软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(8):2048−2067 [doi: 10.13328/j.cnki.jos.005060]

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

面向新型处理器的数据密集型计算*

王鹤澎, 王宏志, 李佳宁, 孔欣欣, 李建中, 高 宏

(哈尔滨工业大学 计算机科学与技术系,黑龙江 哈尔滨 150006)

通讯作者: 王宏志, E-mail: wangzh@hit.edu.cn

摘 要: 近年来,随着数据量的不断增大,数据密集型计算任务变得日益繁重.如何能够快速、高效地实现在大规模数据集上的计算,已成为数据密集型计算的主要研究方向.最近几年,研究人员利用新型的硬件处理器对数据密集型计算进行加速处理,并针对不同新型处理器的特点,设计了不同形式的加速处理算法.主要对新型硬件处理器基于数据密集型计算的研究进行了综述.首先概述了新型硬件处理器的特点;然后,分别对新型处理器 FPGA 和 GPU 等硬件进行性能分析,并分析了每种处理器对数据密集型计算的效果;最后提出了进一步的研究方向.

关键词: FPGA;GPU;CPU;数据密集型计算

中图法分类号: TP311

中文引用格式: 王鹤澎,王宏志,李佳宁,孔欣欣,李建中,高宏.面向新型处理器的数据密集型计算.软件学报,2016,27(8):2048−2067. http://www.jos.org.cn/1000-9825/5060.htm

英文引用格式: Wang HP, Wang HZ, Li JN, Kong XX, Li JZ, Gao H. New processor for data-intensive computing. Ruan Jian Xue Bao/Journal of Software, 2016,27(8):2048−2067 (in Chinese). http://www.jos.org.cn/1000-9825/5060.htm

New Processor for Data-Intensive Computing

WANG He-Peng, WANG Hong-Zhi, LI Jia-Ning, KONG Xin-Xin, LI Jian-Zhong, GAO Hong

(Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150006, China)

Abstract: In recent years, with increased data volume, data-intensive computing tasks become increasingly critical. How to efficiently and effectively implement data-intensive computing on large data sets becomes a major research direction for data-intensive computing. Currently, researchers attempt to use new processors to accelerate the data-intensive computing process. Different acceleration approaches could be adopted according to the characteristics of new processors. In this paper, the new processors, well as the algorithms, for data-intensive computing research are surveyed. First, the new features of processors are reviewed. Then the capability of each new processors and their performance over data intensive computing are analyzed. Finally, the future research directions are discussed.

Key words: FPGA; GPU; CPU; data-intensive computing

针对急剧上升的大规模数据,传统的数据密集型计算已经无法再在此种场景下适用.对此,大量的学者们对算法进行不断的改进,并提出利用新型的处理器来进行复杂的数据密集型计算.在众多的新型处理器中,FPGA 与 GPU 体现了良好的并行计算等特性,使得复杂的数据密集型计算能够快速、高效地在并行机器集群上快速地进行并行计算.

处理器的主要功能是用于对计算机的指令进行解释以及处理计算机应用中的数据.由于内存空间急速变大,但是处理器的处理效率却无法与其匹配,适用于各类处理不同场景的新型处理器迅速出现,并展现了其高效

* 基金项目: 国家自然科学基金(61472099, 61133002); 国家科技支撑计划(2015BAH10F01)

Foundation item: National Natural Science Foundation of China (61472099, 61133002); National Key Technology Research and Development Program of China (2015BAH10F01)

收稿时间: 2015-11-03; 修改时间: 2015-12-25, 2016-03-03; 采用时间: 2016-03-14; jos 在线出版时间: 2016-04-20

CNKI 网络优先出版: 2016-04-19 15:38:48, http://www.cnki.net/kcms/detail/11.2560.TP.20160419.1538.001.html

的性能.不同的处理器针对不同类型的计算算法以及不同的面向场景,并体现出不同的性能优势.

