DOI:10.3969/j.issn.1673-4785.201403072
网络出版地址:http://www.cnki.net/kcms/doi/10.3969/j.issn.1673-4785.201403072.html

GPU 通用计算及其在计算智能领域的应用

丁科[1,2],谭营[1,2]

(1.北京大学 机器感知与智能教育部重点实验室,北京 100871; 2.北京大学 信息科学技术学院,北京 100871)

摘 要:在日趋复杂的图形处理任务的推动下,GPU 已经演化成为具有众多计算核心、计算能力强大的通用计算设备,并被越来越多地应用于图形处理之外的计算领域。GPU 具有高并行、低能耗和低成本的特点,在数据并行度高的计算任务中,相比与传统的 CPU 平台有着显著的优势。随着 GPU 体系结构的不断演进以及开发平台的逐步完善,GPU 已经进入到高性能计算的主流行列。GPU 通用计算的普及,使个人和小型机构能有机会获得以往昂贵的大型、超级计算机才能提供的计算能力,并一定程度上改变了科学计算领域的格局和编程开发模式。GPU 提供的强大计算能力极大地推动了计算智能的发展,并且已经在深度学习和群体智能优化方法等子领域获得了巨大的成功,更是在图像、语音等领域取得了突破性的进展。随着人工智能技术和方法的不断进步,GPU 将在更多的领域获得更加广泛的应用。

关键词:计算智能;群体智能;演化算法;机器学习;深度学习;图形处理器;GPU 通用计算;异构计算;高性能计算
中图分类号:TP301 **文献标志码**:A **文章编号**:1673-4785(2015)01-0001-11

中文引用格式:丁科,谭营. GPU 通用计算及其在计算智能领域的应用[J]. 智能系统学报, 2015, 10(1): 1-11.
英文引用格式:DING Ke, TAN Ying. A review on general purpose computing on GPUs and its applications in computational intelligence[J]. CAAI Transactions on Intelligent Systems, 2015, 10(1): 1-11.

A review on general purpose computing on GPUs and its applications in computational intelligence

DING Ke[1,2], TAN Ying[1,2]

(1. Key Laboratory of Machine Perception (MOE), Peking University, Beijing 100871, China; 2. School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract: The GPU enjoys the characteristics of high parallelism, low energy consumption and cheap price. Compared with the traditional CPU platform, it is especially suitable for tasks with high data parallelism. GPU computing has come into the mainstream of high performance computation (HPC) due to the emerging of development platforms like CUDA and OpenCL. The GPU's enormous computational power greatly promotes computational intelligence. A great success has been achieved in the fields such as deep learning and swarm intelligence optimization, and several breakthroughs have been seen in image, and speech recognition because of GPU. Though suffering some drawbacks, GPUs provide common people and small institutions with enormous computing power. This has changed the set-up of scientific computing and programming model because it could only be provided by expensive supercomputers. To help researchers in the field of computational intelligence better utilize GPUs, a detailed survey of GPGPU is given in this paper。First, the characteristics and advantages of GPUs against CPUs are presented. Then we briefly review the development of GPU hardware followed by a survey of the evolution of development tools for GPGPU; special attention is drawn to two major platforms, CUDA and OpenCL. We end this paper with our perspectives of the challenges and trends of GPGPU. We point out that embedding and cluster are two major trends for GPGPU and as both academia and industry continue to see increasing progress in artificial intelligence, the GPU will be more widely used in more domains.

Keywords: computational intelligence; swarm intelligence; evolutionary algorithms; machine learning; deep learning; graphics processing unit (GPU); general purpose computing on GPUs; heterogonous computing; high performance computing (HPC)

收稿日期:2014-12-18. **网络出版日期**:2015-01-13.
基金项目:国家自然科学基金资助项目(61375119, 61170057, 60875080).
通信作者:谭营. E-mail:ytan@pku.edu.cn.

GPU 通用计算(general purpose computing on GPUs, GPGPU)是一种利用图形处理器(graphics

processing unit, GPU)解决通用计算任务的异构计算方式。

最初的 GPU 只是功能专一的协处理器,用以帮助中央处理器(central processing unit, CPU)从繁重的图形图像处理任务中解脱出来。为满足人们对图像处理实时性和精确度不断高涨的需求,在过去的十多年间,GPU 的体系结构和制造工艺都取得了长足的进步;GPU 的计算性能及计算精度都得到了大幅提升。如今,GPU 的浮点计算能力早已远超同时期的 CPU。

图像处理任务具有高度并行的特点,这使得 GPU 从诞生之日起,便采用多核心的设计方案。随着 GPU 性能的不断增强和可编程性的日渐提高,GPU 的用途不再局限于传统的图形图像处理。目前,GPU 已经广泛应用于从小到图像解码,大到超级计算机的各种计算领域,进入到了高性能计算的主流行列 。

Owens 等[1-2] 全面综述了 2007 年之前的 GPU 通用计算的进展情况。然而,在过去的几年里,GPU 软硬件条件都发生了极大的发展变化。GPU 高级编程平台,如 CUDA、OpenCL 等的出现,更是使得 GPU 通用计算的面貌发生了深刻的变化。

本文在已有文献综述的基础上,总结了 GPU 通用计算领域最新的软硬件发展及在计算智能领域的应用。文章首先介绍 GPU 通用计算出现的背景,包括传统 CPU 面临的困难与挑战以及 GPU 作为通用计算器件的特点与优势。接着,文章回顾 GPU 通用计算开发工具和平台的发展历程。然后,本文转向 GPU 通用计算的现状,重点介绍主流的 GPU 编程平台——NVIDIA GPU 专属的 CUDA 平台和开放标准的 OpenCL 平台。GPU 通用计算的主要在计算智能领域的重要应用将在之后予以介绍。最后,文章对 GPU 通用计算的目前存在的挑战及未来发展趋势进行展望。

1 GPU 通用计算背景

1.1 多核计算时代

CPU 是计算机系统的核心部件,它为各种任务提供计算能力,是一种通用的计算单元。CPU 的计算能力可以用每秒钟执行指令数(instructions per second,IPS)来衡量,而 IPS 则是由 CPU 的指令吞吐量(instructions per clock,IPC)和时钟频率 F 所共同决定的:

$$\mathrm{IPS} = \mathrm{IPC} \times F \tag{1}$$

为了提高 CPU 的性能,可以从两方面入手:增加指令吞吐量(IPC)和提高时钟频率(F)。

随着制作工艺的不断提高,CPU 芯片的集成度越来越高[3]。一方面,在集成度更高的芯片中,各个单元在物理上更加靠近,从而可以使器件运行在更高的频率上。另一方面,更多的晶体管也使得设计更复杂精细的逻辑控制结构,以提高单位时钟周期的指令吞吐量。

