异构计算并行编程模型综述

邬江兴,祁晓峰,高彦钊

(中国人民解放军战略支援部队信息工程大学 信息技术研究所,河南 郑州 450000)

摘　要: 异构计算架构是目前高性能计算研究的重要领域。在异构计算架构中,不同种类的计算器件协同工作需要解决如任务调度、数据通信、存储、同步优化等问题。这些问题会对异构计算架构系统的运行性能、功耗、可靠性等指标产生重要影响。为解决异构系统的应用开发与系统优化问题,近年出现许多面向异构计算架构的并行编程模型。本文介绍异构并行编程模型的研究进展,针对异构并行计算需要解决的关键问题进行讨论,最后对异构体系架构的发展方向做出总结。

关键词: 异构计算; 并行编程; 编程模型; 中间表示; 任务调度; 负载均衡

中图分类号: TP 338.6　　**文献标志码:** A　　**DOI:** 10.19328/j.cnki.2096-8655.2021.04.001

Review of Programming Models for Heterogeneous Parallel Computing

WU Jiangxing, QI Xiaofeng, GAO Yanzhao

(Institute of Information Technology, People's Liberation Army Strategic Support Forces Information Engineering University, Zhengzhou 450000, Henan, China)

Abstract: Heterogeneous computing architecture is one of the most important fields in high performance computing. In a heterogeneous architecture, it is complex for different processors working together because of the problems of task scheduling, data movement, memory, synchronism, etc. All of these issues have significant effects on the performance, power consumption, and reliability of the heterogeneous system. In order to optimize heterogeneous system and develop the application effectively, there are many programming models for heterogeneous computing and parallel programming. In this paper, the development of heterogeneous parallel programming models is reviewed. The key of heterogeneous parallel programming is analyzed, and the development trend of heterogeneous computing architecture in the future is summarized.

Key words: heterogeneous computing; parallel programming; programming model; intermediate representation; task scheduling; load balancing

0　引言

科研工程等领域的应用对计算能力(简称算力)的需求越来越大。例如在深度学习模型中,计算系统处理的计算任务呈现出网络规模庞大、问题复杂、需求多样等特征。回顾微处理器发展历史,如图1所示,在20世纪80年代,通过指令集、编译优化、超标量、多级缓存推动了计算性能的提升;90年代,通过提高时钟频率,使计算系统的单线程性能提升,但功耗随之增加;21世纪以来,对多核并行计算的研究逐渐增多,直至当前,摩尔定律逐渐失效。JOHN等[1]在ISCA 2018中指出,单核处理器性能提升的幅度已经缩小至每年只增加3%,高性能计算需要计算体系结构的创新。

计算体系结构创新的内涵有两个方面:其一是规模,即由单核转向多核并行计算。传统的高性能计算研究聚焦在单核性能上,当前则是依靠多核集

收稿日期: 2021-03-17; **修回日期:** 2021-06-25

基金项目: 国家核高基重大专项基金项目(2017ZX01030301)

作者简介: 邬江兴(1953—),男,博士,中国工程院院士,教授,博士生导师,主要研究方向为信息与通信技术、网络空间安全。

通信作者: 祁晓峰(1992—),男,硕士,助理研究员,主要研究方向为高性能计算。

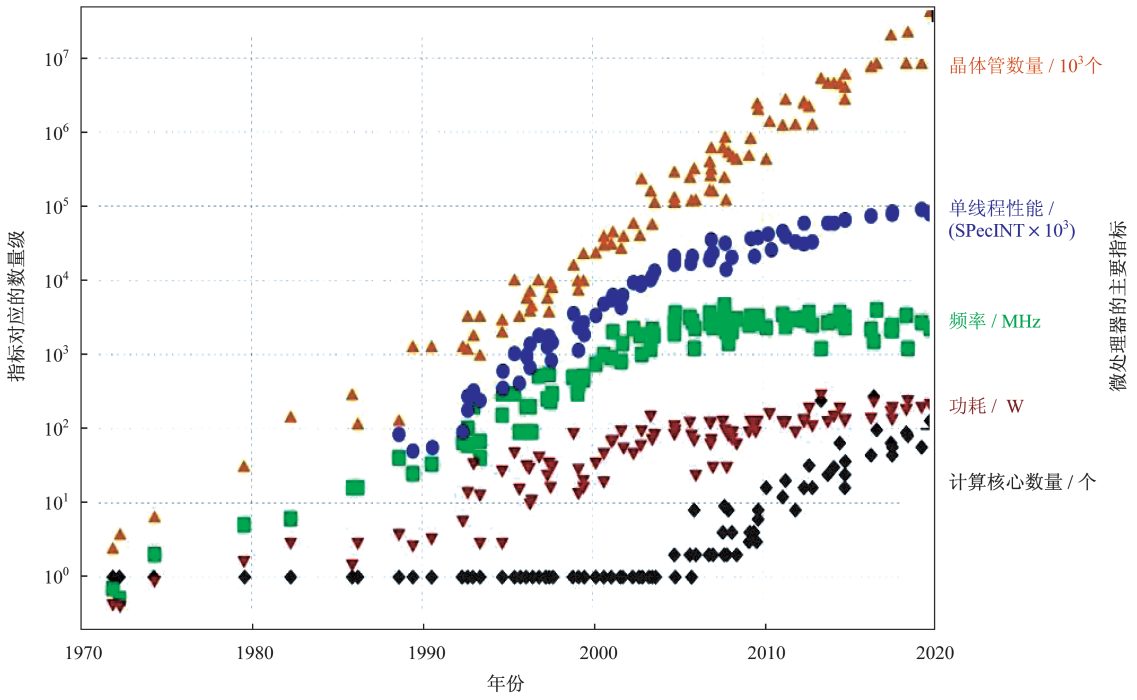

图 1　48 年微处理器发展趋势[2]

Fig.1　48 years of microprocessor trend data[2]