针对不同处理器,研究人员对不同的任务提出了大量新算法.面向新型处理器的高效数据密集型计算算法研究已成为当前数据管理等领域的热点之一.为了有效地对当前面向新处理器的数据密集型计算技术的新进展进行梳理,本文首先对数据密集型计算、新型处理器背景做出简要的概述(第 1 节).然后,分别在第 2 节和第 3 节分别对新型处理器 FPGA,GPU 进行分析.然后,基于其他硬件平台对大规模数据进行数据密集型计算的综述(第 4 节).针对其特点进行概述,再根据其特性对数据库系统以及机器学习中的数据密集型计算做出相应的改进(第 5 节).

1 研究背景

大量的用户在生活中、网络中持续产生活动记录,使得需要管理和计算的数据以指数形式急速增加.针对这些大规模的数据,如何让其更好地服务用户变得日益重要.数据密集型计算实现数据的获取、存储及管理,对得到的数据集进行分析与计算,通过计算分析得到用户所需的结果.针对大规模的数据集,数据密集型计算也变得更加复杂、困难,计算的效率也随着数据集的增加而降低.所以,如何解决大规模数据集的存储操作与密集计算,成为当前亟待解决的问题.面对该问题,研究人员针对各种数据密集型计算进行了并行处理,将算法划分成子计算模式进行并行计算,从而提高算法执行效率.因此,随着大数据时代的到来,大规模的分布式计算框架在数据密集型计算中变得更加重要,针对海量数据的数据库操作也越来越受到重视.但是随着数据量的增加,算法计算量也变得更加复杂,普通硬件(如 CPU)的计算速度已经无法满足用户需求.学者们开始通过对硬件的改变,使数据密集型计算变得高效.由于 FPGA 具有可重构及可扩展等特点,GPU 具有高并行等特点,十分适用于当今的数据密集型计算,因此,以 FPGA,GPU 为代表的新型的硬件得到了快速、高需求的发展.如何将这些新型处理器有效应用于数据密集型计算,成为现今数据库领域和高性能计算领域关注的热点问题之一.研究人员近年进行了大量的研究工作,提出了一系列技术.本文主要根据新型处理器对数据库上的密集型操作计算以及机器学习方向上的数据密集型计算进行综述.

在数据库上针对大规模的数据集进行数据密集型操作时,随着计算机内存空间不断变大,现代数据库系统能够在内存中对其进行处理.但是随着内存容量的增加,并没有适当地降低内存延迟,从而导致了内存在处理数据较多时延迟过长.因此,数据在内存层之间的移动成为了影响内存数据系统时间效率的瓶颈.以前的设计中,针对一个特定的查询,所有与查询结果相关的数据必须在内存层之间进行移动.针对这种情况,Sam 等人[1]提出了 JAFAR,一种嵌入在 DRAM 中的 Near-Data Processing(NDP)硬件加速器,即利用新型的列存储方案,能够将选择操作推至内存中执行.此外,应用 NDP 加速器来实现其他关系型数据操作,也具有非常广阔的发展空间.近年来,随着 FPGA,GPU 等新型加速器的迅速发展,其优异的性能同时被应用于数据库操作中.

众所周知,FPGA 是以并行计算为主的处理器,GPU 是图像应用领域的最成功的加速器.在其他领域,也存在着大量的加速器应用于各种数据密集型计算中.例如,卷积引擎能够针对图像内核和模板计算进行处理,M7 能够加快数据库查询[2],ASiPEC 能够对 H.264 视频编码器进行加速[3].各种新型处理器能够对复杂计算进行加速处理,大量基于大规模数据集上的数据库操作项目被提出来.例如,利用 FPGA 进行编译(例如 LINQits, Teradata),工业上将 Xeon 服务器和 FPGA 结合起来以实现数据密集型计算(例如 IBM Netezza[4]提出的方案).另外,部分研究人员使用已经存在的加速器(例如网络处理器、GPU 等)来实现针对关系运算的加速.这些项目都能够利用特定的电路来加速数据库查询操作,从而提高在大规模数据集上数据库的操作效率.

