。

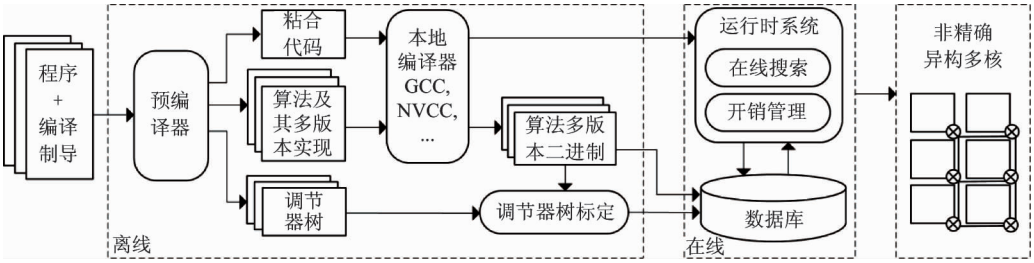

图 4　方法的框架示意图

3　实验及评价

3.1　实验环境

目前并没有商用的非精确异构多核平台,为评估本文提出的方法,本文开发了一个基于片上网络(NoC)连接的片上多核模拟器。该模拟器包含一个精确的 CPU 核(作为基准)、一个非精确 CPU 核、一个非精确 GPU 核、一个非精确 CGRA 加速器、一个非精确 H.264 解码加速器和一个非精确机器学习(ML)加速器,每个核的具体功能、配置和来源如表 3所示。

表 3　异构多核平台的配置

核	功能描述	配置	来源
CGRA	对循环硬件展开加速	300 个块 频率:400MHz,	Huang[13] 等
ML	BP 神经网络硬件加速器	尺寸:90×16×10, 频率:100MHz	Du[10] 等
H.264	硬件解压 H.264 视频	主频:1.5MHz 2.5KSRAM	Xu[14] 等
CPU	通用处理器	1.8GHz,2 核, 24K 一级缓存	模拟 Intel Atom N2800
GPU	通用图形计算处理器	频率:672MHz, 72 核	模拟 Nvidia Tegra 4

整个芯片使用 32 纳米工艺,其物理布局如图 5 所示。芯片上的存储的组织如下:两个 CPU 核共享一个 30MB 的片上 eDRAM[19],ML 加速器和 CGRA 加速器则分别有自己私有 1.5MB 的 SRAM,H.264 加速器则拥有一个 2.5kB 的 SRAM 用作帧存储器,缓存解码的视频图像。芯片上所有的核通过一个基于二维网格的 NoC 进行连接。实验中,我们使用 BookSim2.0[20] 作为 NoC 模拟器,具体配置如下:1.5GHz,128-bit 的网络位宽,路由的流水级是 4,每个路由包含 8 个虚拟通道,每个通道包含 5 个缓冲区。我们使用基于蛀洞路由的方式,维序路由算法(dimension order routing, DOR)进行数据传输,每个网络包含有 32 个网络帧。