在经过了多年的发展之后,这 2 种策略都遭遇到了巨大的瓶颈。时钟频率的提高使得芯片消耗的功率密度($\mathrm{W/cm^2}$)随之提高,供电和散热即将达到某些基本的物理极限。如图 1 所示,CPU 的时钟频率在经历了长期高速增长后,其提升势头难以为继。

通过复杂的逻辑结构提高指令吞吐量也逐渐达到了收益递减点。现代处理器的内部结构已经十分复杂,并已经能够从指令流中压榨出大量的并行性。通过设计更复杂的结构来提高指令的执行能力变得更加困难[3]。

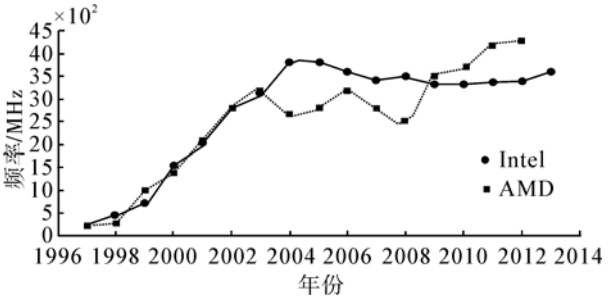


图 1 CPU 时钟频率变化趋势

Fig.1 The tendency of CPU's frequency

为应对上述困难,多核解决方案被提出,以便充分利用晶体管数量的增加,进一步改善 CPU。

事实上,2004 之后,主要的 CPU 厂商如 Intel 和 AMD 等相继转移到多核解决方案,多核计算时代已全面到来。

1.2 GPU 的特点与优势

GPU 最初是为了加速图形处理而设计的专用硬件。图形渲染等任务具有很强的并行性,需要密集的计算与巨大的数据传输带宽。因此,GPU 从诞生之日起便被设计为拥有众多核心,具有高吞吐量的计算。并且,由于 GPU 的专用性,GPU 芯片的晶体管也更多的用于计算,而非控制和缓存(如图 2 所示)。由于 GPU 与 CPU 在设计上存着巨大的差别,GPU 通用计算有着自身的特点与优势。

1)计算能力强大

与 CPU 相比,GPU 具有更加强大的计算能力。在相同时期,GPU 的浮点计算的理论峰值能力相比 CPU 要高出一个数量级[6]。

强大的计算能力使 GPU 在高性能计算领域获得了广泛的应用。在全球最快的超级计算机之中,中国国家超级计算天津中心的天河 1A 号超级计算机和美国橡树岭实验室的 TITAN 号超级计算机,都是依靠装配 GPU 来提升性能,并提供大部分运算能力。GPU 为高性能计算提供了新的巨大的发展潜能。

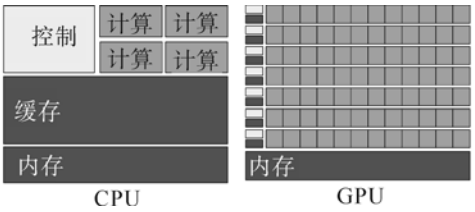


图 2　GPU 的晶体管更多的用于计算单元

Fig.2　GPUs devote more transistors to data processing

2)廉价易得

相比与基于 CPU 的计算机系统,GPU 更加廉价,也更容易为个人和小型机构所利用。

以 NVIDIA 的旗舰版 GPU TITAN 为例。TITAN GPU 价格不足一万元人民币,共有 2 688 个计算核心,双精度浮点运算理论峰值计算能力达到 1 500 GFLP/s,额定功耗约为 250 W。作为对比,北京大学理论与应用地球物理研究所的集群,由 1 个服务器(Dell PowerEdge R710)、46 个计算结点(Dell PowerEdge M610)、3 个刀片机箱(Dell PowerEdge M1000)组成。每个计算结点有 2 颗四核 Intel Xeon E5520(2.26 GHz, Nehalem-EP 架构)处理器,共 368 个计算核心,理论计算峰值 3 327 GFLP/s,满载功耗约为 12 kW。

可以看到,在提供相同数量级的计算能力的情况下,GPU 非常的廉价,也易于装配(只需要一台能够支持独立显卡的 PC),个人和小型实验室都有条件利用。而集群等传统高性能计算设备,不但价格昂贵,对场地、供电和维护等有较高的要求。可以说,GPU 为 PC 提供了小型集群的计算能力[7]。

3)开发灵活

现代的 GPU 都采用统一渲染架构,具有很强的可编程性。GPU 编程的软件开发平台也趋于成熟,CUDA、OpenCL 等工具的推出,使开发者能够使用高级编程语言为 GPU 开发通用计算程序。现在主流的 GPU 也对单精度和双精度浮点运算提供了满足 IEEE 标准的完全支持。对于 GPU 编程,我们会在下面的章节详细介绍。

2　GPU 通用计算编程平台

最初的 GPU 功能固定,只能执行图形流水线中的特定任务,缺乏可编程性。此时,为了利用 GPU 进行通用计算,必须将计算任务封装成图形处理任务,然后交由 GPU 完成。这可以通过底层的汇编命令(低级着色语言)编程实现。更为高效的,通常我们可以通过诸如 OpenGL、Cg 等图形编程接口,利用高级着色语言(shading language, SL)进行编程。

但即使利用高级着色语言,这种方式也还存在诸多不便。首先,开发者需要熟悉图形处理的流程,掌握一系列图形处理接口,并且需要以与任务完成完全无关的一套图形处理思路来考虑问题的编程求解。其次,图形接口是专用的程序接口,缺乏通用计算所需要的灵活性,开发者必须面临如何将问题重新表述成图形处理问题的挑战。

为了简化 GPU 编程,具有更高层硬件层次的编程模式被相继提出。GPU 的统一着色器被抽象为与具体硬件无关的、能够以特定方式并行执行线程的流处理器单元。GPU 的内存、纹理缓存等存储单元,被抽象为具有层级结构的存储体系。经过抽象后,可以在流处理模式下,利用高级语言为 GPU 编写程序,完成通用的计算任务。

本节首先介绍与 GPU 通用计算密切相关的流处理模式。之后,转向介绍基于流行的 GPU 编程语言和工具。

2.1　流处理模式

流处理模式(stream programming model)是一种类似于单指令多数据(SIMD)的编程模式。

通过将数据组织成数据流,将计算表达成作用于数据流的核函数的方式,流处理能够暴露程序中内在的并行性。通过流处理模式,程序开发者能够更容易地利用 GPU、FGPA 等计算硬件的并行能力而不用显式地处理空间分配、同步和计算单元间的通信等问题。对于数据并行性良好的应用,流处理能够取得良好的性能[8]。

