异构众核系统及其编程模型与性能优化技术研究综述

巨 涛,朱正东,董小社
(西安交通大学电子与信息工程学院,陕西西安 710049)

摘 要: 异构众核系统已成为当前高性能计算领域重要的发展趋势.针对异构众核系统,从架构、编程、所支持的应用三方面分析对比当前不同异构系统的特点,揭示了异构系统的发展趋势及异构系统相对于传统多核并行系统的优势;然后从编程模型和性能优化方面分析了异构系统存在的问题和面临的挑战,以及国内外研究现状,结合当前研究存在的问题和难点,探讨了该领域进一步深入的研究方向;同时对两种典型的异构众核系统 CPU + GPU 和 CPU + MIC 进行不同应用类型的 Benchmark 测试,验证了两种异构系统不同的应用特点,为用户选择具体异构系统提供参考,在此基础上提出将两种众核处理器(GPU 和 MIC)结合在一个计算节点内构成新型混合异构系统;该新型混合异构系统可以利用两种众核处理器不同的处理优势,协同处理具有不同应用特点的复杂应用,同时分析了在该混合异构系统下必须要研究和解决的关键问题;最后对异构众核系统面临的挑战和进一步的研究方向进行了总结和展望.

关键词: 异构众核系统; 高性能计算; 异构计算; 编程模型; 性能优化
中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112 (2015)01-0111-09
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.01.018

The Feature, Programming Model and Performance Optimization Strategy of Heterogeneous Many-Core System: A Review

JU Tao, ZHU Zheng-dong, DONG Xiao-she
(*School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi* 710049, *China*)

Abstract: The heterogeneous many-core system has emerged as a promising developing trend in the high performance computing area. In this paper, we first revealed the developing trend and dominant position of the heterogeneous systems via analyzing their architectures, programming and application characteristics. Secondly, we discussed the programming model and performance optimization of current heterogeneous systems, and summarized the relative research trends. Thirdly, we verified the different application behaviors of the GPU and MIC heterogeneous system by running different types of Benchmark, which provides the reference for user to select the specific heterogeneous computing platform and, the basis of composing the hybrid heterogeneous system which combines the two types of many-core processor(GPU and MIC) into an individual computing node. This hybrid heterogeneous system can fully exploit the computing potential of the two types of many-core coprocessors to coordinate handling the complex application with different application behaviors. Finally, some open issues and future directions in the heterogeneous system were viewed.

Key words: heterogeneous many-core system; high-performance computing; heterogeneous computing; programming model; performance optimization

1 引言

为了达到高性能低功耗的目的,新型异构众核系统通常采用主处理器 + 协处理器的体系架构,该架构下主处理器(CPU)负责处理复杂的逻辑控制任务,协处理器负责处理计算密度高、逻辑分支简单的大规模数据并行任务[1],两者协同为具体的应用提供高效的计算平台.

2012 年 11 月发布的 TOP500 榜单中[2],有 62 套系统使用了主处理器 + 协处理器(加速器)混合架构,其中位居榜首的 Titan 和排名第八的我国 Tianhe-1A 都采用了 NVIDIA GPU 作为加速计算部件.而上届榜单中只有 58 套系统使用了主处理器 + 协处理器(加速器)架构,这一变化充分说明在高性能计算领域异构系统发展迅速.首次进入该榜单 TOP10,排名第七的 Dell 公司的

收稿日期:2013-09-18;修回日期:2014-03-10;责任编辑:李勇锋
基金项目:国家自然科学基金(No.61173039,No.61202041);国家 863 高技术研究发展计划(No.2012AA010904,No.2012AA01A306);国家科技支撑计划(No.2011BAH04B03)

Stampede 系统使用了最新的 Intel MIC Knights Corner 协处理器,同时 TOP500 整个榜单中还有其它 6 个系统也同样采用了最新的 Intel MIC Knights Corner 协处理器.而仅仅六个月前的上次 TOP500 排名中,只有一套系统使用了 Intel MIC Knights Corner 协处理器,且位列 TOP500 第 150 位.2013 年 6 月 17 日发布的最新 TOP500 榜单中,我国 Tianhe-2 以每秒 33.86 千万亿次的浮点运算速度荣登 TOP500 榜首,其采用的也是主处理器 + 协处理器的混合架构,且使用了最新的 Intel MIC Knights Corner 协处理器.这一变化进一步说明新型异构系统在高性能超级计算机领域的地位越来越重要.TOP500 中系统架构的整体发展趋势,反映出异构系统将是未来高性能计算机系统重要的发展方向.随着新的协处理器技术的出现,这种混合架构将会在后续的万万亿次、百亿亿次超级计算机系统中发挥更重要的作用.异构计算也被业界视为继单核、多核之后的第三个时代,它将打破摩尔定律,有效解决能耗、可扩展性等问题,成为高性能计算领域的一种重要新兴模式[1,3,4].

针对当前高性能计算领域四种典型的异构系统架构 CELL、CPU + GPU、APU 和 CPU + MIC,具体分析了各种架构的特点及其应用范围;对两种典型的片间异构系统 CPU + GPU 和 CPU + MIC 的编程特点进行了分析对比;重点讨论了异构系统编程模型和性能优化方面存在的问题及国内外的研究现状,总结了相关的研究趋势和解决相应问题的研究思路;利用不同的 Benchmark 测试程序对两种典型异构系统(CPU + GPU 和 CPU + MIC)进行测试,依据测试结果,分析它们处理应用时的不同特点,总结了它们不同的应用范围;提出了将两种异构架构结合形成一种混合异构系统的思路,以更好的发挥两种架构的各自优势来处理更复杂的大规模计算问题,同时分析了在该混合架构下必须要研究和解决的关键问题.