群，集中分布式算力，通过规模化效应提升算力。其二是架构，如中央处理器(Center Process Unit，CPU)、图像处理器(Graph Process Unit，GPU)和可编程门阵列(Field Program Gate Array，FPGA)等计算器件均有各自擅长的处理领域，按照一定组织方式组成多核异构的系统。每种计算器件有各自计算存储模型，不同计算器件的组合使得系统具有异构性。目前，混合两种及两种以上计算器件的多核异构芯片、系统、集群都已经出现。研究人员试图组合不同的计算器件，从性能、功耗等方面的需求出发，通过新型计算体系架构将硬件性能发挥到极致。

异构计算器件组合可以满足多样化的计算任务需求。但是，器件集成在同一系统并不意味着算力相加。如图 2 所示异构系统中，包含不同计算器件，但是各个计算器件的计算、存储和连接等均有独立标准，异构器件之间的任务划分、调度、通信、存储、同步等机制的不同为系统管理带来新的问题。若不解决这些问题，异构系统难以发挥其理论上的性能和能耗优势，异构特征反而会成为系统负担。

解决该问题的难度体现在开发者既要具备丰富的硬件架构知识，又要熟悉各种软件优化方法。

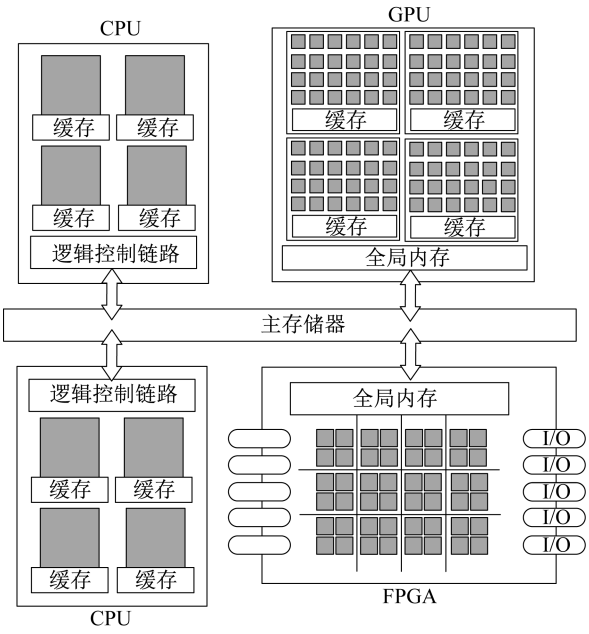

图 2　异构计算系统抽象表示

Fig.2　Abstraction architecture of heterogeneous computing system

对开发者而言，易用的软硬件协同编程工具和自动优化功能是异构计算编程追求的目标。因此，对异构并行编程模型及运行优化的研究是异构并行系统研究的重要内容。本文回顾和梳理异构系统并行编程最新成果，介绍了代表性异构编程模型，研

究了异构编程模型优化方法,讨论了异构并行计算的关键问题,最后对异构计算体系架构的研究方向进行总结。

1　异构并行编程模型

异构并行编程模型把底层硬件抽象表示,提供统一编程接口,解决异构系统并行计算、存储、通信等一系列问题。文献[3]介绍了主流的异构并行编程模型,例如,OpenCL[4]、CUDA[5]等。早期编程模型接口相对低级,开发者需大量手工操作实现细节优化,这不仅增加编程难度导致较低开发效率,也会导致适用性差、易出错等问题。文献[6]总结了各种异构系统编程模型的接口分类,补充许多实验室异构编程模型项目,例如,Merge[7]、C++AMP[8]、Lime[9]、Copperhead[10]、Grag[11]等。这些编程模型的出现一定程度上克服了CUDA、OpenCL等编程模型的缺点。

分析现有异构并行编程模型优化研究,可分为两种:一种从硬件底层优化出发,针对特定异构架构设计编程模型;另一种是基于现有编程框架改进或扩展,通过封装编程接口降低编程难度,提高现有异构编程框架的可用性。

1.1　显式异构编程模型

多数异构编程模型针对特定架构,例如CPU+GPGPU的架构。异构系统处理任务划分、调度、通信、同步等问题,对应大量的编程接口。异构编程模型以显式编程形式实现异构系统优化,代表性的编程模型有OpenCL、CUDA。它们采用Host/Device的分工形式,Host运行CPU的控制代码,Device运行加速设备的代码,如图3和图4所示。

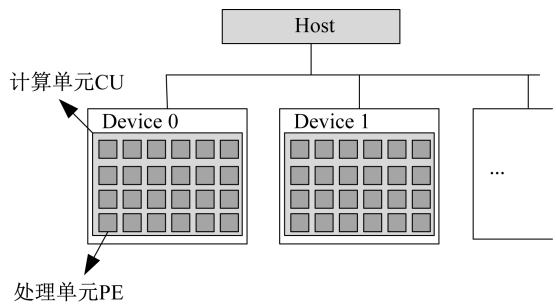

图3　OpenCL编程模型

Fig.3　OpenCL programming model

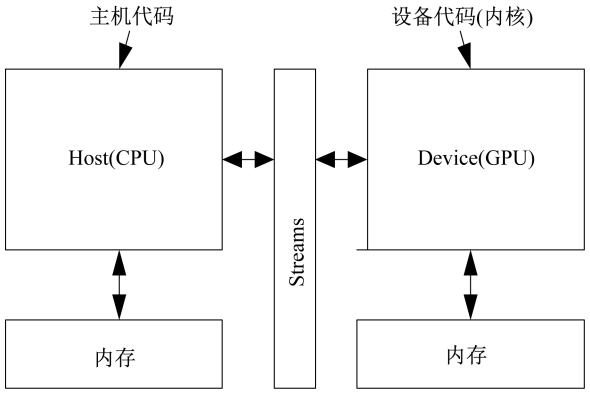

图4　CUDA编程模型

Fig.4　CUDA programming model

CUDA和OpenCL的编程接口偏底层,因此,其对开发能力要求高,并且对并行程序的分析和验证比较困难。因此,对显式编程模型的优化研究是将复杂的编程接口简化,构建结构化的编程框架。例如,文献[12]使用脚本语言和OpenCL设计结构化的并行编程框架,通过脚本调用OpenCL接口控制任务。文献[13]提出一种结合OpenCL和CUDA的接口库HCLOOC,解决计算核心和多核通用CPU之间通信链路有限带宽的约束问题。HCLOOC库包含调用各类计算硬件的基本模块,支持OpenCL命令队列、任务卸载、复制和执行CUDA流等功能。