如图 3 所示,流处理与 SIMD 最大的不同之处在于,SIMD 是以指令为操作单位的,而流处理模式则是以指令序列(核函数)为操作单元。数据流(输入流)与核函数一同进入流处理器,流中每个数据元素同时被核函数执行,从而完成指定的操作,得到输出数据(输出流)。

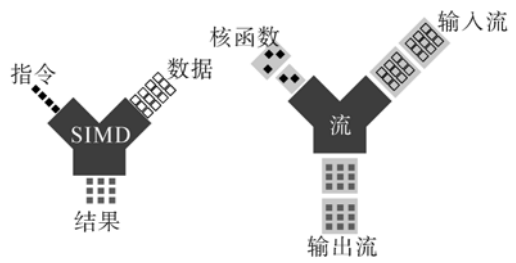


图3 流处理与SIMD

Fig.3 Stream processing and SIMD

流(stream)和核函数(kernel)是流处理模式中最重要的2个概念。流是一个数据集合,包含待处理的数据元素,并且这些数据的处理方式是相同或相似的。核函数是由若干操作(指令)组成的操作序列,它决定着要对每个数据做何种操作和处理。对GPU平台而言,纹理单元充当了流的角色,待处理数据需要作为2维的纹理信息被流处理器读取和操作。许多问题也可以很自然的映射为2维的纹理,如矩阵代数、物理仿真等等;而可编程着色器则起着核函数的作用。

2.2 Brook

Brook是一种基于标准C语言的流编程语言。它旨在基于流编程模式,对图形API进行封装,将GPU视为能够进行并行计算的协处理器,从而简化GPU通用计算的编程[8]。

Brook使人们看到了将GPU用于通用计算的巨大潜力,对GPU通用计算的发展有着深远的影响。Brook将GPU封装为通用的并行处理器的思想及语言规范更是为现在主流的GPU编程平台CUDA和OpenCL所借鉴,而这两者正是现在GPU编程的主流开发平台。

2.3 OpenACC

OpenACC是一个异构并行计算标准。与OpenMP类似,OpenACC定义了一套编译器指令,这些指令应用于C、C++和Fortran代码,以指导编译器将指定的循环或代码块,从主机CPU转移到相连的加速设备(例如GPU)上进行运算。

简言之,OpenACC提供了一种简洁的利用并行硬件的开发方式。目前,OpenACC已经在多种操作系统下实现,并支持NVIDIA、AMD和Intel的多种硬件计算设备[21]。未来,OpenACC可能会并入OpenMP标准,形成一个同时支持多核与众核并行编程的统一标准。

2.4 C++AMP

C++ accelerated massive parallelism(C++ AMP)是一个由微软倡导的开放标准,旨在直接利用C++语言实现数据的并行化编程[9]。

相比于常规的C++代码,C++ AMP只需要按规定格式对数据进行绑定,然后应用并行原语启动数据操作(核函数)的运行。因此,C++ AMP完全隐藏了硬件的细节。开发者只需要预先绑定数组和数组元素上的操作,并行化由C++ AMP负责完成。C++ AMP能够根据实际的硬件情况(GPU或CPU,计算核心算数等),决定并行化策略。当实际硬件不支持并行化时,代码还可以串行执行。

2.5 高级编程平台

在OpenACC和C++ AMP这2种异构编程模式下,硬件对开发者而言完全是透明的。一方面,这简化了并行程序的开发——串行代码只需要少量的改动,便可以利用GPU等多核计算设备的并行计算能力。但这种简化并非没有代价,它使得并行化的过程不能被程序员显示地加以控制。而在很多情况下,根据硬件和程序的具体情况,对负载和并行粒度进行调整,对于充分利用计算设备能力、提高程序性能是至关重要的。

正如前文所述,CUDA和OpenCL是当前基于GPU的异构编程的主流平台。它们的共同特点是,既对硬件的计算和存储单元进行了很好的抽象,同时又对计算设备的计算核心数量这一重要参数,加以适度地暴露,并且对并行粒度和并行方式也给予了更大的控制可能。这样便对完全隐藏硬件这一极端情况的进行了良好的折衷。同时,这2种平台为GPU通用提供了从编程到调试再到性能测试的完整解决方案。

3 统一计算设备架构

统一计算设备架构(compute unified device architecture,CUDA)是由NVIDIA公司于2006年11月发布的基于NVIDIA GPU的并行程序开发架构[10]。它包括一个SDK(NVCC编译器、调试器、函数库等)、一套API,以及添加了少量扩展的C/C++语言(CUDA C/C++)。由它开发的程序能够在所有支持CUDA的NVIDIA GPU上运行。CUDA的推出大大简化了对GPU进行并行编程的难度。图3示意了NVIDIA GPU的主要计算和存储部件。接合图3,对CUDA开发平台的核心逐一进行简要介绍。

3.1 核函数

CUDA允许使用C/C++为CUDA计算核心

(CUDA Core,在较早的文档中也称了 Streaming Processor (SP),即流处理器)编写程序——“核函数”(kernel)。

不同于常规的 C/C++函数,当核函数被调用后,它会被 N 个不同的 CUDA 线程(thread)并行的执行 N 次。执行核函数的每个线程都被赋予一个一的线程索引 ID。

3.2　线程的层级结构

如图 4 所示,CUDA 中的线程具有层次结构。一定数量线程组成线程块(block);线程块再组成更大的单位线程格(grid)。

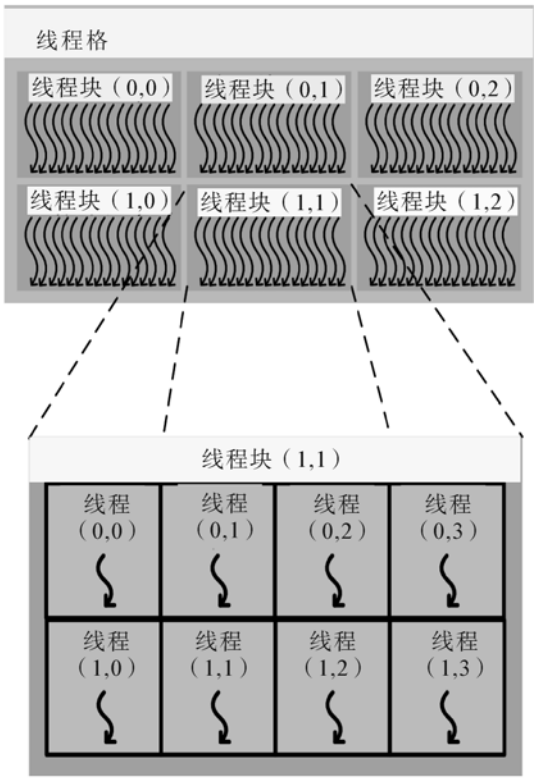


图 4　线程的层级结构

Fig.4　Hierarchy of threads

核函数启动后,每个线程块(thread block)被分配给某个流多处理器(streaming multiprocessor, SM)上执行。线程块执行的先后顺序是不确定的,它们必须相互独立地被执行。即不论它们按何种次序,也不论是按并行还是串行方式,最后的执行结果都应该一致。