2 异构系统架构特点

当前异构系统主要有以下几种架构:Cell/B.E、CPU + GPU、APU、CPU + MIC 及 CPU + FPGA.其中 FPGA 主要用在嵌入式系统中的高性能需求中,而传统意义上的高性能领域使用的异构系统架构主要有 Cell/B.E、CPU + GPU、APU 和 CPU + MIC.本文主要针对传统意义上的节点级异构,故对 FPGA 不作详细讨论,对其详细的论述可参考文献 [1].

2.1 Cell/B.E.异构系统架构

Cell/B.E.(Cell Broadband Engine)是依据 CBEA(Cell Broadband Engine Architecture)技术所实现的处理器,即通常所说的 Cell 处理器.就技术而言,Cell/B.E.是以原有 PowerPC 架构为基础,再针对娱乐用 3D 绘图运算的需求添加协处理单元设计而成.Cell 处理器架构如图 1 所示[1].Cell 处理器是一种完全独立的异构处理器,使用成本较高,现在只用在一些高端服务器上[5].由于其程序开发难度较大,限制了 Cell 处理器在通用高性能领域的普及,同时也制约了它的进一步发展,它在异构系统中的优势逐渐衰退.

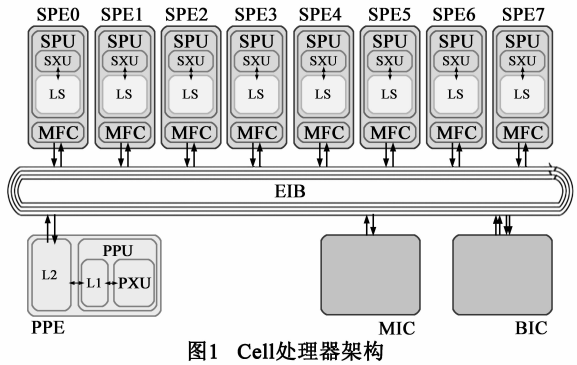

图1 Cell处理器架构

2.2 GPU 异构系统架构

GPU 异构系统主要采用 CPU + GPU 架构.GPU 的主要设计目标是以大量的线程实现面向高吞吐量的数据并行计算,适合于处理计算密度高、逻辑分支简单的大规模数据并行任务.而 CPU 具有复杂的逻辑控制单元和大容量的缓存,有低的数据传送延迟,能够适应各种不同类型的运算情况,尤其擅长复杂的逻辑运算.采用 CPU + GPU 架构,将 CPU 和 GPU 各自优点结合起来,让 GPU 处理数据密集型的并行任务,而由 CPU 进行复杂的逻辑事务处理,从而充分发挥 CPU 和 GPU 各自优势,最大限度的利用异构系统的处理能力,降低计算成本和能耗.图 2 为 GPU 异构系统架构[6].

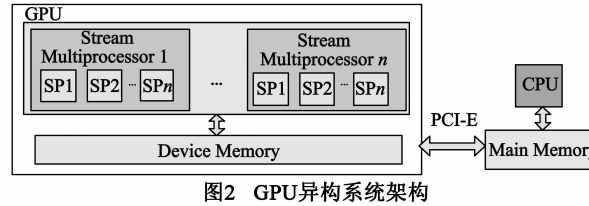

图2 GPU异构系统架构

2.3 APU 异构系统架构

APU(Accelerated Processing Unit)是 AMD 推出的整合了 x86/x64 CPU 处理核心和 GPU 处理核心的新型“融聚”(Fusion)处理器.具体架构如图 3 所示[7].APU 的主要特点是它内含由标量和矢量硬件构成的全部处理能力,由于将两种计算核心整合在一起,受芯片空间和制造工艺及散热的制约,处理器的主频及处理核数目受到限制,同时当前 APU 中 X86 CPU 和 GPU 矢量处理器还没有实现无缝集成[8],所以其整体的计算性能与主处理器和协处理器分离的架构相比稍有不足.

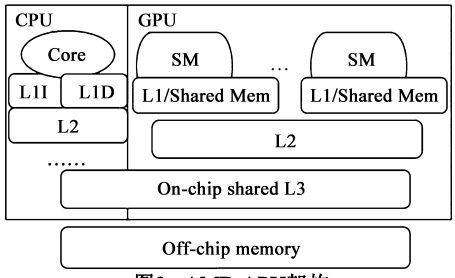

图3　AMD APU架构

2.4　MIC 异构系统架构

MIC(Many Integrated Core),是 Intel 2011 年推出的新一代集成众核协处理器,它通过对传统的微处理器进行向量扩展,然后将多个扩展后的核心整合在一起,来进一步提升计算能力.MIC 协处理器的独特之处在于:它不像传统的加速器,它本身具有独立的微操作系统 μos,所以更像是一个能被访问、被编程、功能全面的高性能计算节点,在应用程序看来,它就如同一个运行着自身的基于 Linux 的操作系统(与主操作系统无关)的计算机.图 4 为 MIC 异构系统架构[9,10].

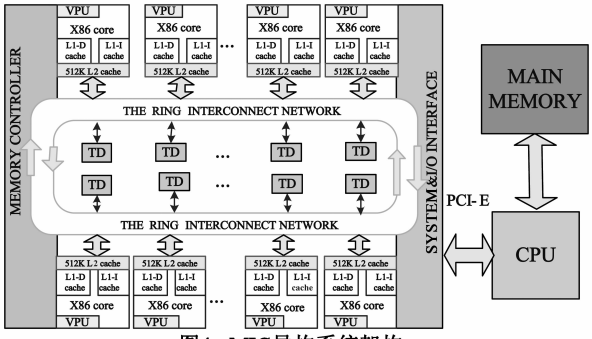

图4　MIC异构系统架构

由以上分析可知,Cell 和 APU 是一种片内的异构系统,而 GPU 和 MIC 则是要和主处理器(CPU)配合以对应用进行加速的片间异构系统.根据目前的发展趋势,GPU 和 MIC 这种以加速卡的运行方式更具通用性,同时这种架构的使用成本相对较低.因为采用 GPU 和 MIC 加速不需要从物理上对已有的硬件系统进行大的改动,只需要将 GPU 卡和 MIC 卡通过 PCI-E 接口和已有的通用处理器相连,就可以构成异构系统,可以在保留原有硬件投资的基础上,容易的以较低的成本实现原有计算系统向异构系统的扩展,为普通高性能计算用户提供一种经济、可行、实际有效的计算、仿真和验证平台.