1.2　制导式编程模型

制导式编程模型(pragma)方法在源代码上以注释的形式,告诉编译器并行代码区域的位置,如何在代码的不同部分访问变量以及如何在同步点执行。编译器根据制导语句生成代码、执行代码段、传输数据和执行同步操作。生成代码通过调用运行时系统管理硬件资源以及跨不同内存层级传输数据。代表性的制导式异构编程模型包括OpenACC[14]、OpenMP[15]、OpenHMPP[16]等。

对制导式异构编程模型研究主要是基于内存的优化。例如,文献[17]提出了一个基于OpenMP开发的工具包HyCOMP,工具包模拟分布式CPU和GPU上的虚拟共享内存空间,分布式共享内存系统可以有效防止GPU在处理主机到设备内存拷贝过程中,因大量页错误而导致的性能下降。制导式异构编程模型简化了异构并行编程的难度,并对优化异构系统性能起到显著作用。

2 异构编程模型的优化

2.1 异构编程中间表示

在生成执行机器码过程中,运行时系统优化代码操作能够有效提高系统性能,即中间表示(Intermediate Representation, IR)技术。HAS[18]、ROCm[19]、CUDAPTX[5]、SPIR[20]等都是专门为异构体系结构设计的IR。

HSA是由HSA基金会开发的异构体系结构规范,HSA定义了中间语言HSAIL(HSA Intermediate Language),包括对异常、虚拟函数和系统的调用。AMD基于HSA和HSAIL开发了ROCm,针对OpenCL和CUDA程序进行编译。CUDA PTX是NVIDIA为CUDA程序开发的中间表示。CUDA编译器将CUDA代码转换为PTX IR,但仅适用于NVIDIA系列的GPU。SPIR是一种基于LLVM-IR的二进制中间语言,用于图形计算和内核计算。LLVM可以作为多种语言编译器的后台来使用,将代码编译成多种不同硬件架构的指令集。

2.2 异构编程语言接口

开发者希望使用易于理解的高级语言实现抽象编程语言到底层框架的映射,多数研究以此为目标优化异构编程框架。异构编程语言可分为非托管式的和托管式的。

非托管编程语言指可以直接处理内存的编程语言。并行编程框架采用非托管式语言,因为它能为系统提供更高性能。文献[21-25]使用类C语言,设计高层级统一的并行编程架构。通过源到源的编译器,将类C语言转成CUDA或OpenCL代码,实现不同规模的异构底层映射。使用非托管式编程语言的特点是所有规则、语法、执行模型和内存模型都是能够通过编程语言定义。但是非托管语言的抽象级别相对较低,需要程序员具备更多的软硬件开发专业知识。

托管式编程语言是指内存可由运行时系统自动管理的编程语言。托管式编程语言实现内存自动管理,依赖于高效的语言虚拟机、解释器和编译器执行。文献[26-28]基于Java语言设计异构并行计算编程架构,以高级抽象的方式管理硬件资源,并在内部运行时系统进行中间优化操作。文献[14]提供Python封装接口,使用PyOpenCL调用接口。文献[29]设计高级语言Occanpi,用于异构粗粒度硬件构建、编译和生成机器码,解决异构可重构平台的可访问性问题。尽管托管编程语言简化了开发过程,但抽象程度越高,平台性能优化越困难。例如在Java中,所有数组都从Java端复制到JNI(Java Native Interface)端。JNI需要为所有数组创建额外副本。因此,应用程序的性能将低于用非托管语言实现的应用程序性能。

2.3 融合的并行编程模型

异构计算的规模不仅限于芯片,还扩展到异构计算集群。大规模异构计算集群对计算节点的并行性能要求增加,系统往往具有更为复杂的并行层次和内存层级。文献[30]使用多核控制器管理异构设备,可跨不同计算硬件类型移植通用内核,简化数据分区和映射。运行时系统自动为每个设备选择和部署合适的内核,管理数据移动并隐藏启动细节。

高度并行异构系统需要处理好粗粒度并行和细粒度并行。混合粗粒度和细粒度并行的编程框架,通常为MPI+X的多层结构。文献[31]设计一种通用、混合和优化稀疏的工具包,按照MPI+X的范式,为多核计算器件提供内核和资源管理。文献[32-35]均采用MPI+OpenMP+CUDA(MOC)混合并行编程模型,通过基于节点间(粗粒度)和节点内(细粒度)的并行处理架构实现CPU和GPU协作的大规模异构并行计算。文献[36-37]则是提出MPI+OpenMP+OpenCL的并行化方法,增强异构系统的数据可移植性。

2.4 虚拟化异构编程模型

尽管MOC范式的融合异构并行编程框架具有良好的性能和能效,但是其编程复杂度较高。文献[38]提出了一种基于异构集群的分布式虚拟机。虚拟机实际是一个分布式系统,将混合的CPU和GPU看作单一的大规模GPU集群,只需要使用CUDA开发应用程序,而不需要将MPI和多线程API结合起来。以虚拟化的方式消除异构系统开发、设备内存空间和线程配置的复杂性,可以实现自动化处理异构处理器间负载均衡、设备内存和线程的配置约束问题。