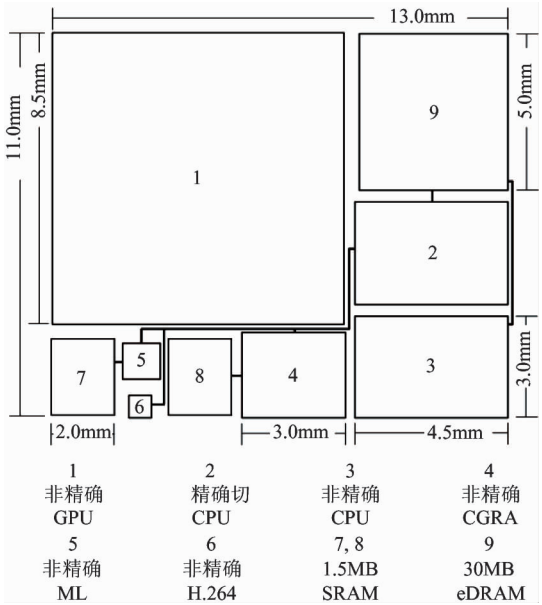

图 5　非精确异构多核芯片的物理布局

对于 H.264、CGRA、ML 加速器和在 CPU 以及 GPU 中使用的算术逻辑单元(ALU),使用 Verilog 语言实现且根据 1.1 节中提到的方法给它们引入非精确性。为更准确地评估本文的方法,使用 Synopsys Design Compiler 和 TSMC 32nm 工艺库、高电压阈值进行综合,使用 Synopsys ICC 进行布局布线。对于能耗的测量,整个芯片的能耗由各个计算核心、NoC、片上的 SRAM 和 eDRAM 存储组成。对于 CPU,根据 Sampson 等[3] 提出的能耗模型和计算单元布局布线后的结果估计其能耗,使用 CACTI5.3[21] 估计其高速缓存的能耗。对 GPU 使用 Hong 等[22] 提出的 GPU 模型估计其能耗。对于 CGRA、ML 和H.264 加速器,做了综合和布局布线来获得其平均功耗(P_{avg}),然后使用公式 $E = P_{avg} \times T$(T 为程序运行时间)来计算其能耗。对于 NoC 的能耗,使用 BookSim2.0 模拟片上网络的行为和负载,并使用 Ori-

on2.0[23]估计能耗。对于 SRAM 和 eDRAM 存储,我们分别使用 CACTI5.3 和 IBM eDRAM[19]的电器参数来估计其能耗。

如表 4 所示,采用了 5 个基准程序。这些程序来自于 Rodinia[24]基准程序集和开源社区。我们为每个程序收集或生成了 20000 个不同的数据集,每个数据集只使用一次,以模拟真实的应用场景。对于每个基准程序,关注于其内部的一些热点算法,表格第 4 列显示了这些算法所占整个程序的时间比例。表格中第 5 列显示了这些算法各自包含哪些平台的版本变种。我们将这些程序及其算法的各个版本的实现使用 2.1 节中的编程方法组织好。对于 JPEG 和 H.264 使用了 SSIM 公式计算结果的精确度,对于 STREAMCLUSTER 和 KMEANS,我们使用 BCubed 公式[25]去评估聚类的同构型和完全性作为其精确度。对于 BACKPROP 计算其输出的均方误差(mean square error, MSE)。

表 4 基准程序集

程序	数据集 (20,000/程序)	算法	时间比例 (%)	算法运行平台
BACKPROP	来自 UCI[26]	Propagation	14.2	(1)(2)(3)(4)(5)
		WeightUpdate	45.7	(1)(2)(3)(4)
STREAM-CLUSTER	来自[18]	KMedian	65.0	(1)(2)(3)(4)
JPEG	自由版权. bmp 图片	ColorConversion	10.9	(1)(2)(3)
		DCT	29.1	(1)(2)(3)(4)
		QT	19.9	(1)(2)(3)(4)
		HuffmanEncoding	37.1	(1)
KMEANS	来自[18]	DIST	38.1	(1)(2)(3)(4)
H.264	自由版权 H.264 视频	BitstreamParser	-	(1)(6)
		IntraPrediction		
		InterPrediction		
		DeblockingFilter		

注:(1)精确 CPU,(2)非精确 CPU,(3)CGRA,(4)GPU,(5)ML,(6)H.264。

3.2 实验评估

我们将分三个部分对方法进行实验:首先评估分层调节器树的参数级调节器;其次评估其核级调节器及其标定机制;最后评估该方法适应变化精确度需求的能力。

(1)参数级调节器评估

参数级调节器的主要任务是调节核的非精确硬件参数,不涉及核的选择,所以我们对每个核依次独立地进行能耗评估。实验使用表 4 中的基准程序和数据集。运行时搜索策略在规定的精确度(0.95)下,仅通过搜索程序中所有算法在同一种核上的参数级调节器组成的优化空间来实现能耗减少。由于模拟了该程序的长期执行效果,我们使用累积的能耗节省来衡量[18]。如图 6 所示,每根柱子代表了 5 个程序在该核上的平均结果。可见,运行时经过对参数级调节器的搜索,可以在每个非精确核上实现可观的能耗减少,其中 CGRA 核可以获得最高 38% 的能耗减少。整个平台可平均降低能耗 24%(以精确的 CPU 核为比较基准)。由此显示出参数级调节器的有效性以及对每个核的非精确硬件设计的有效性。同时,非精确异构多核平台相对于通用 CPU 展示了巨大节能优势。