不同流多处理器上的线程是相互独立的,相同流多处理器上的线程则是阻塞式的[10]。前者意味着,不同流多处理器上的线程由不同的处理器以任意先后顺序执行;后者意味着,当一个活跃线程受到阻塞(如执行延迟较大的数据读写操作)时,流多处理器可以切换执行其他的线程,避免浪费宝贵的计算资源。

3.3　内存层级结构

同基于 CPU 的常规计算机体系类似,GPU 的存储体系也具有层次性:容量大的存储器,远离 GPU,读写也延迟大;延迟小的存储单元容量小,位置靠近 GPU。

如图 5,CUDA 程序在运行过程中,线程能够从不同的内存空间中读取数据。每个线程都有仅本线程可见的局部内存(local memory)。局部空间一般是片上的寄存器,可以几乎无延迟的读写。每个线程块分配有一块共享内存(shared memory),块中的所有线程都可以读写共享内存的数据。共享内存靠近 GPU,虽然不如寄存器读写速度快,但访问延迟也非常小。离 GPU 最远的是大容量的全局内存。所有线程都可以访问全局内存,但读写延迟非常大,大约要比共享内存高出一个数量级。

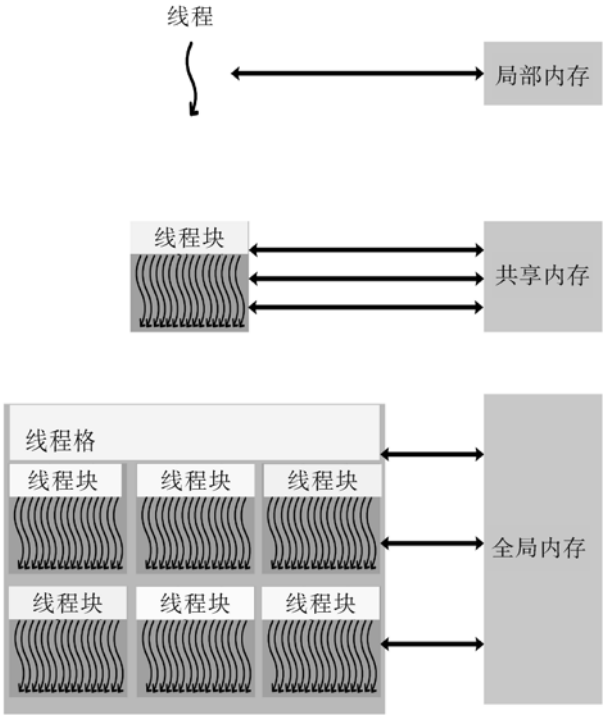


图 5　线程的层级结构

Fig.5　Hierarchy of threads

表 1 对比了在 NVIDIA GeForce 560 Ti GPU 中,不同层级内存的数据传输率。

表 1　GPU 内存数据传输率(560 Ti,PCIe 2.0 x16)

Table 1　GPU memory bandwidth (560 Ti, PCIe 2.0 x16)

	内存/(GB·s^{-1})	全局内存/(GB·s^{-1})	共享内存/(GB·s^{-1})	局部内存
带宽	16	128.26	1 024	无延迟

除以上介绍的几种存储空间外,还存在 2 种对

所有线程可见的只读存储空间——常量内存(constant memory)和纹理内存(texture memory)。常量内存和纹理内存的访问延迟同全局内存相同,但前者配备有缓存(cache),在数据局部性较好时,访问延迟要小于全局内存(参见图 6)。

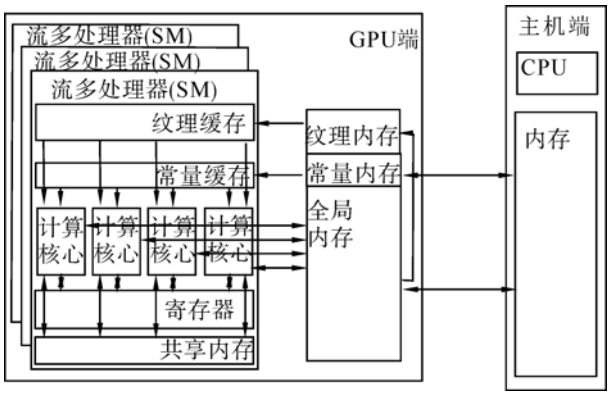


图 6　NVIDIA GPU 主要结构简图

Fig.6　Diagram of NVIDIA GPU structure

同一线程块内的线程可以通过共享内存(shared memory)(块中每个都可以读写共享内存中的数据)和同步机制来相互协作。具体而言,程序员可以在核函数内调用内建函数来设置同步点,以使线程执行到该处后,停下来等待,直到同一块中的所有线程均达到同步点。

3.4　函数库

除了提供以 C/C++为基础语言的高级编程平台,CUDA SDK 还提供了许多函数库,高效实现常见操作,提高开发速度。例如,矩阵计算的 cuBLAS 和 cuSPARSE, 实施快速傅里叶变换的 cuFFT,用于批量生成随机数的 cuRAND,以及用于加速人工神经网络的 cuDNN[12-13]等。除了函数库外,还有大量第三方开源或商业函数库可供选择。这些函数库涵盖基本代数运算、数值计算、统计、图像处理、机器视觉、视频编码解码、GIS 等领域。这些函数的数量和范围也在不断的增加,最新进展可以参见文献[11]。

3.5　调试与性能分析

调试和优化是编程开发重要的环节,也是 GPU 编程的一个难点。为简化调试和优化流程,提高开发效率,CUDA 提供了一系列软件工具。

Parallel Nsight 是集调试和性能测试于一体的编程辅助工具。Nsight 可以集成在 Visual Studio(Windows 下)和 Eclipse(Linux 下),实现交互式调试。在 Linux 环境下,开发者还可以在 cuda-gdb 在命令下进行调试,或将 cuda-gdb 同 IDE 配合使用。此外,CUDA SDK 还提供了 cuda-memcheck 程序,检查内存越界访问错误。