3　异构系统编程特点

本节针对通用高性能计算领域,以易扩展和有效实用性为基础,重点对目前通用的 GPU 和 MIC 两种典型的节点级异构系统从编程特点方面作进一步讨论.

3.1　GPU 编程特点

在 GPU 异构系统上最有代表性的编程模型是 NVIDIA 的 CUDA(Compute Unified Device Architecture),也是当今最流行的 GPGPU(General Purpose Computing on Graphics Processing Unit)语言.与以往 GPU 编程语言不同的是,CUDA 不再面向图形计算,它提供编译器和开发工具,让开发人员用一种类 C 的编程语言来开发 GPGPU 程序,并同时运行于 CPU 和 GPU 上[6].CUDA 为 GPU 通用计算提供了统一的编程接口,提供近似高级语言的语法特性,突破了以往 GPU 编程时必须要深入了解底层硬件和图形处理接口限制,简化了 GPU 编程难度.但 CUDA 并不是一种完全透明的语言,它没有封装存储系统的异构性,在进行 CUDA 程序开发时,程序员还是需要对各种存储器的特点和局限性有清楚的认识,要根据硬件特点将任务进行合理分解,进行具体的线程配置,并在编程时对数据传输、存储器访问及指令流进行优化,从根本上增加了编程难度和程序调试难度,且容易出错.从而影响了 GPGPU 程序的开发效率,阻碍了 GPGPU 向更广泛的具体应用领域的普及.

3.2　MIC 编程特点

MIC 架构的优点之一是可运行已有的 x86 应用程序,而不用将程序完全移植到一个新的编程环境中,可以比较方便的利用协处理器的高计算性能.MIC 架构处理器能够支持现有的标准化编程工具,大大方便了开发人员.MIC 架构在单个芯片中融合了众多处理核心,这些核心都能够通过使用标准的 C、C + + 和 FROTRAN 进行编程.MIC 卡拥有灵活的编程方式,即可以作为一个协处理器存在,又可以被看作是一个独立的节点.较常用的编程模式主要有以下三种:(1)以 CPU 为主 MIC 为辅的编程模式;(2)CPU 和 MIC 对等的编程模式;(3)MIC 原生编程模式[11].

4　异构系统编程模型和性能优化研究

采用主处理器 + 协处理器异构架构,可以使异构系统从整体上达到高性能和低功耗的目的,从而为具体的应用提供高效的计算平台.Owens J D 等[12]对 GPU 通用计算相关的硬件及软件、编程模型、编程语言及相应的编译调试环境、性能优化技术及相关的 GPGPU 应用作了综述.Brodtdorb A R 等[5]对 GPU 硬件架构和传统的优化技术进行了综述.王海峰等[13]从应用的角度对 GPU 体系架构、编程模型、开发语言、存储模型、通信模型、负载均衡、可靠性、低功耗优化等方面进行了综述,同时对 GPU 当前的应用领域及进展进行了分析,最后总结了 GPGPU 研究中存在的问题及面临的挑战.

目前针对异构众核系统硬件架构缺乏较好的统一编程模型,没有提供一种有效的完全透明的编程语言,

现有的编程模型没能较好的封装底层硬件架构的异构性.用户开发异构系统程序的难度较大,同时由于缺乏较好的性能优化机制,异构系统的计算资源得不到充分利用,不能充分发挥异构众核系统高效计算能力.基于以上异构众核系统存在的问题,学术界对异构系统下如何降低编程难度,如何提高编程效率及性能优化方面做了大量的研究工作.

4.1 异构系统编程模型研究现状分析

并行编程模型是底层体系结构与上层应用程序之间的桥梁,向上隐藏并行处理器的细节,提供给程序员并行表达的方法;向下充分利用硬件资源、高效正确地实现应用[14].并行编程框架是对某一类问题解决方案的最高抽象形式,它由一套应用程序接口 API 和系统运行时函数组成.用户需要完成的工作是根据具体的应用问题和该框架的并行处理方式,设置接口函数的参数值,调用相应的接口函数.接口函数根据用户指定的参数值,对操作数据集进行划分,然后调用相应的任务处理函数和系统运行时函数自动完成问题的并行求解.异构系统下同样是按编程模型或编程框架的方式来解决可编程性问题.

当前异构系统编程模型主要的一种方法是通过对已有的语言进行扩展,提供一种高级的异构多核编程抽象机制,让用户利用较通用的并行编程语言或更接近高级语言的简单语言编写具体的应用程序,然后通过特定的编译器生成异构系统底层的目标代码.用户在开发应用程序时不用考虑异构系统具体的硬件特点,主要从算法层面考虑应用程序的开发,而具体的任务分配、线程配置、数据传输、存储访问、指令执行的优化工作由编译器来完成,相应的编译器将用户代码自动转换成适合异构系统特点的目标代码,在异构系统上运行.