2.5　小结

本章详细介绍了异构并行编程优化的最新研究成果,优化目标从编程难度、计算性能、并行规模3个维度出发,对异构架构并行编程的技术挑战提出部分解决方案。实际上,一个异构并行编程框架难以同时具备编程难度低、计算性能高和异构规模大这3个特点,只能满足三取其二。这种特征构成了异构并行编程优化的“不可能三角”,如图5所示。

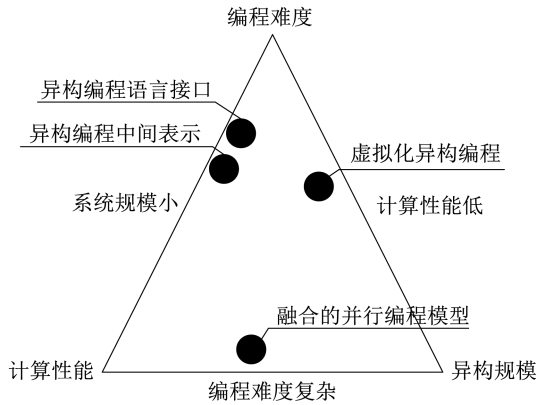

图5　异构并行系统“不可能三角”

Fig.5　Impossible trinity of heterogeneous parallel system

这个“不可能三角”的3个顶点分别是编程难度低、计算性能高、并行规模大,即异构并行编程模型优化的3个目标。然而这3个目标最多只能做到2个。例如,如果追求编程模型易上手,通过少量代码或指令实现异构编程,同时保证系统具有高度并行和计算能力。这种情况下就需要高度封装的编程语言接口或者中间表示优化,从而牺牲编程灵活性,因此,难以在大规模异构系统上实现。另外两种情况也类似。如果想要计算性能高,同时异构规模大,那就需要多层级存储和混合颗粒度的并行编程框架,这将增加系统编程的复杂度;如果想要编程难度低,并且异构规模大,那么目前的方法就是采用虚拟化的方式,牺牲一定系统性能,从而实现大规模异构计算资源的统一管理。因此,在进行异构编程模型优化时,核心是两个方面:

1) 对开发者来说,只有异构编程模型的编程复杂度是可控的,因此,开发或优化新型异构编程模型应由实际需求驱动。

2) 在当前研究框架下,由于优化目标相互限制,还未出现完美的异构编程框架,理想的研究方向应是如何避免这个三元悖论或者将3个优化目标进行折解。

3　异构并行计算关键问题

提高资源利用率和减少通信开销是异构计算研究的关键问题。为充分提高资源利用率,需要考虑并行任务使用哪些硬件计算资源;在不同的计算器件上,任务处理速度和效率不同,系统性能很大程度上受调度方案的影响,合理的负载均衡策略、服务质量管理都可以提高资源利用率。异构系统各类计算器件的传输、存储方式不同,合理的数据存储结构能够节省大量的数据读取时间;数据传输时可以通过叠加或流水等方式将计算或存储操作隐藏。此外,对系统性能的评估也需要灵活的评估方法和工具,综合考虑系统通信、能耗、可靠性以及其他技术指标的平衡。下面分别介绍异构系统任务调度映射、负载均衡、服务质量、存储优化、传输优化、通信隐藏、性能评估等方面的最新研究成果。

3.1　提高资源利用率

3.1.1　调度映射

同一计算任务在不同的硬件节点中执行,在性能功耗等方面各有所长,因此,异构系统需要将任务分配给合适的计算设备。

调度映射方式可分为静态和动态。静态调度方法利用系统约束和任务属性,将问题转化为多约束条件下的求最优解问题。求解方法分为线性规划方法和启发式方法。规划求解策略根据系统的多约束条件通过线性规划等方式求最优解。例如,文献[39]用线性规划方法,根据处理器类型和时间表约束进行任务调度,最小化通信延迟和能耗。启发式方法的核心是评价函数,以评价函数为标准通过不断迭代找到最优解。例如,文献[40]提出了混合双目标并行遗传算法,采用加权求和的方法,根据用户的偏好平衡能耗和执行时间。文献[41]基于搜索的调度算法,通过剪枝技术缩小搜索空间,通过启发式函数指导最优解搜索。文献[42]使用混合的元启发式集合方法,寻找异构系统的最优帕累托映射方案,同时优化执行时间和能耗。

动态调度方法通过感知系统能耗,动态实时调度子任务。文献[43]把能耗感知作为大规模异构计算环境中并行调度策略的重要参考依据,将不同

并行层级任务映射到目标计算平台。文献[44-45]将动态电压频率标度和动态功率管理等系统参数量化,基于多目标全局优化元启发式算法,研究能耗感知和融合能效感知的调度模型。文献[46-47]在异构系统中,采用启发式的电源感知和能效调度算法,近似求解最小化动态能耗开销的调度策略。

3.1.2 负载均衡

负载均衡主要用于工作负载分配,通常以能耗感知的方式调整计算资源的使用,平衡计算能耗。

1)将特定异构模型和任务调度联合优化。文献[48-49]基于CPU和Intel-MIC结构提出工作负载分配方法,解决计算器件间的负载平衡问题。文献[50]采用聚合计算,改善CPU和加速器之间的负载平衡。文献[51-52]通过基于贪婪策略的异构动态调度器实现负载均衡。文献[53]设计了双层工作负载分配方案,为每个计算节点分配最合适的工作负载,并在每个节点内合理划分CPU和GPU之间的工作负载。文献[54]采用基于FPM的异构分解方法在异构处理设备之间分配工作负载。

2)依据系统能耗的动态反馈结果实现运行时的负载均衡。例如,文献[55]通过工作负载划分算法确定负载分布,最小化应用并行执行的能耗。文献[56]将性能和能耗作为工作负载函数的参数,评估并行平台的资源竞争,返回任务划分的帕累托最优解。文献[57]使用CMT-bone工具进行异构架构的负载均衡,平衡优化性能和能耗。