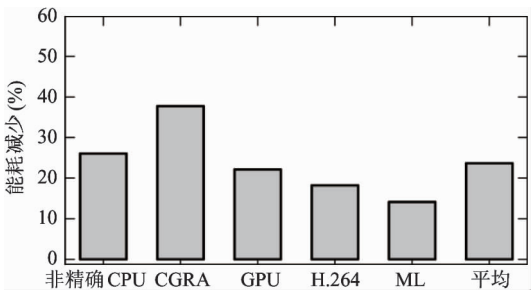

图 6 参数级调节器搜索带来的能耗减少

深入分析程序中不同算法在不同核上的结果,可以发现算法总有自己最适合的核。例如:BACKPROP 中的算法 Propagation 在 ML 核上效率最高,而在 GPU 上能耗甚至大于 CPU。这是由于该算法的

规约同步操作太多,在 GPU 上十分低效。GPU 更适合完全数据并行的算法以便利用其大量轻量级的线程和核,例如 JPEG 的 DCT 算法。CGRA 则是做硬件循环展开,由于受到芯片面积的限制,对于大循环,展开的次数不足,效率偏低,例如 DCT(223 条指令)。因此 CGRA 更适合 KMEANS 和 STREAMCLUSTER,其热循环仅含有 10 条左右的指令。由此可见,为算法选择合适的核十分必要,这正是调节器树的核级调节器要做的事。

(2)核级调节器及其标定机制评估

图 7 显示了引入核级调节器后整个方法可以获得的能耗收益。实验中,我们先使用 2.2 节中所述的方法对核级调节器进行标定,然后使用运行时搜索核级调节器为程序中每个算法选择合适的核,同时也搜索参数级调节器选择合适的硬件非精确参数。最终在牺牲 5% 的精确度的情况下,获得平均 40% 的能耗减少,最高可以达到 50%。如图 7 所示,横轴以基准程序分组,每组中最右边的柱子代表了使用双层调节器树搜索的结果,其它柱子则代表了仅在单一核上使用参数级调节器搜索的结果。可见,对所有的基准程序包含的每个算法,核级调节器都可以选择到最优的核。运行时分别为 JPEG 的 CC 和 DCT 算法选择了 CGRA 和 GPU 核,为 BACKPROP 的 Propagation 和 WeightUpdate 算法选择了 ML 核和 CGRA 核。这使得这两个程序整体的加速器和能耗减少比仅使用单一加速器的情况得到了很大的改善。而对于 STREAMCLUSTER 和 KMEANS 这两个程序,结果比使用单一 CGRA 加速器略微差,这是由于这两个程序仅适合运行在 CGRA 上,当运行时尝试其他核时,便会带来开销,然而策略很快换回到 CGRA 使得结果损失很小。即便如此,自动选择策略也是相当必要的,这是由于随着优化需求的变化,最适合的加速器也可能变化,不去固定使用一种加速器将是一种灵活性好的策略。

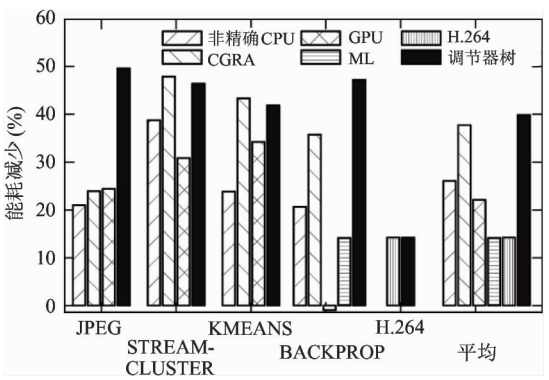

图 7　双层调节器树搜索获得能耗减少

(3)适应性评估

最后,我们在两种场景下评估了运行时对变化的精确度的适应情况。首先,把 JPEG 运行在 CGRA 上,且在第 5000 次运行时,将精确度需求从 0.95 降低为 0.85,其能耗减少的演化曲线如图 8(a)中的实线所示。作为对比,虚线则是如果一直保持精确度需求为 0.95 的情况。可见,在精确度需求调低之后,策略会逐渐适应并重新寻找更合适解决方案,从而最终得到更大的能耗减少。图 8(b)使用 BACKPROP 运行在非精确 CPU 上,展示了相反的场景,即第 10000 次执行时,提高精确度,能耗减少能力下降。此外,图中能耗曲线在程序起初执行时略微下降,这和在线开销管理策略相关,详见 2.2 节的准则(1)。