这方面研究大多针对 CPU + GPU 异构众核架构,具有代表性的工作主要有:

Lee S 等[15]提出了一种将通用并行编程语言 OpenMP 编写的标准应用程序转换成 GPGPU 代码的自动编译和优化框架.该编程框架的主要思路是让用户利用已有的通用并行编程模型来编写基于异构系统的并行程序,而由源到源编译器来完成具体的代码转换、性能优化及代码到具体硬件的映射工作,从而提高异构系统的可编程性.利用编译器来屏蔽异构系统底层的编程和存储访问的实现细节,代价是系统性能受到限制,要进一步提升性能需要额外手动的对编译器产生的底层代码进行调节,所以整个系统的扩展性,编程灵活性,系统性能都会受到影响.文献[16] OpenMPC 编程框架是在扩展文献[15]工作的基础上,通过增加指导语句和环境变量及优化策略的方式,实现更细粒度的 OpenMP 到 CUDA 的转换及性能优化.

Han T D 等[17]提出了一种基于指导语句的高级语言 hiCUDA,它利用源到源编译技术为用户提供了一种用简单的方式完成一般的应用程序到 CUDA 程序的移植.该类研究通常只针对特定的异构系统进行编译指导支持,如果底层硬件架构发生变化,就要重新设计编译和指导机制,系统的可扩展性有限.研究能够提供一套标准的指导语句规范,根据异构系统不同的底层架构,使用不同的编译选项进行不同的编译,以适应异构系统底层硬件架构不断发展变化的要求,这是解决系统可扩展性一个重要的研究方向.

Baskaran M M 等[18]实现一种通用串行 C 程序到 CUDA 程序的自动转换系统.该方法与前两种方法相比,对用户来说可编程性更好,是一种比较理想的解决异构系统可编程性的思路,用户直接用以前熟悉的串行编程模型进行程序开发,而由编译器将通用的串行程序直接转换成满足异构系统的并行代码,整个过程中无需用户参与,但这种方法只针对特定的硬件架构,程序的并行性受系统编译和自动并行转换技术的限制,系统的性能和可扩展性受到制约.

Linderman M D 等[19]提出的 Merge 框架,为异构多核平台提供了并行编程模型、编译器和运行时支持;Dubach C 等[20]专门针对 GPU 的异构系统,设计了一种兼容 JAVA 的面向异构系统的高级编程语言 Lime,这也是一种通过语言扩展的方式来提高异构系统可编程性的策略;Liu W 等[21]提出了能够在 CPU 和加速器之间平衡计算任务,充分利用两类处理器计算能力的异构系统编程模型;Gelado I 等[22]提出了一种非对称的分布式共享存储的异构计算编程模型,该编程模型通过轻量级方式实现虚拟的共享存储,从而可以克服对称分布式共享存储的不足.虚拟共享存储是异构计算的一个重要研究方向.

以上通过设计编程模型或编程框架来解决异构系统可编程性问题的主要特点是:侧重于如何降低异构系统的编程复杂性,用户在编程时主要从算法层面考虑,不用考虑具体的底层硬件架构特点;具体性能优化的工作由编译器完成,对程序员透明.

总体上,提高异构系统的可编程性可以从以下方面进行深入的研究:第一,将原有的通用并行编程模型编写的并行程序通过源到源编译技术转换成满足特定异构系统要求的目标代码;第二,通过扩展现有的通用编程语言,在原有的程序中加入一些指导语句,结合编译技术,实现通用程序到异构系统目标代码的转换;第三,提供一种和异构系统底层硬件架构完全独立的共享虚拟存储的编程模型和语言,结合编译器和相应的运行时来提高异构系统的可编程性,可扩展性及性能;

第四,实现完全自动的将通用串行程序转换成满足异构系统要求的目标程序.

4.2 异构系统性能优化研究现状分析

由于异构众核系统和传统的多核系统在硬件架构上有很大的差异,在具体应用时,用户要对异构系统计算资源进行显式管理,如果缺乏较好的性能优化机制,将导致异构系统的计算资源得不到充分有效的利用.研究已表明优化和未优化的应用程序性能可能相差数十倍或更大[23~25].Ryoo S 等[23]分析了在 GPU 异构平台上进行性能优化时所要考虑的优化问题,总结了具体的优化原理,同时对具体的优化策略进行了测试验证,为异构系统性能优化提供了参考.目前针对异构系统性能优化主要通过设计编译器、提供优化工具及函数库、设计运行时系统三种方式实现:

(1)通过设计编译器进行性能优化

Baskaran M M 等[24]设计了一个对具有规则循环嵌套的 GPGPU 程序进行优化的编译器框架,利用具有抽象数据依赖分析和程序转换特点的多面体模型编译器对全局和共享存储访问进行优化,通过实验探测的方法确定任务分割及循环展开参数,使程序在具体执行时线程数和相应的计算资源能够合理的匹配,从而保证异构系统的计算资源得到有效的利用,提高系统的性能.Jang B 等[25]针对异构系统存储子系统具有的异构性和分布式的特点,建立能反映循环嵌套中存储访问模式的数学模型,使用数学的方法对程序的存储访问特点进行建模,具有较强的抽象性和通用性.Sundaram N 等[26]提出了一个在 GPU 上运行专用应用程序的软件框架,该框架主要考虑了 GPU 编程中的两个关键问题,一是所要处理应用程序数据远远超过 GPU 存储容量限制的程序运行问题;二是最小化 CPU 和 GPU 之间数据传输问题,这为异构系统下,优化和处理大规模专用应用程序提供了借鉴.He B 等[27]在 GPU 异构平台上实现了 MapReduce 框架.Liu Y 等[28]通过探测程序输入参数对 GPU 程序优化的影响,提出了一个基于编译器的自适应优化框架 G-ADAPT.

以上通过设计编译器的方法对异构系统进行性能优化,主要使用静态的方法,结合一些抽象的数学变换来抽取出程序中的并行部分,分析各并行部分的依赖关系,从任务划分、存储访问、数据传输、负载均衡和向量化等方面进行优化.但这些优化大多只针对具有规则并行特征的应用程序,或对某一类专用的应用程序进行优化,整个优化的通用性有限,要使通用应用程序在异构系统上获得理想的性能,目前仍然需要相应的手动优化.