此外,针对大规模异构集群,文献[58]研究了MOC结构异构环境下,生成可执行程序时自动分配计算负载的方法。

3.1.3 服务质量

由于并行任务在线程同步、数据共享和并行化模式的多样性,管理执行性能要求不同的并行任务是一个复杂的问题。在资源有限的异构计算系统上并行执行任务,为不同任务定义不同的优先级,对异构系统的任务映射和资源管理具有重要意义。文献[59]研究异构系统上的并行任务服务质量管理方式,通过分析一系列任务映射策略对应用程序性能的影响,认为在异构硬件上的有效的服务质量管理策略能提高任务映射策略的性能,同时保持了系统优化目标的平衡。

3.2 减少通信开销

3.2.1 数据存储

异构体系结构中的共享末级缓存(Last Level Cache,LLC)对系统的整体性能至关重要,当与CPU应用程序协同运行时,GPU应用程序很容易占据LLC的大部分,使得CPU资源严重匮乏,针对异构体系共享LLC的设计和管理,文献[60]将传统的SRAM与新的存储器技术STT-RAM相结合来扩展共享LLC,并通过综合管理策略减少计算器件间的资源争用。

3.2.2 数据传输

异构系统计算器件上的数据需要频繁交互。一方面,主节点与设备节点进行同步,会对系统带来额外的开销。文献[61]提出优化消息传递模型MEMPHA减少异构器件之间的拥塞,缩短总通信量。文献[62-63]设计并行算法,通过GPU排序队列减少通信开销。另一方面,数据传输过程的接口协议需要相互转换。针对数据转换,文献[64]提出一种动态二进制转换任务迁移方法,减少因数据转换造成的开销。

3.2.3 通信隐藏

异构规模的增加使通信延迟成为性能瓶颈。为减少通信延迟,文献[65]采用空间域分解和射线并行相结合的方法进行异构并行计算,使MPI通信时间与GPU上的传输扫描时间重叠,隐藏GPU与CPU之间的通信和数据传输时间。文献[66]通过性能分析模型识别性能影响因素,实现数据计算与数据传递操作相互重叠。文献[67]提出流水线混合并行的优化方法,加速GPU之间的流水线混合并行,减少通信延迟和内存使用频率。

3.3 性能评估

异构系统还包括对计算和通信行为的性能评估模型的研究。文献[68-69]设计形式化的通信性能的表达方式,用于评估通信代价、最小化通信开销。文献[70]提出一种精确通信成本评估函数,用于在执行数据并行应用程序的过程中,评估混合平台中计算存储数据移动的通信时间。异构系统优化的另一个目标是准确估计不同计算器件的能耗。文献[71]采用系统级功率测量表估计应用组件级能耗。对于安全性要求高的并行应用,必须满足可靠性目标的同时最小化成本。文献[72]针对异构

嵌入式系统上并行计算的资源消耗成本最小化问题,将应用程序的可靠性目标转化为每个任务的可靠性目标并分配给处理器,以启发式的方法最小化能耗。

3.4 小结

本章详细介绍了异构并行计算在提高资源利用率,减少通信开销以及系统性能评估等方面的最新研究成果。相比于传统并行编程模型的计算方法,异构模型增加了系统的计算复杂度,这种计算复杂度主要体现在三个方面:

1) 任务的编译不仅面向CPU,还要映射到GPU、FPGA等含有局部存储结构的加速器件上。编译过程增加了任务划分、映射、负载和执行优先级等工作,直接增加系统计算负担。

2) 异构器件的存储和传输成为新的性能瓶颈,根据计算任务不同,系统应支持以灵活的方式改变存储结构及通信传输方式实现性能优化。

3) 异构特性使系统在评估性能时的可选指标更加多元化,因此,应支持根据实际需求或应用场景的变化,可灵活定义异构系统内部结构及其评价标准。

从以上特点可以看出,当前异构并行计算优化方法主要继承传统并行计算优化中的任务映射、分级存储和并行流水等策略,在策略的生成过程中人工干预较多,系统缺乏主动认知和自主决策的能力,并且明显缺点是各类计算优化只针对特定体系结构的异构系统。从研究趋势以及研究需求上看,未来计算体系结构的特点应具备“应用决定结构”的主动认知的动态重构过程,因此,对灵活可变特征的异构体系结构模型有待进一步研究。

4 结束语

本文回顾了主流异构并行编程模型及研究进展,针对异构并行计算关键问题进行讨论。前文已经指出异构并行计算架构系统的三元悖论对开发者来说一直都是挑战,随着计算器件多样化,支持软件定义的异构融合模型软件工具链内涵仍需不断扩展和完善。2008年,邬江兴院士根据“结构决定功能、结构决定性能、结构决定效能”的公理,提出拟态计算(Mimic Structure Calculation,MSC)架构。拟态计算是在异构计算的基础上实现基于主动认知的多维环境动态重构,实时感知计算任务关于时间的负载分布和能耗状况,调度合适的软硬件功能模块,协同完成计算任务以拟合期望的能效曲线。结合拟态计算的思想,理想的异构编程模型应具备融合多类计算资源、结构的主动感知和动态重构的特征。异构融合的编程模型能够提供灵活的底层资源的抽象,通过软件定义形式实现多模态计算模型。结合研究团队在拟态计算和高效能计算方面的积累,对新型异构体系架构的研究有以下几个方向:

1) 软件定义互连。异构内涵不断丰富,未来的网络体系包含多种异构计算资源,存储资源和相互间的各种网络连接资源。软件定义互连基于可重构和可编程技术,具有可扩展的硬件/软件架构,允许应用程序对硬件进行在线定义,动态更改交换互连的功能、性能,实现对不同应用场景的最佳拟合适配。软件定义互连将是针对未来多样化需要的新型网络技术,是交换与互连技术及产业的未来演进形态,其发展趋势主要包括两点:一是突破软件定义互连技术基础理论、基础支撑技术和共性关键技术问题,形成覆盖标准规范、芯片、设备、网络平台、软件、工具等在内的体系化成果;二是着眼关键应用的新型体系结构发展,打造支持软件定义互连为内核的产业体系,形成从终端到边缘计算、汇聚接入、核心交换的软件定义大互连生态。