除了上面提到的 Nsight 可以用于性能分析外,CUDA SDK 还提供 Visual Profiler 工具,能够以可视化的方式展示程序各部分的运行时间,方便寻找性能瓶颈,以便有针对性地进行优化。

4　开放计算语言

开放计算语言(open computing language, OpenCL)是一个完全开放和免费的异构计算编程标准,它由非盈利性技术组织 Khronos Group 负责维护。OpenCL 提供了一种跨平台、硬件无关的并行计算解决方案。由它编写的程序不仅能够在多核 CPU、APU 及 GPU 上运行,还能运行在单核 CPU 甚至 DSP 和 FPGA 等硬件设备上。OpenCL 获得了主要硬件厂商 Intel、AMD 和 NVIDIA 的积极支持。

OpenCL 的目标是提出一种跨硬件、跨平台的异构编程框架标准。OpenCL 规范主要由平台模型、执行模型、内存模型和编程模型[13]4 部分组成,它在硬件的抽象描述的基础上,制定了异构的并行编程模式。在制定 OpenCL 标准时,Khronos Group 大量借鉴了 CUDA[14],因此,2 种并行解决方案有很多相似之处,其中,大部分关键概念都可以一一对应。

4.1　设备的层级模型

OpenCL 定义了具有多级结构的异构并行体系模型。如图 7 所示,这一体系包括一个主机(host),多个设备(device);每个设备中包括若干计算单元(compute unit, CU)。计算单元对应于 AMD GPU 的计算单元(CU)或 CUDA 中的流多处理器(SM),但它也可以是 CPU 核心,还可以是 DSP 和 FPGA 等运算设备。计算单元又由若干处理单元(processing element, PE)组成。图 8 给出了内存模型。

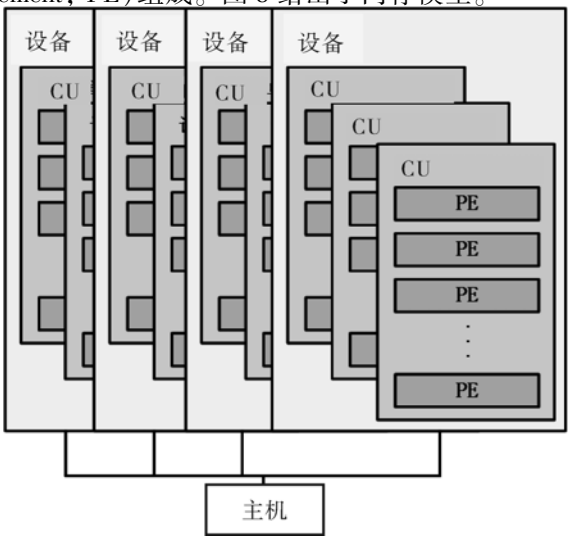


图 7　OpenCL 设备模型

Fig.7　OpenCL device model

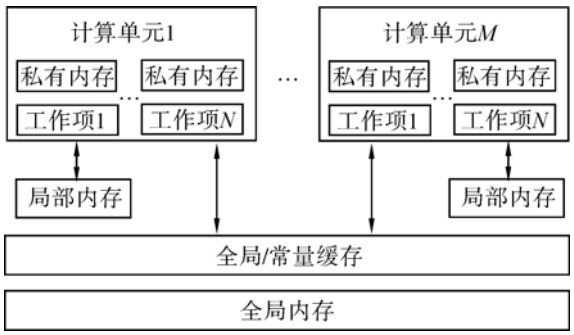


图 8　OpenCL 内存模型

Fig.8　OpenCL memory model

如图 8 所示,OpenCL 规范还定义了内存的层级结构和可见性:远离计算单元的全局内存(global memory)和常量内存(constant memory)是全局可见的,片上的局部内存(local memory)、计算单元内部的私有内存(private memory)则是局部可见的。

4.2　数据并行模型

OpenCL 定义了程序的执行模式。图 9 给出了 OpenCL 抽象的数据并行模型。在 OpenCL 的概念中,执行单元也同是以层级结构进行组织的:工作项(work item)组成工作组(work group),工作组进而构成索引空间——NDRange。OpenCL 的工作项,相当于 CUDA 的线程,是核函数执行的最小单元;工作组对应于 CUDA 的线程块,工作组内的项有更加方便的数据共享和同步机制,便于彼此协作。工作组又被置于更大的索引空间 NDRange 中,它发挥着 CUDA 中线程格的作用,为工作组和工作项提供全局索引。

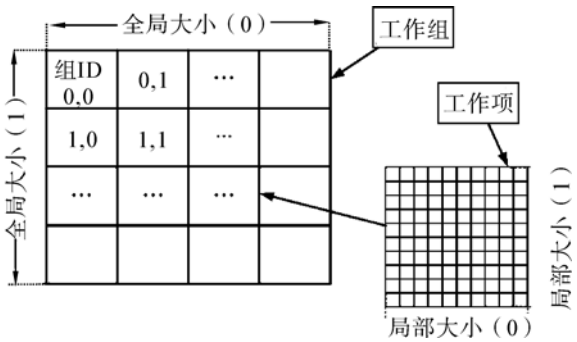


图 9　OpenCL 并行执行模型

Fig.9　OpenCL execution model

同 CUDA 一样,工作项除了拥有在工作组内的坐标外,也具有工作组内的唯一的索引和唯一的全局索引;工作组也有着类似的索引结构。

4.3　设备管理与核函数启动

OpenCL 的核函数的基本结构同 CUDA 完全相同,只是在具体编写时所使用的关键字不同。

OpenCL 的核函数启动后,每个工作项都会执行核函数中指令,这一点也与 CUDA 相同。由于 OpenCL 是与具体硬件无关的,因此,核函数的启动相对 CUDA 要复杂一些。(CUDA 中,复杂的维护工作由运行时负责,用户也可以通过 Driver API 来显示的管理和查询。由于 CUDA 是针对专用硬件平台,因此,一般而言没有这样做的必要。)

如图 10,OpenCL 首先需要创建一个上下文(context)来管理设备,不同类型的硬件对应不同类型的上下文。上下文对硬件设备进行了封装,为主机调用设备的提供了统一的接口。主机启动核函数后,核函数被放在一个命令队列(CMD queue)中,等待执行。当设备完成一个核函数的执行后,会取出队列中的第一个核函数继续执行。关于 OpenCL 更详细的说明,可以参考[13,15];关于基于 AMD GPU 的编程开发,可以参见[16]。