(2)通过优化工具和函数库进行性能优化

通过提供优化工具和函数库的方式帮助程序员对编写好的满足具体异构系统要求的初始并行应用程序进行优化.程序员在编写具体应用程序时不用过多的考虑底层硬件特点和具体的优化,只要编写出基本的满足异构系统的代码,利用所提供的优化工具对源程序进行检测和分析,找出程序中影响性能的瓶颈部分,然后调用相应的函数库从任务分解、线程配置、存储访问和指令执行方面按特定的策略进行优化,从而充分利用异构系统的计算资源,提高应用程序的计算性能.同时针对具体的异构系统,将一些科学计算中常用的典型数据密集型应用经优化和并行化后封装成完整的核心函数库,用户在开发具体程序时直接通过 API 调用对应的函数库就可以充分利用异构处理器来提升整个应用程序的计算性能.如 NVIDIA 的 CUBLAS 数学库[6]和 Intel MIC MKL 核心函数库[11]等.

Lee J 等[29]提出了一种多线程感知的预取机制来优化 GPGPU 性能.Y Yang 等[30]提出了一种 GPGPU 新的优化编译器框架.同时针对 GPU 异构系统,为了充分挖掘系统的处理能力,最大的发挥主处理器和协处理器的计算性能,文献[7]从 CPU 和 GPU 如何更好的协作,文献[31]从如何通过共享存储复用技术来提高可同时运行在 GPU 上的线程数方面对系统的整体性能优化进行了研究.文献[32]提出了一种面向 GPU 通信与计算重叠(overlap)的数据划分方法.文献[33]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、没有一个全局优化策略的问题,提出了一种基于剖分的优化策略库、剖分工具库和策略配置模块组成全局性能优化方法,通过三个模块的相互配合以渐近的方式完成应用的全局优化过程.

采用优化工具和函数库的优化机制大多数只侧重某一方面的性能优化问题,或针对某种具体应用进行优化,多数情况是用手工编码的方法实现优化,如何将各种优化技术整合,实现动态自适应优化,从整体上提高计算性能仍存在一定的困难.

(3)利用运行时支持技术实现性能优化

通过为异构多核平台提供运行时支持实现应用程序的自动优化和软件到硬件的映射,达到降低异构多核系统编程难度、提高性能的目的.

Wang P H 等[34]提出了一种将异构加速器表示成基于 ISA 的 MIMD 应用级计算资源的体系架构 EXOCHI;Luk C K 等[35]提出了一种自动的将计算任务映射到异构多核处理器运算单元上的自适应映射(Adaptive mapping)技术;Jablin T B 等[36]提出了一种新的完全自动的 CPU-GPU 通信运行时管理系统.

以上针对异构多核架构,从如何提高异构系统的通用性,如何降低编程难度及性能优化方面所做的工作仍然要依赖于底层具体的硬件架构和对应的软件支

持,如果底层硬件架构一旦发生变化,就要根据具体硬件架构特点重新设计整个编译及运行时系统、性能优化库和工具.所以系统的可扩展性和软件可移植性不能较好的适应新型异构系统的快速发展趋势.

4.3 进一步的研究方向

由于目前缺乏一个标准化编程环境来统管异构系统内呈多样化发展态势的各种资源,异构系统的发展和普及受到了限制.在提高异构系统的通用性方面,OpenCL 创造了一个独立于硬件的软件开发环境[37],它支持不同层次的并行,并能高效映射到由 CPU、GPU、FPGA 和其他潜在的未来设备组成的单一或多设备的同构或异构系统.但与 CUDA 类似,OpenCL 提供了一种底层语言抽象层接口,使用者需要对整个设备的硬件架构有相当的了解,用户编程难度依然很大,业界仍需要一种能够有效支持跨平台且低学习门槛的统一编程规范的出现.OpenACC 是最新出现的一种跨平台基于指令的隐式的编程模式[38],致力于为不同的异构硬件平台提供统一的编程环境.与 OpenCL 相比,OpenACC 保持了良好的跨平台性并且拥有更高层次的抽象.目前已经有多家厂商提供设备和编译器支持[39~41].

通过以上分析可知,为了适应快速发展的异构系统硬件架构的要求,通过设计与异构系统底层硬件架构和对应的支持软件完全独立的编程框架来解决异构系统的可编程性、可扩展性和软件可移植性问题是一种发展趋势.该框架下,用户只从程序算法层面进行并行程序开发,而不用考虑具体异构系统底层硬件特点.编程框架会根据应用程序特点将程序转换成通用的中间代码,然后再由相应的运行时系统负责将中间代码映射到具体的异构系统上运行.而运行时系统在实现具体映射时,根据在操作系统中设置的系统信息感知模块获取必要的硬件信息来完成任务到硬件上的映射执行.当底层硬件架构发生变化时,对整个编程框架来说,不用重新设计运行时系统和前端的通用程序到中间代码的转换部分,只根据具体的硬件架构特性改变或增加相应的运行时库,就可以完成软件到新的异构系统的移植,保证整个编程框架具有较好的可扩展性,以适应不同的异构系统.

5 GPU 和 MIC 两种异构系统性能测试

测试对比两种异构系统 CPU + GPU 和 CPU + MIC 性能差异,分析引起性能差异的原因,得出两种异构系统处理不同类型应用的特点,指出两种异构系统各自擅长的应用,从而为用户选择具体的异构计算平台提供参考.

5.1 测试环境及测试程序

测试平台分别采用了实验室配置的浪潮 TS10K 集群中的两个独立的 MIC 计算节点和 GPU 计算节点.两个计算节点配置如下:

MIC 节点有两块 Xeon Phi 7110P MIC 卡.GPU 节点有两块 Nvidia 的 kepler 架构的 k20m GPU 卡.MIC 节点和 GPU 节点都配备了两颗 E5-2670 CPU,主存大小 64G,CPU 主存和加速卡之间采用通道带宽为 x16 的 PCI-E 总线进行数据传输.操作系统均为 RedHat Linux 5 企业版,MIC 节点安装 Intel parallel-studio-xe-2013-update3-intel64 软件开发环境,GPU 节点安装了 CUDA SDK 5.0 软件开发包.所采用的测试程序及其特点如表 1 所示.