2) 内生安全。异构体系架构的核心特征是基于能效的软硬件变结构协同计算和基于性能的变结构协同处理。在此基础上,拟态计算架构支持在效能和性能目标间自由转换,并具备动态优化和管理的功能。拟态计算所具有的多样性、动态性和随机性的协同处理特点,恰好能够弥补传统系统在应对基于内生安全问题攻击时的静态性、确定性和相似性安全缺陷。因此,利用对威胁感知的功能等价动态变结构协同处理环境的非确定性,就可以创造以内源性的“测不准”效应规避内生安全问题的新型防御体制。从应用维度上看,基于拟态计算的异构变结构协同计算能够适配基于构造效应的内生安全机制。在未来的研究中,功能等价条件下的软硬件变结构协同处理体制机制将成为解决内生安全问题的关键。

3) 类脑计算。类脑计算以融合科学计算和神经网络计算为核心,基于拟态计算与软件定义互连

技术,实现软件与硬件、控制与计算的分离,计算、存储、互连等资源可灵活定义的"左右脑"融合计算架构。类脑计算通过资源动态自适应匹配方法,构建主动认知重构的处理结构,生成多目标约束下的资源调度方法,实现任务与资源的优化匹配。基于软件定义的可重构计算方法结合人脑工作机理的抽象与建模,实现深度神经网络与拟态神经计算的融合,推动人工智能从原理模型、算法到实现架构的发展,为边缘终端、移动终端等设备的智能化实时信息处理提供强力支撑,是未来计算系统发展的一条可行的技术途径。

随着软件定义互连、内生安全、类脑计算技术的发展,本团队将应用场景进一步扩展至晶上系统(System on Wafer,SoW)领域。软件定义晶上系统立足我国集成电路成熟工艺和工具"代差落后"的基本国情,借助领域专用软硬件协同计算结构和晶圆级互连拼装集成的联合迭代创新,解决SoW发展初期领域专用混合粒度预制件设计、异构多Die之间互连、动态可重构网络等关键问题,并将作为SoW内部Die-to-Die之间的计算互连规范,致力于构建一个开放式的SoW生态。

参考文献

[1] JOHN H, DAVID P. A new golden age for computer architecture: domain-specific hardware/software co-design, enhanced security, open instruction sets, and agile chip development[C]// International Symposium on Computer Architecture (ISCA). Washington D. C., USA: IEEE Press, 2018: 27-29.

[2] MOHAMED Z. Heterogeneous computing: hardware and software perspectives [M]. New York: ACM, 2018:3-4.

[3] XIE G, ZHANG Y L. A few of the most popular models for heterogeneous parallel programming [C]// International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES). Washington D. C., USA: IEEE Press, 2017:15-18.

[4] OpenCL™. Open standard for parallel programming of heterogeneous systems[EB/OL]. (2020-09-30) [2021-02-28]. https://www.khronos.org/opencl/.

[5] DeveloperNvidia. CUDA工具包[EB/OL]. (2020-03-14) [2021-02-28].https://developer.nvidia.com/zh-cn/cuda-toolkit.

[6] 刘颖,吕方,王蕾,等.异构并行编程模型研究与进展[J].软件学报,2014,25(7):1459-1475.

[7] LINDERMAN M D, COLLINS J D, WANG H, et al. Merge: aprogramming model for heterogeneous multi-core systems [C]// Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). New York, USA: ACM, 2008:287-296.

[8] GREGORY K, MILLER A. C++ AMP: accelerated massive parallelism with Microsoft Visual C++ [J]. Microsoft Press Corp, 2012, 37(5):475-482.

[9] AUERBACH J S, BACON D F, CHENG P, et al. Lime: a Java-compatible and synthesizable language for heterogeneous architectures [J]. ACM SIGPLAN Notices, 2010, 45(10):89-108.

[10] BRYAN C, MICHAEL G, KURT K. Copperhead: compiling an embedded data parallel language [C]// 16th ACM symposium on Principles and Practice of Parallel Programming (PPoPP '11). New York, USA: ACM, 2011: 47-56.

[11] RAHUL G, JOSÉ N A. Compiling python to a hybrid execution environment[C]// 3rd Workshop on General-Purpose Computation on Graphics Processing Units. New York: ACM, 2010: 19-30.

[12] 李安民,计卫星,廖心怡,等.一种面向异构计算的结构化并行编程框架[J].计算机工程与科学,2019,41(3):424-432.

[13] KHALEGHZADEH H, ZHONG Z, REDDY R, et al. Out-of-core implementation for accelerator kernels on heterogeneous clouds [J]. Journal of Supercomputing, 2018,74(2) :551-560.

[14] What is OpenACC? [EB/OL]. (2020-11-18) [2021-02-28].https://www.openacc.org/.

[15] The OpenMP examples document has been updated with new features found in the OpenMP 5.0 Specification [EB/OL]. (2020-06-01) [2021-02-28]. https://www. openmp. org/wp-content/uploads/openmp-examples-5-0-1.pdf.

[16] HIRAM E C. OpenHMPP[R]. England, UK: NHBS Ltd, 2012.

[17] LI H F, LIANG T Y, LIN Y J. An OpenMP programming toolkit for hybrid CPU/GPU clusters based on software unified memory [J]. Journal of Information Science and Engineering, 2016, 32(3): 517-539.

[18] HSA Foundation China Regional Committee & China Standards Group for heterogeneous system architecture 2nd technical symposium to be held in Nanjing [EB/OL]. (2018-06-12) [2021-02-28]. http://www.hsafoundation.com/.