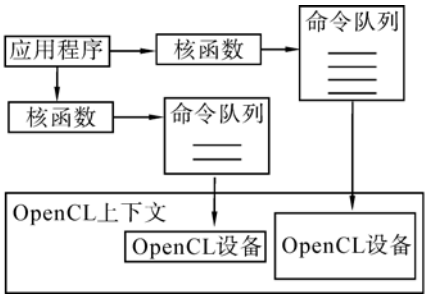


图 10　OpenCL 通过命令队列和上下文管理核函数执行

Fig. 10　OpenCL manages the kernels through command queue and context

4.4　函数库

OpenCL 的生态系统也提供了大量的函数库,方便开发者利用。开源的数学函数库 clMath 实现了快速傅里叶变换(FFT)和基础线性代数全程库(BLAS)。其中,FFT 支持 1~3 维的实数和复数变换;BLAS 支持全部 1~3 级别的代数运算。在 clMath 的基础上,MAGMA 提供了矩阵的 LU、QR 和 Chokesky 分解、特征值计算等重要的矩阵操作的 OpenCL 实现。

4.5　调试和性能工具

目前已有许多优秀的工具,用于 OpenCL 程序的调试和优化。AMD APP SDK 中的 CodeXL 包含了一整套软件工具,帮助开发者最大限度地发掘 GPU 的计算潜力。CodeXL 包括功能强大的调试器,能够对 CPU 和 GPU 端的代码进行细致的监测,同时还能对 OpenCL 的 kernel 函数进行表态分析。CodeXL 可以

配合 Visual Studio 使用,也可以单独使用。

5 计算智能领域应用

GPU 通用计算已经进入高性能计算的主流行列,被用于流体模拟、物理仿真、图像和信号处理、数值计算等诸多,并取得了良好的加速效果。文献[1-2]和[17]给出了 GPU 在许多关键领域和问题上的重要应用。本节主要回顾 GPU 在计算智能领域的应用。

计算智能算法一般具备并行性的特点,因此,特别适用利用 GPU 平台进行高效的并行化实现。

5.1 人工神经网络

人工神经网络(artificial neural network, ANN),模仿生物神经网络的结构和功能,通过联结大量的人工神经元进行计算。由于出色的特征学习能力,深度神经网络已经在图像识别、语言识别等领域取得了突破性进展。图 11 展示了深层卷积神经网络在学习图像的低阶和高阶特征上表现出的良好性能[18]。

图 11 通过深度神经网络学习到的低阶和高阶特征[27]

Fig.11 Low level and high level features learned by DNN[27]

随着深度学习[19](deep learning)研究的不断深入,人工神经网络的规模越来越大,计算复杂性不断膨胀。通常,深度神经网络包含上百万甚至上亿个自由参数,需要在海量的数据集上进行学习。因此,加速深度神经网络的训练速度是工程和科学领域的重要研究内容,直接关系到对深度神经网络的研究以及其在实际中的应用。

超级计算机和大规模集群的成本高昂,个人和中小型研究机和企业构难于负担。GPU 通用计算,极大地降低了深度神经网络的研究和应用的门槛。

2012, Krizhevsky 等[20],搭建了一个深度神经网络并用于图像识别。这个神经网络含有 6 000 万个参数,650 000 个神经元。这个神经网络在 ImageNet 测试集(共有 1 000 个类别)上,取得了 17%的 Top 5 分类错误率,取得了突破性的进展。而这个超大型神经网络的训练,正是得益于两块 GPU 的强大计算能力。

Stanford 大学的研究人员提出了基于 GPU 阵列和分布式运算加速神经网络的计算框架[21]。这个由 16 个 GPU 组成的 GPU 阵列,能够训练比 1 000 CPU 的 Google 集群大 6.5 倍规模的深度神经网络。巨大的实用性使得基于 GPU 的人工神经网络受到科研机构和 IT 企业的重视,是 GPU 能用计算研究热点。

5.2 群体智能优化算法

群体智能优化算法是一类基于群体的启发式随机优化算法,在工程和科学领域有着广泛的应用。然而,群体算法优化过程中需要对目标函数进行大量的评估,这也极大的限制了群体算法在某些问题中的应用。群体算法的内在并行性,使得它们可以很好地利用 GPU 多核心、高度并行的特性。

Zhou 等[22]最早利用 GPU 进行 PSO 算法的加速工作,在当时的硬件条件下,取得了 8X 倍的加速比。随后,基于 GPU 的 PSO 变种及多目标群体算也相继被提出[23-24]。

作为典型的基于 GPU 的群体智能优化方法实现,Ding 等[25]提出了基于 GPU 并行加速的烟花算法(GPU-FWA)[26]。图 12 示意了烟花算法基于 GPU 实现的基本流程。

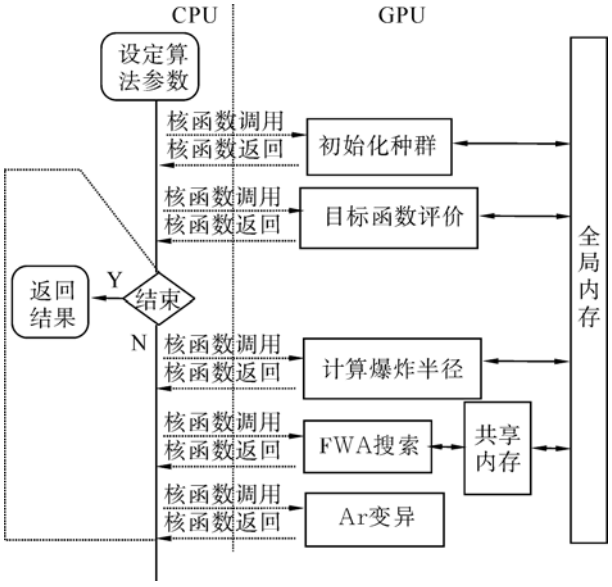


图 12 基于 GPU 的烟花算法实现示意图

Fig.12 Diagram of GPU-FWA

GPU-FWA 充分利用了 GPU 内存层级结构,主要数据通讯被限制在共享内存之间,从而减少了数据传输开销。同时,GPU-FWA 通过一种改进后的

交互机制,进一步提高了运行速度。在对经典基准测试函数上的测试表明,GPU-FWA 相对于传统基于 CPU 的烟花算法,加速近 200 倍。

群体智能优化方法也在大量实际问题获得良好应用。Rymut 等[27]实现的 GPU 版本的基于 PSO 的体追踪算法提速 20~40 倍。Mussi 等[28]实现的基于异步 PSO 的人体追踪算法,将原程序由 CPU 端的几分钟,提升到了几秒钟内完成,实现了跟踪的实时性,提高的算法的实用性。Nobile 等[29]将 GPU 多种群 PSO 应用于生物系统的参数估计,运行时间从 6 h 缩减到 14 min,大大加快了参数选择过程。利用 GPU 加速群体算法,不仅提高了原有应用的速度,而且也扩展了群体优化方法的应用范围。新的基于 GPU 的群体算法的应用在不断的出现,成研究的热点。