表 1 测试程序特点及其在不同平台上的运行时间(时间:秒)

测试程序	CPU	CPU + GPU	CPU + MIC	主要特征	处理方式	应用领域
SPGEMM[43,44]	20.6374	4.8178	8.0779	计算受限,规则应用	数据并行	稀疏线性代数
Monte Carlo[45]	14.203	0.93992	0.4416	计算受限,规则应用	任务并行	物理模拟,金融计算
Histogram[44]	44.6723	4.9269	15.3257	带宽受限,非规则应用	数据并行	图像处理
FFT[46]	129.38	52.62	9.47	带宽受限,规则应用	任务并行	信号处理
N-body[47]	23.334	1.0416	0.4672	通信受限,非规则应用	任务并行	物理模拟
K-means[48]	16.2588	3.6576	7.8793	通信受限,非规则应用	数据并行	数据挖掘,稠密线性代数

5.2 测试结果及分析

从表 1、图 5 和图 6 的测试结果可以看出,SPGEMM、Histogram 和 K-means 三类应用在 GPU 异构系统上的计算性能优于 MIC 系统,而 Monte Carlo、FFT 和 N-body 三类应用在 MIC 异构系统上表现出了较好的计算性能.这是因为 SPGEMM、Histogram 和 K-means 属于细粒度高密度的数据并行类应用,而 GPU 是以大量的线程实现面向高吞吐量的数据并行计算,擅长处理计算密度高、逻辑分支简单的大规模数据并行任务,所以以上三种应用在 GPU 上表现出了较好的性能.其它三类应用属于粗粒度任务并行类应用,该类应用是对同一批数据反复进行大量的运算,在 MIC 异构系统上运行则能发挥出 MIC 处理粗粒度并行任务的优势,获得相对较好的计算性能.总体来说,GPU 适合处理细粒度大规模的数据并行型的应用,这类应用属于 SIMD 运算模式,所有运算核心都执行同一个指令,只是作用在不同

的数据上;MIC 更适合处理粗粒度的任务并行类应用,它的特点是对一批数据反复进行大量运算,属 MIMD 的运算模式.用户在具体应用时可以根据应用本身的特征,结合 GPU 和 MIC 不同的计算特点,选择合适的异构平台,以获得较好的计算性能.

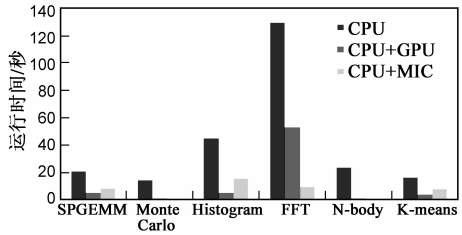

图5　运行时间对比

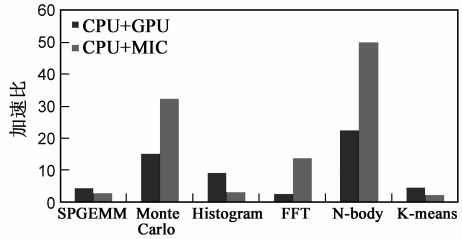

图6　GPU和MIC相对于CPU的加速比

6　GPU 和 MIC 结合的混合异构系统

从以上分析可知,MIC 和 GPU 具有不同的并行计算模式,在运行不同的应用程序时表现出了不同的计算性能.为了能更好的将两种协处理器各自的处理优势充分发挥出来,可以将两种协处理器和 CPU 结合共同组成一种混合架构的异构系统.

在同一个节点内,可以同时配备 MIC 卡和 GPU 卡,从而和节点内的 CPU 组成节点级的混合异构系统,如图 7 所示.CPU 通过协调控制 MIC 卡和 GPU 卡,将一个复杂的任务根据不同的应用特点划分成不同的部分,然后将不同的子任务分派到不同的协处理器上执行,从而充分发挥 MIC 和 GPU 各自的优势来协同完成复杂任务的计算.在该混合异构系统下,不同的加速卡之间可以直接进行通信,协同完成任务的处理,从而可以实现 MIC 和 GPU 优势互补,克服以往异构系统处理任务时的各种限制,进一步扩展异构系统的应用范围,进一步提高复杂应用程序的性能.

该混合异构系统将两种不同的加速卡整合在同一个计算节点内,要能将两种不同的加速设备的优势充分发挥出来,实现较好的协同计算,在编程时要考虑和解决的问题会更复杂.如何划分任务,如何在两种加速设备 GPU 和 MIC 间进行任务调度,如何根据所处理的任务特点在两者之间进行很好的协调,如何设计合理的通信机制,还要有可同时编译两种加速设备代码的编译器的支持等,这些将会是该混合异构系统下必须要研究和解决的问题.

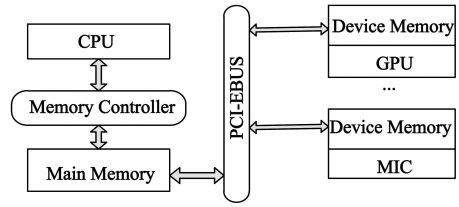

图7　混合异构系统架构

7　总结与展望

本文针对异构众核系统,从系统架构、编程模型和性能优化方面对存在的问题、面临的挑战和具体的研究现状进行了论述,分析了相关的研究和发展动态及应用特点,总结和分析了进一步的研究方向.