[19] GitHub-OpenMp/examples: Latex examples document source [EB/OL]. (2016-11-10) [2021-02-28]. https://github.com/RadeonOpenCompute/ROCm.

[20] SPIR. The industry open standard intermediate language for parallel compute and graphics [EB/OL]. (2019-09-13) [2021-02-28]. https://www. khronos.org/spir/.

[21] 吴树森,董小社,王宇菲,等.UPPA:面向异构众核系统的统一并行编程架构[J].计算机学报,2020,43(6):990-1009.

[22] 李雁冰,赵荣彩,韩林,等.一种面向异构众核处理器的并行编译框架[J].软件学报,2019,30(4):981-1001.

[23] ANDRÉS R, NAVARRO A, ASENJO R, et al. Parallel multiprocessing and scheduling on the heterogeneous Xeon+FPGA platform[J]. The Journal of Supercomputing, 2020, 76(10): 4645-4665.

[24] SOUROURIM, BADEN S B, CAI X. Panda: a compiler framework for concurrent CPU+GPU execution of 3D stencil computations on GPU-accelerated supercomputers[J]. International Journal of Parallel Programming, 2017, 45(3):1-19.

[25] SOUROURIM, BADEN S B, CAI X. Panda: a compiler framework for concurrent CPU+GPU execution of 3D stencil computations on GPU-accelerated supercomputers[J]. International Journal of Parallel Programming, 2017, 45(3): 711-729.

[26] AMIR K. Execution migration of managed languages programs among heterogeneous-ISA processing units [C]// 21st International Middleware Conference Doctoral Symposium (Middleware' 20 Doctoral Symposium). New York: ACM, 2020: 38-41.

[27] PINHEIRO A, JUNIOR F, ARRUDA N, et al. Fusion: abstractions for multicore/manycore heterogenous parallel programming using GPUs[C]// Brazilian Symposium on Programming Languages. Heidelberg, German: Springer, 2014: 109-110.

[28] ALEEM M, PRODAN R, ISLAM M A, et al. On the parallel programmability of Javasymphony for multi-cores and clusters[J]. International Journal of Ad Hoc and Ubiquitous Computing, 2019, 30(4): 247-264.

[29] ZAIN U A, SVENSSON B. A retargetable compilation framework for heterogeneous reconfigurable computing[J]. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2016, 9(4): 24.

[30] MORETON F A, GONZALEZ E A, LLANOS D R. Multi-device controllers: a library to simplify parallel heterogeneous programming[J]. International Journal of Parallel Programming, 2019, 47: 94-113.

[31] KREUTZER M, THIES J, RHRIG-ZLLNER M, et al. GHOST: building blocks for high performance sparse linear algebra on heterogeneous systems[J]. International Journal of Parallel Programming, 2017, 45: 1046-1072.

[32] ASHRAF M U, EASSA F A, ALBESHRI A A, et al. Performance and power efficient massive parallel computational model for HPC heterogeneous exascale systems[J]. IEEE Access, 2018, 6: 23095-23107.

[33] ASHRAF M U, EASSA F A, ALBESHRI A A, et al. Toward exascale computing systems: an energy efficient massive parallel computational model [J]. International Journal of Advanced Computer Science and Applications, 2018, 9(2):118-126.

[34] SONG P T, ZHANG Z J, ZHANG Q, et al. Implementation of the CPU/GPU hybrid parallel method of characteristics neutron transport calculation using the heterogeneous cluster with dynamic workload assignment[J]. Annals of Nuclear Energy, 2020, 135: 106957.

[35] KYLASA S B, AKTULGA H M, GRAMA A Y. Reactive molecular dynamics on massively parallel heterogeneous architectures[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28 (1): 202-214.

[36] GOROBETS A, SOUKOV S, BOGDANOV P. Multilevel parallelization for simulating compressible turbulent flows on most kinds of hybrid supercomputers [J]. Computers & Fluids, 2018, 173:171-177.

[37] OYARZUN G, BORRELL R, GOROBETS A, et al. Efficient CFD code implementation for the ARM-based Mont-Blanc architecture [J]. Future Generation Computer Systems, 2017, 79(3):786-796.

[38] LIANG T Y, LI H F, LIN Y J, et al. A distributed PTX virtual machine on hybrid CPU/GPU clusters[J]. Journal of Systems Architecture, 2016, 62: 63-77.

[39] AITABA M, ZAOURAR L, MUNIER A. Efficient algorithm for scheduling parallel applications on hybrid multicore machines with communications delays and energy constraint [J]. Concurrency and Computation Practice and Experience, 2020, 32(15): e5573.

[40] SAROJA S, REVATHI T. Hybrid dual-objective parallel genetic algorithm for heterogeneous multiprocessor scheduling [J]. Cluster Computing, 2019, 23: 441-450.

[41] DIETZE R, RÜNGER G. The search - based scheduling algorithm HP* for parallel tasks on heterogeneous platforms [J]. Concurrency and Computation Practice and Experience, 2020, 32(12): e5898.

[42] BELKACEMI D, DAOUI M, BOUZEFRANE S, et al. Parallel applications mapping onto network on chip based on heterogeneous MPSoCs using hybrid algorithms [J]. International Journal of Distributed Systems and Technologies (IJDST), 2019, 10(2): 37-63.

[43] VU T T, DERBEL B. Parallel branch-and-bound in multi-core multi-CPU multi-GPU heterogeneous environments [J]. Future Generation Computer Systems, 2014, 56: 95-109.

[44] 王静莲,龚斌,刘弘,等.支持绿色异构计算的能效感知调度模型与算法[J].软件学报,2016,27(9):2414-2425.

[45] 王静莲,龚斌.面向异构计算的能效感知调度研究[J].电子学报,2016,44(4):893-897.

[46] SCHRANZHOFER A, CHEN J J, THIELE L. Dynamic power-aware mapping of applications onto heterogeneous MPSoC platforms [J]. IEEE Transactions on Industrial Informatics, 2010, 6(4): 692-707.