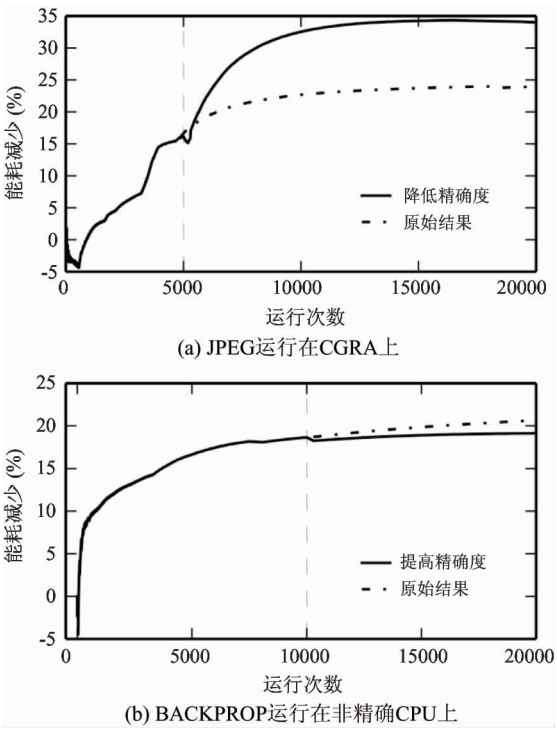

图 8　适应变化的精确度需求

4　结　论

由于能耗驱动,近年来,非精确计算和异构多核技术得到了飞速发展。两者结合,可以为程序带来更大的节能收益,同时也为平台的抽象和搜索技术带来了挑战。针对该挑战,本文提出并实现了分层调节器树的运行时技术,在该平台获得平均 40% 的

能耗减少。未来的工作主要涉及编程和硬件两个方面,在编程方面,我们将在 Li[17] 等的工作基础上为非精确算法引入算法的兼容性判定技术,完善算法库的组织和算法替换功能。在硬件方面,我们拟将新兴的神经网络加速器[8] 引入到非精确异构多核平台。

参考文献

[1] Esmaeilzadeh H, Blem E, AmantR S, et al. Dark silicon and the end of multicore Scaling. In: Proceedings of the 38th Annual International Symposium on Computer Architecture, San Jose, USA, 2011. 365-376

[2] Baek W, Chilimbi T M. Green: a framework for supporting energy-conscious programming using controlled approximation. In: Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation, Toronto, Canada, 2010. 198-209

[3] Sampson A, Dietl W, Fortuna E, et al. Approximate data types for safe and general low-power computation. In: Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, San Jose, USA, 2011. 164-174

[4] Agarwal A, Rinard M, Sidiroglou S, et al. Using code perforation to improve performance, reduce energy consumption, and respond to failures. MIT-CSAIL-TR-2009-042. Cambridge, MA, USA: MIT, 2009

[5] Ansel J, Wong Y L, Chan C, et al. Language and compiler support for auto-tuning variable-accuracy algorithms. In: Proceedings of the 9th Annual IEEE/ACM International Symposium on Code Generation and Optimization, Chamonix, France, 2011. 85-96

[6] Liu S, Pattabiraman K, Moscibroda T, et al. Flikker: Saving refresh-power in mobile devices through critical data partitioning. In: Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems, Newport Beach, USA, 2011. 213-224

[7] Lingamneni A, Enz C, Nagel J L, et al. Energy parsimonious circuit design through probabilistic pruning. In: Design, Automation & Test in Europe Conference & Exhibition, Grenoble, France, 2011. 1-6

[8] Chen T, Du Z, Sun N, et al. A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning. In: Proceedings of the 19th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Salt Lake City, USA, 2014

[9] Esmaeilzadeh H, Sampson A, Ceze L, et al. Towards Neural Acceleration for General-Purpose Approximate Computing. In: Proceedings of the 45th Annual IEEE/ACM International Symposium onMicroarchitecture, Vancouver, Canada, 2012. 449-460