总之,异构系统已经成为高性能计算领域一种重要的发展趋势,结合最新的众核技术设计合理、高效、节能的新型异构系统架构对异构系统的发展至关重要.在新的异构系统下,利用协处理器对应用程序进行加速过程中面临的主要问题是:如何充分发挥协处理器的高效处理能力;如何使主处理器和协处理器很好的协作;如何通过较简单的编程方法最大程度的利用异构系统高效的处理能力,提高应用程序的计算性能.解决这些问题要通过设计能够适应新型异构系统、支持细粒度和粗粒度并行、支持可移植性、可扩展性、支持能效编程的简单有效的高层抽象的编程方法来实现.研究能够在异构系统上实现简单高效的开发应用程序的统一的编程模型,使开发出的应用程序能够适应不同的硬件架构和对应的底层支持软件,结合相应编译、运行时系统及性能优化机制,充分发挥异构系统高效的计算能力来处理复杂的科学计算问题,这是提高异构系统可编程性、可移植性、可扩展性,提高能效必须要研究和解决的关键问题,也是目前异构计算重要的研究方向.

参考文献

[1] Brodtkorb A R, Dyken C, Hagen T R, Hjelmervik J M, Storaasli O O. State-of-the-art in heterogeneous computing [J]. Scientific Programming, 2010, 18(1): 1 – 33.

[2] Top 500 supercomputer sites [OL]. http://www.top500.org/, 2012 – 12.

[3] Kothapalli K, Banerjee D S, et al. CPU and/or GPU: Revisiting the GPU Vs CPU Myth[J]. arXiv, 2013, 1303(2171): 1 – 20.

[4] Saha B, Zhou X, Chen H, et al. Programming model for a heterogeneous x86 platform[A]. Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and

Implementation [C]. ACM, 2009. 431 – 440.
[5] Brodtkorb A R, et al. Graphics processing unit(GPU) programming strategies and trends in GPU computing[J]. Journal of Parallel and Distributed Computing, 2013, 73(1): 4 – 13.
[6] Nvidia Corporation. Compute unified device architecture programming guide [OL]. http://developer. nvidia. com/object/cuda. html, 2007.
[7] Yang Y, Xiang P, Mantor M, et al. CPU-assisted GPGPU on fused CPU-GPU architectures[A]. Proceedings of the 18th International Symposium on High Performance Computer Architecture [C]. IEEE, 2012. 1 – 12.
[8] Daga M, Aji A M, Feng W. On the efficacy of a fused cpu + gpu processor(or apu) for parallel computing[A]. Proceedings of the Symposium on Application Accelerators in High-Performance Computing [C]. IEEE, 2011. 141 – 149.
[9] Liu X, Smelyanskiy M, Chow E, et al. Efficient sparse matrix-vector multiplication on x86-based many-core processors[A]. Proceedings of the 27th International Conference on Supercomputing [C]. ACM, 2013. 273 – 282.
[10] Saini S, Jin H, Jespersen D, et al. An early performance evaluation of many integrated core architecture based SGI rackable computing system [A]. Proceedings of the 2013 International Conference for High Performance Computing, Networking, Storage and Analysis [C]. ACM, 2013. 94.
[11] Jeffers J, Reinders J. Intel Xeon Phi Coprocessor High Performance Programming[M]. Newnes, 2013.
[12] Owens J D, Luebke D, Govindaraju N, et al. A survey of general purpose computation on graphics hardware[J] Computer Graphics Forum, 2007, 26(1): 80 – 113.
[13] 王海峰,陈庆奎. 图形处理器通用计算关键技术研究综述[J]. 计算机学报, 2013, 36(4): 757 – 772.
WANG Hai-Feng, CHEN Qing-Kui. General purpose computing ofgraphics processing unit: a survey[J]. Chinese Journal of Computers, 2013, 36(4): 757 – 772. (in Chinese)
[14] 王蕾,等. 任务并行编程模型研究与进展[J]. 软件学报, 2013, 24(1): 77 – 90.
Wang L, et al. Research on task parallel programming model [J]. Journal of Software, 2013, 24(1): 77 – 90. (in Chinese)
[15] Lee S, Min S J, Eigenmann R. OpenMP to GPGPU: a compiler framework for automatic translation and optimization[J]. ACM Sigplan Notices, 2009, 44(4): 101 – 110.
[16] Lee S, Eigenmann R. OpenMPC: extended openMP programming and tuning for GPUs [A]. Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis [C]. IEEE, 2010. 1 – 11.
[17] Han T D, Abdelrahman T S. hiCUDA: High-level GPGPU programming[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(1): 78 – 90.
[18] Baskaran M M, Ramanujam J, Sadayappan P. Automatic C-to-CUDA code generation for affine programs [J]. Compiler Construction, 2010, 6011: 244 – 263.
[19] Linderman M D, Collins J D, Wang H, et al. Merge: a programming model for heterogeneous multi-core systems[A]. Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems[C]. ACM, 2008. 287 – 296.
[20] Dubach C, Cheng P, Rabbah R, et al. Compiling a high-level language for GPUs: (via language support for architectures and compilers)[A]. Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation[C]. ACM, 2012. 1 – 12.
[21] Liu W, Lewis B, Zhou X, et al. A balanced programming model for emerging heterogeneous multicore systems [A]. Proceedings of the 2nd USENIX Conference on Hot Topics in Parallelism[C]. USENIX Association, 2010. 3 – 3.
[22] Gelado I, Stone J E, Cabezas J, et al. An asymmetric distributed shared memory model for heterogeneous parallel systems [A]. Proceedings of the 15th edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems[C]. ACM, 2010. 347 – 358.
[23] Ryoo S, Rodrigues C I, Baghsorkhi S S, et al. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA [A]. Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming[C]. ACM, 2008. 73 – 82.
[24] Baskaran M M, Bondhugula U, Krishnamoorthy S, et al. A compiler framework for optimization of affine loop nests for GPGPUs[A]. Proceedings of the 22nd Annual International Conference on Supercomputing[C]. ACM, 2008. 225 – 234.
[25] Jang B, Schaa D, Mistry P, et al. Exploiting memory access patterns to improve memory performance in data-parallel architectures[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(1): 105 – 118.
[26] Sundaram N, et al. A framework for efficient and scalable execution of domain-specific templates on GPUs[A]. Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing[C]. IEEE, 2009. 1 – 12.
[27] He B, Fang W, Luo Q, et al. Mars: a MapReduce framework on graphics processors[A]. Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques[C]. ACM, 2008. 260 – 269.
[28] Liu Y, Zhang E Z, Shen X. A cross-input adaptive framework for GPU program optimizations[A]. Proceedings of the 2009 International Symposium on Parallel & Distributed Processing [C]. IEEE, 2009. 1 – 10.
[29] Lee J, Lakshminarayana N B, et al. Many-thread aware prefetching mechanisms for gpgpu applications[A]. Proceed-