[47] ZHOU J L, WEI T Q, CHEN M S, et al. Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2016, 35(8): 1269-1282.

[48] SZUSTAK L, HALBINIAK K, KUCZYNSKI L, et al. Porting and optimization of solidification application for CPU-MIC hybrid platforms [J]. Experimental Mechanics, 2018, 32(4):523-539.

[49] MIAO X Q, JIN X L, DING J H, et al. An approach to enhance the performance of large-scale structural analysis on CPU-MIC heterogeneous clusters [J]. Concurrency and Computation Practice and Experience, 2017, 29(8): e4033.

[50] COJEAN T, GUERMOUCHE A, HUGO A, et al. Resource aggregation for task-based cholesky factorization on top of modern architectures[J]. Parallel Computing, 2019, 83(4):73-92.

[51] 王鸿琰,关雪峰,吴华意.一种面向CPU/GPU异构环境的协同并行空间插值算法[J].武汉大学学报(信息科学版),2017,42(12):1688-1695.

[52] WANG H Y, GUAN X F, WU H Y. A hybrid parallel spatial interpolation algorithm for massive LiDAR point clouds on heterogeneous CPU-GPU systems [J]. Multidisciplinary Digital Publishing Institute, 2017, 6(11): 363.

[53] WAN L J, LI K L, LI K Q. A novel cooperative accelerated parallel two-list algorithm for solving the subset-sum problem on a hybrid CPU-GPU cluster[J]. Journal of Parallel and Distributed Computing, 2016, 97(11): 112-123.

[54] LIU X C, ZHONG Z M, XU K. A hybrid solution method for CFD applications on GPU-accelerated hybrid HPC platforms[J]. Future Generation Computer Systems, 2016, 56: 759-765.

[55] KHALEGHZADEH H, FAHAD M, MANUMACHU R R, et al. A novel data partitioning algorithm for dynamic energy optimization on heterogeneous high performance computing platforms [J]. Concurrency and Computation Practice and Experience, 2020, 21(7): e5928.

[56] KHALEGHZADEH H, FAHAD M, SHAHID A, et al Bi-objective optimization of data-parallel applications on heterogeneous HPC platforms for performance and energy through workload distribution [J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(3): 543-560.

[57] GADOU M, BANERJEE T, ARUNACHALAM M, et al. Multiobjective evaluation and optimization of CMT-bone on multiple CPU/GPU systems [J]. Sustainable Computing: Informatics and Systems, 2019, 22(6): 259-271.

[58] ANDRIANOV A N, BARANOVA T P, BUGERYA A B. Distribution of computations in hybrid computing systems when translating NORMA language programs [J]. Computational Methods and Programming, 2019,

20(3): 224-236.

[59] ZHANG Y, ZHAO L, ILLIKKAL R, et al. QoS management on heterogeneous architecture for multiprogrammed, parallel, and domain-specific applications[J]. IEEE Systems Journal, 2017, 11(4): 2096-2107.

[60] GAO L, WANG R, XU Y L, et al. SRAM- and STT-RAM-based hybrid, shared last-level cache for on-chip CPU-GPU heterogeneous architectures [J]. The Journal of Supercomputing, 2018, 74(7): 3388-3414.

[61] KOOHI S Z, HAMID N A W A, OTHMAN M, et al. MEMPHA: model of exascale message-passing programs on heterogeneous architectures [J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(11): 2570-2581.

[62] VIDAL P, ALBA E, LUNA F. Solving optimization problems using a hybrid systolic search on GPU plus CPU[J]. Soft Computing, 2016, 21(12): 1-19.

[63] GOWANLOCK M, KARSIN B. A hybrid CPU/GPU approach for optimizing sorting throughput[J]. Parallel Computing, 2019, 85(7):45-55.

[64] SIMON R, ERVEN R, STEVEN D. Hybrid-DBT: hardware/software dynamic binary translation targeting VLIW [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 38(10): 1872-1885.

[65] LIANG L, ZHANG Q, SONG P T, et al. Overlapping communication and computation of GPU/CPU heterogeneous parallel spatial domain decomposition MOC method [J]. Annals of Nuclear Energy, 2020, 135(1):1-9.

[66] SONG P T, ZHANG Z J, LIANG L, et al. Study on optimization of parallel efficiency of CPU-GPU heterogeneous parallelization for MOC neutron transport calculation [J]. Atomic Energy Science and Technology, 2019, 53(11):2209-2217.

[67] ZHANG J H, ZHAN J, LI J G, et al. Optimizing execution for pipelined-based distributed deep learning in a heterogeneously networked GPU cluster [J]. Concurrency and Computation Practice and Experience, 2020, 32(23): e5923.

[68] RICO-GALLEGO J A, MORENO- ÁLVAREZS, DÍAZ-MARTÍNJ C, et al. A tool to assess the communication cost of parallel kernels on heterogeneous platforms [J]. The Journal of Supercomputing, 2019, 76(10): 4629-4644.

[69] RICO-GALLEGO J A, DÍAZ-MARTÍN J C, CALVO-JURADO C, et al. Analytical communication performance models as a metric in the partitioning of data-parallel kernels on heterogeneous platforms [J]. Journal of Supercomputing, 2019, 75(12): 1654-1669.

[70] MALIK T, LASTOVETSKY A. Towards optimal matrix partitioning for data parallel computing on a hybrid heterogeneous server[J]. IEEE Access, 2021, 9(1):17229-17244.

[71] FAHAD M, SHAHID A, MANUMACHU R R, et al. Accurate energy modelling of hybrid parallel applications on modern heterogeneous computing platforms using system-level measurements [J]. IEEE Access, 2020, 8(3): 93793-93829.

[72] XIE G Q, CHEN Y K, LIU Y, et al. Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems [J]. IEEE Transactions on Industrial Informatics, 2017, 13(4): 1629-1640.