6　分析与讨论

6.1　GPU 通用计算的缺点

首先,并非所有的计算任务都适合利用 GPU 进行计算。GPU 适用于存在大量的数据并行性并且数据之间无复杂逻辑依赖的计算任务(典型的如矩阵计算)。对于存在复杂逻辑和随机读写的应用场景,GPU 可能并不是最佳的选择。

其次,GPU 对于算数运算的支持还有待提高。虽然现在主流 GPU 都支持符合 IEEE 标准的浮点运算,但在对双精度浮点的支持上还有待提高。例如,经验表明,对于同样的网络结构和训练方法,使用单精度运算的出错概率要明显高于双精度运算。同时,普通消费级别的 GPU 不支持 ECC 校验,这一点在开发对运算可靠性要求较高的应用时需要特别注意。

同常规的并行编程一样,GPU 的高度并行性也使得程序的调试工作较为困难。虽然已有大量工具可以用来辅助调试,但对于逻辑错误等的调试依然是一项艰巨的挑战。

另外,目前基于 GPU 开发的程序在可移植上还存在不足。CUDA 由于其简易性和相对完善的生态,在 GPGPU 领域处于绝对的领先地位,然而它仅仅支持 NVIDIA 的 GPU。基于开放标准的 OpenCL 的程序虽然具有移植性,但是移植性能情况取决于硬件平台 OpenCL 驱动性能。目前,OpenCL 的流行度正在不断提升,软硬件厂商对 OpenCL 支持力度也在不断增大。在未来几年,GPU 编程的可移植性有望逐渐改善。

6.2　GPU 通用计算发展趋势

由于智能手机、平板电脑和可穿戴设备的普及,以及人们对于语音、视频等多媒体资源需要的不断高涨,GPU 也在这是相对新兴的领域得到广泛应用。未来,GPU 通用计算也很有可能在这些领域发挥重要影响,有望在这一领域带来一场计算变革。而这些非传统的计算设备往往具有低功耗和嵌入式的特点,这也给 GPU 计算带来了新的挑战[30,31]。

互联网的日新月异,大数据的来势汹汹,也为大规模服务器端计算带来了巨大的挑战。将 GPU 用于服务器将是应对挑战的一种潜在的应对方案。由于 GPU 缺乏 CPU 的通用性和灵活性,选择合适的组织方式便尤为重要。一种可能的方案是将 GPU 和 CPU 按照一定的比例组合(例如,一个计算结点配置若干数量的 CPU 和 GPU)。目前,针对 GPU 阵列或分布式集群,已经有一些硬件连接解决方案和中间件支持。然而,这 CPU 和 GPU 之间的通信可能带来潜在的性能瓶颈。混合架构是另一种可能的选择方案。例如,AMD 等硬件厂商提出 HSA(heterogeneous system architecture)概念,试图将 GPU 和 CPU 在物理架构上进行深度整合,通过共享物理内存方式减小通信开销。由于基于 HAS 这一新型体系结构的硬件还未大规模面世,因此其性能还有待在理论和实践中进一步检验。

7　结束语

在计算领域,已经从多核时代跨向众核时代,具有众核架构的 GPU 已经进入高性能计算的主流行列。目前,GPU 通用编程的软硬件平台已经相对成熟,成为了计算领域的重要力量。GPU 能以相对低廉的价格提供巨大的计算能力,从而获得了广泛的应用。GPU 通用计算的普及,使个人和小型机构能有机会获得以往昂贵的大型、超级计算机才能提供的计算能力。可以说,GPU 在一定程度上改变了计算领域的格局和编程开发模式。

GPU 高度并行的特点使得 GPU 能够高效地实现计算智能算法,以应对大规模复杂问题,并且已经在人工神经网络和群体智能优化算法等方面获得了大量的成功应用。GPU 已经成为深度学习领域中事实上的标准计算平台,在图像、语音自然语言处理等领域发挥着不可替代的作用。基于多 GPU 和 GPU 集群的深度网络实现与训练,也是深度学习领域的研究热点。群体智能优化方法以其内在并行性,在 GPU 平台获得了良好的加速效果,从而扩大了群体算法求解问题的规模和范围。随着多目标优化应用和研究的日益广泛,基于 GPU 的多目标群体

智能优化方法将逐步流行,以更好地应对大规模复杂优化问题。

总之,作为通用计算单元, GPU 将以更加多样化的形式活跃在智能计算及其他计算领域。

参考文献:

[1] 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.

[2] OWENS J D, LUEBKE D, GOVINDARAJU N, et al .GPU computing[J]. Proceedings of the IEEE, 2008, 96(5): 879-899.

[3] SUTTER H. The free lunch is over: a fundamental turn toward concurrency in software [J]. Dr. Dobb's Journal, 2005, 30(3): 202-210.

[4] ROSS P E. Why CPU frequency stalled [J]. Spectrum, 2008, 45(4): 72-78.

[5] BORKAR S. Getting gigascale chips: challenges and opportunities in continuing Moore´s Law [J]. Queue, 2003, 1(7): 26-33.

[6] NVIDIA. CUDA C programming guide v6.5[R]. Santa Clara, CA, USA: NVIDIA Corporation, 2014.

[7] JARARWEH Y, JARRAH M, BOUSSELHAM A, et al. GPU-based personal supercomputing[C]//2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies. Amman, 2013: 1-5.

[8] KAPASI U J, RIXNER S, DALLY W J, et al. Programmable stream processors[J]. Computer, 2003, 36(8): 54-62.

[9] BUCK I, FOLEY T, HORN D, et al. Brook for GPUs: stream computing on graphics hardware[J]. ACM Transactions on Graphics, 2004, 23(3): 777-786.

[10] Microsoft. C++ accelerated massive parallelism[Z]. Redmond, WA, USA: Microsoft, 2013.

[11] NVIDIA. CUDA C best practices guide version 4.1[R]. Santa Clara, CA, USA: NVIDIA Corporation, 2012.