[10] Du Z, Lingamneni A, Chen Y, et al. Leveraging the Error Resilience of Machine-Learning Applications for Designing Highly Energy Efficient Accelerators, In: Proceedings of the 19th Asia and South Pacific Design Automation Conference, Singapore, 2013. 201-206

[11] Sorber J, Kostadinov A, Garber M, et al. Eon: A language and runtime system for perpetual systems. In: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems, Sydney, Australia, 2007. 161-174

[12] Ansel J, Chan C, WongY L, et al. A language and compiler for algorithmic choice. In: Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation, Dublin, Ireland, 2009. 38-49

[13] Huang Y, Ienne P, Temam O, et al. Elastic CGRAs. In: InternationalSymposium on Field-Programmable Gate Arrays, Monterey, USA, 2013. 171-180

[14] Xu K. Power-efficient Design Methodology for Video Decoding:[Ph. D. dissertation]. Hong Kong: The Chinese University of Hong Kong, Electronic Engineering, 2007. 1-275

[15] Li X. Rethinking memory redundancy: optimal bit cell repair for maximuminformation storage. In: Proceedings of the 48thACM/EDAC/IEEE Design Automation Conference, San Diego, USA, 2011. 316-321

[16] Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004, 13(4): 600-612

[17] 李恒杰,何文婷,陈莉等. 支持算法组件自动替换的编程范式及编译框架. 高技术通讯,2013,23(11):1131-1138

[18] Chen Y, Fang S, Eeckhout L, et al. Iterative optimization for the data center. In: Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, UK, 2012. 49-60

[19] Wang G, Anand D, Butt N, et al. Scaling deep trench based eDRAM on SOI to 32nm and Beyond. In: IEEE InternationalElectron Devices Meeting, Baltimore, USA, 2009. 1-4

[20] Jiang N, Becker D U, Michelogiannakis G, et al. A detailed and flexible cycle-accurate Network-on-Chip simulator. In: IEEE International Symposium on Performance

Analysis of Systems and Software, Austin, USA, 2013. 86-96

[21] Thoziyoor S, Muralimanohar N, Ahn J H, et al. CACTI5.3. http://hpl.hp.com/research/cacti: HP, 2008

[22] Hong S, Kim H. An integrated GPU power and performance model. In: Proceedings of the 37th Annual International Symposium on Computer Architecture, Saint-Malo, France, 2010. 280-289

[23] Kahng A B, Li B, Peh L S, et al. A fast and accurate NoC power and area model for early-stage design space exploration. In: Proceedings of the Conference on Design, Automation and Test in Europe, Nice, France, 2009. 423 - 428

[24] Che S, Boyer M, Meng J, et al. Rodinia: A Benchmark Suite for Heterogeneous Computing. In: IEEE International Symposium onWorkload Characterization, Austin, USA, 2009. 44-54

[25] Amig'o E, Gonzalo J, Artiles J, et al. A comparison of extrinsicclustering evaluation metrics based on formal constraints. *Information Retrieval*, 2009, 12(4): 461-486

[26] Aha D, Murphy P, Merz C, et al. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/: 2007

Run-time technology for low-power oriented inexact heterogeneous multi-core

Fang Shuangde[*,**,***], Du Zidong[*,**,***], Fang Yuntan[*,**,***], Huang Yuanjie[*,**,***],
Li Huawei[*,**], Chen Yunji[*,**], Wu Chengyong[*,**]

([*] State Key Laboratory of Computer Architecture, Chinese Academy of Sciences, Beijing 100190)

([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([***] University of Chinese Academy of Sciences, Beijing 100049)

Abstract

To reduce the energy consumption of a heterogeneous multi-core processor, a hierarchical-regulator based hardware abstraction and search approach is proposed for the inexact heterogeneous multi-core architecture. The approach organizes heterogeneous cores and inexact parameters into an abstract regulator tree, then calibrates the tree with energy efficiency scores, and finally, designs an online tree search algorithm to find the optimal core combination and inexact parameter settings for each algorithm of a program. The experimental results show that compared to running the programs directly on an exact CPU, this approach can achieve an average energy reduction of 40%, without violating user-defined accuracy constraints. It is demonstrated that the approach is adapted to the changing of accuracy constraints.

Key words: inexact computing, heterogeneous multicore, hierarchicalregulator, run-time, energy reduction