ings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture [C]. IEEE, 2010. 213 – 224.
[30] Yang Y, Xiang P, et al. A GPGPU compiler for memory optimization and parallelism management[A]. Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation[C]. ACM, 2010. 86 – 97.
[31] Yang Y, Xiang P, et al. Shared memory multiplexing: a novel way to improve GPGPU throughput[A]. Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques[C]. ACM, 2012. 283 – 292.
[32] 张保,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, et al. Novel GPU data partitioning method to overlap communication and computation[J]. Journal of Xi'an Jiaotong University, 2011, 45(4): 1 – 6. (in Chinese)
[33] 张保,等.CPU-GPU 系统中基于剖分的全局性能优化方法[J].西安交通大学学报,2012,46(2):17 – 23.
Zhang Bao, et al. Profiling based optimization method for CPU-GPU heterogeneous parallel processing system[J]. Journal of Xi'an Jiaotong University, 2012, 46(2): 17 – 23. (in Chinese)
[34] Wang P H, Collins J D, Chinya G N, et al. EXOCHI: architecture and programming environment for a heterogeneous multicore multithreaded system[A]. Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation[C]. ACM, 2007. 156 – 166.
[35] Luk C K, Hong S, Kim H. Qilin: exploiting parallelism on heterogeneous multi-processors with adaptive mapping[A]. Proceedings of the the 42nd Annual IEEE/ACM International Symposium on Microarchitecture[C]. IEEE, 2009. 45 – 55.
[36] Jablin T B, Prabhu P, et al. Automatic CPU-GPU communication management and optimization[A]. Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation[C]. ACM, 2011. 142 – 151.
[37] Lee J, Kim J, Seo S, et al. An OpenCL framework for heterogeneous multicores with local memory[A]. Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques[C]. ACM, 2010. 193 – 204.
[38] Reyes R, López-Rodríguez I, Fumero J J, et al. accULL: An OpenACC implementation with CUDA and OpenCL support [J]. Euro-Par Parallel Processing Lecture Notes in Computer Science, 2012, 7484: 871 – 882.
[39] Wienke S, Springer P, et al. OpenACC-first experiences with real-world applications[J]. Euro-Par Parallel Processing Lecture Notes in Computer Science, 2012, 7484: 859 – 870.
[40] Maillard N. Hybrid parallel programming: evaluation of OpenACC[D]. Universidade Federal do rio Grande do Sul, 2012.
[41] Chen C, Yang C, Tang T, et al. OpenACC to Intel offload: automatic translation and optimization[J]. Computer Engineering and Technology Communications in Computer and Information Science, 2013, 396: 111 – 120.
[42] Kumar Pusukuri K, Gupta R, et al. ADAPT: a framework for coscheduling multithreaded programs[J]. ACM Transactions on Architecture and Code Optimization, 2013, 9(4): 45.
[43] Volkov V, Demmel J W. Benchmarking GPUs to tune dense linear algebra[A]. Proceedings of the 2008 ACM/IEEE Conference on Supercomputing[C]. IEEE, 2008. 31.
[44] Stratton J A, et al. Parboil: a revised benchmark suite for scientific and commercial throughput computing[R]. Illinois, US: Center for Reliable and High-Performance Computing of University of Illinois at Urbana-Champaign, 2012.
[45] Podlozhnyuk V, Harris M. Monte Carlo option pricing[R]. California, US: NVIDIA Corporation, 2008.
[46] Govindaraju N K, Lloyd B, Dotsenko Y, et al. High performance discrete Fourier transforms on graphics processors[A]. Proceedings of the 2008 ACM/IEEE Conference on Supercomputing[C]. IEEE, 2008. 2.
[47] Nyland L, Harris M, Prins J. Fast n-body simulation with cuda [J]. GPU Gems, 2007, 3(1): 677 – 696.
[48] Che S, Boyer M, Meng J, et al. Rodinia: A benchmark suite for heterogeneous computing[A]. Proceedings of the 2009 International Symposium on Workload Characterization [C]. IEEE, 2009. 44 – 54.

作者简介

巨　涛　男,1980 年 9 月出生于甘肃兰州,现为西安交通大学电子信息与工程学院计算机系博士研究生.主要研究方向为异构计算、异构并行编程模型及性能优化.
E-mail: immensewaves@163.com

朱正东　男,1963 年 11 月出生于江苏宜兴,分别获西安交通大学学士、硕士、工学博士学位,现为西安交通大学计算机系高级工程师.主要研究方向为高性能计算与云计算.
E-mail: zdzhu@mail.xjtu.edu.cn

董小社(通讯作者)　男,1963 年 10 月出生于陕西咸阳,分别获西安交通大学学士、硕士学位,日本庆应大学工学博士学位.现为西安交通大学计算机系教授、博士生导师.主要研究方向为并行计算机体系结构,网格计算.
E-mail: xsdong@mail.xjtu.edu.cn