[12] NVIDIA. GPU-Accelerated Libraries [OL/EB]. [2015-01-05]. https://developer.nvidia.com/gpu-accelerated-libraries.

[13] JIA Y, SHELHAMER E, DONAHUE J, et al. Caffe: convolutional architecture for fast feature embedding [C]// Proceedings of the ACM International Conference on Multimedia, [s.l.], 2014: 675-678.

[14] GASTER B, HOWES L, KAELI D R,等. OpenCL 异构计算[M]. 北京: 清华大学出版社, 2012: 10-35.

[15] KIRK D B, HWU W W. Programming massively parallel processors: a Hands-on approach[M]. Beijing: Tsinghua University Press, 2010: 205-220.

[16] MUNSHI A, GASTER B, MATTSON T G, et al. OpenCL Programming Guide[M]. Boston: Addison_Wesley Professional, 2011: 63-68.

[17] AMD 上海研发中心. 跨平台的多核与从核编程讲义——OpenCL 的方式[M]. 上海: AMD, 2010: 1-154.

[18] FARBER R. 高性能 CUDA 应用设计与开发[M]. 北京: 机械工业出版社, 2013: 1-49.

[19] ZEILER M, FERGUS R. Visualizing and understanding convolutional networks[C]//Proceedings of the 13th European Conference on Computer Vision. Zurich, Switzerland, 2014: 818-833.

[20] HINTON G, OSINDERO S, WELLING M, et al. Unsupervised discovery of nonlinear structure using contrastive backpropagation[J]. Nature, 2006, 30(4): 725-731.

[21] KRIZHEVSKY A, SUTSKEVER I, HINTON G. Imagenet classification with deep convolutional neural networks [C]//Advances in Neural Information Processing Systems 25. Reno, Nevada, USA, 2012: 1106-1114.

[22] COATES A, HUVAL B, WANG T, et al. Deep learning with COTS HPC systems[C]//Proceedings of the 30th International Conference on Machine Learning. Atlanta, USA, 2013: 1337-1345.

[23] ZHOU Y, TAN Y. GPU-based parallel particle swarm optimization[C]//IEEE Congress on Evolutionary Computation. Trondheim, Norway, 2009: 1493-1500.

[24] ZHOU Y, TAN Y. Particle swarm optimization with triggered mutation and its implementation based on GPU [C]//GECCO'10: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. Portland, Oregon, USA, 2010: 1-8.

[25] ZHOU Y, TAN Y. GPU-based parallel multi-objective particle swarm optimization[J]. International Journal of Artificial Intelligence, 2011, 7(A11): 125-141.

[26] DING K, TAN Y. A GPU-based parallel fireworks algorithm for optimization[C]//GECCO'13: Proceedings of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference. Amsterdam, the Netherlands, 2013: 9-16.

[27] TAN Y, ZHU Y. Fireworks algorithm for optimization [C]//First International Conference of Swarm Intelligence. Beijing, China, 2010: 355-364.

[28] RYMUT B, KWOLEK B. GPU-supported object tracking using adaptive appearance models and particle swarm optimization[C]//International Conference on Computer Vi-

sion and Graphics, Warsaw, Poland, 2010: 227-234.

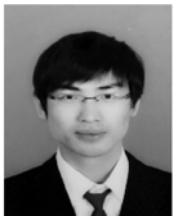
[29] MUSSI L, IVEKOVIC S, CAGNONI S. Markerless articulated human body tracking from multi-view video with GPU-PSO[C]//9th International Conference on Environmental Systems. York, UK, 2010: 97-108.

[30] NOBILE M S, BESOZZI D, CAZZANIGA P, et al. A GPU-based multi-swarm PSO method for parameter estimation in stochastic biological systems exploiting discrete-time target series[C]//10th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Computational Biology. Málaga, Spain, 2012, 7246: 74-85.

[31] MAGHAZEH A, BORDOLOI UD, ELES P, et al. General purpose computing on low-power embedded GPUs: has it come of age[R]. Linköping University Electronic Press, 2013.

[32] HALLMANS D, SANDSTROM K, LINDGREN M, et al. GPGPU for industrial control systems[C]//2013 IEEE 18th Conference on Emerging Technologies Factory Automation. Cagliari, Italy, 2013: 1-4.

作者简介:



丁科,男,1989 年生, 博士研究生,主要研究方向为群体智能、GPU 通用计算、并行编程和机器学习。

谭营,男,1964 年生,教授,博士生导师,主要研究方向为计算智能、群体智能、机器学习、人工免疫系统、智能信息处理及信息安全应用。担任 IJCIPT 主编,IJSIR 副主编,IEEE Trans on Cybernetics 副主编等,IEEE Senior Member, IEEE CIS-ETTC 委员,ICSI 系列会议大会主席。主持国家“863”计划、国家自然科学基金、国际合作交流等科研项目 30 余项。获得 2009 年度国家自然科学二等奖。获国家发明专利授权 3 项,发表学术论文 260 余篇,出版专著 5部。

第 6 届国际群体智能会议和第 2 届金砖国家计算智能大会联合国际会议
The 6th International Conference on Swarm Intelligence in Conjunction with the 2nd BRICS Congress on Computational Intelligence (ICSI-CCI'2015)

June 26 to 29, 2015, Beijing, China

The Sixth International Conference on Swarm Intelligence and the Second BRICS Congress on Computational Intelligence (ICSI-CCI'2015) (http://www.ic-si.org) will be jointly held in Beijing, China, from June 26 to 29, 2015. The theme of the ICSI-CCI'2015 is "SERVING OUR SOCIETY AND LIFE WITH INTELLIGENCE". With the advent of big data analysis and intelligent computing techniques we are facing new challenges to make the information transparent and understandable efficiently. The ICSI-CCI'2015 will provide an excellent opportunity and/or an academic forum for academia and practitioners to present and discuss the latest scientific results and methods, the innovative ideas and advantages in theories, technologies and applications in both swarm intelligence and computational intelligence. The technical program will cover all aspects of swarm intelligence, neural networks, evolutionary computation and fuzzy systems applied to all fields of computer vision, signal processing, machine learning, data mining, robotics, scheduling, game theory, DB, parallel realization, etc.

Prospective authors are formally invited to contribute high-quality papers to ICSI-CCI'2015. Papers presented at ICSI-CCI'2015 will be published by Springer in its prestigious book series of Lecture Notes in Computer Science (LNCS) (indexed by EI, ISTP, DBLP, Scopus, ISI, etc.) and some high-quality papers will be selected for special issues in SCI-indexed International Journals.

Contact Us

E-mail: icsi-cci2015@ ic-si.org

Website: (http://www.ic-si.org)