根据 FPGA 处理器具有可重构的特性,近几年主要研究将其用于加速数据库系统的查询操作.由于 FPGA 处理器能够动态地进行编译,所以针对不同的查询需求,能够局部动态地进行重构.由于 FPGA 处理器每个时钟周期能够处理更多的任务,所以其能更好地执行并行化的算法,比如 *K*-means、贝叶斯网络以及人工神经网络等一些适用于并行的算法.GPU 处理器由于内在的硬件特性,使其对高通量并行计算有着有效支持.将其与数据库系统进行结合,能够加速数据库的查询处理.并且其并行性能够有效地处理一些计算复杂的数据密集型计算算法,例如机器学习上的 *K*-means、遗传算法、贝叶斯算法等.本文将针对各种处理器其不同的性能,对各种数据

密集型计算的加速方法进行综述.

2 基于 FPGA 的数据密集型计算

2.1 FPGA概述

FPGA,即现场可编程门阵列,是一个可重复编程的逻辑设备,且是一个以并行运算为主的处理器.它能够通过可编程的逻辑元件配置构造成一些基本的逻辑门数字电路,例如与门、或门、非门、异或门以及它们更为复杂的组合排列.通过硬件描述语言,例如 Verilog 或者 VHDL 来产生这些数字电路.FPGA 能为用户提供更灵活的数字电路配置.基于 FPGA 的系统是一个具有高性能、高效率的体系结构.FPGA 具有如下几个关键的特性:FPGA 比 CPU 性能高两三个数量级;它能在 6~12 个月内跟上 CPU 的半导体技术;FPGA 是内置可重置的,它能够被重新加载以符合当前的应用需求.基于 FPGA 的架构,能够被应用于嵌入式系统中.同时,这一技术也正在逐步应用于计算密集型和数据密集型的应用中.基于这些特点,FPGA 在数据库上的操作与机器学习上的效率也有更好的体现.

FPGA 的适用范围很宽,包括通信、视频、信息处理等领域.目前,由于 FPGA 的使用成本较低且能耗比其他主流处理器要低,因此比较广泛地用于工业环境中,使其大规模地进行数据密集型计算.其用于工业大规模使用的优势体现在:(1) FPGA 能够集成整个芯片系统(SoC),在一个硬件平台上集成 IP 和软件堆栈,从而大幅度降低成本;(2) FPGA 有较长的产品生命周期,能够通过硬件移植来延长其工业生命周期,从而达到低成本使用的效果;(3) 可重新编程的性能使其能够在工业开发中,利用不同的设计来适应新硬件功能的发展变化.

2.2 基于FPGA在数据库上的查询操作

有一些利用 FPGA 处理查询操作的静态方法.Andreas 等人[5]提出了针对嵌入式设备使用 FPGA 来加速数据库中查询操作的方案,主要针对文本数据进行数据库的查询操作.通过分析一个即将被执行的查询,即时地会产生针对该查询的硬件加速器,并应用部分动态重分布能够将其装载入 FPGA 上,以便执行随后的查询.对于每个 SQL 查询操作,均会在模板库中建立一个预先合成的比特流实现方案.它的处理速度和高端数据库服务器速度一样快,并会产生更少的额外能量消耗.应用该技术,能够确定哪个子查询需要使用硬件加速,哪个子查询不值得使用硬件加速.

Glacier[6]基于 FPGA 在硬件电路上利用编译器实现流式查询的方法.当给定一个查询,编译器把必要的模块实例化到数字逻辑电路上.系统架构如图 1 所示.Glacier 提供接口组件来接受中央处理器(main memory, CPU)以及网络(network)的数据,并将结果流返回到网络或中央处理器.如图 1 所示的配置,FPGA 直接与物理网络接口进行通信,为网络端提供了低延迟的性能.FPGA 内部构造实现了 UDP/IP 网络控制器组件.接受数据流后,将数据从网络直接送入由硬件实现的数据库查询部分.最后,将最终查询结果发送到应用系统中.

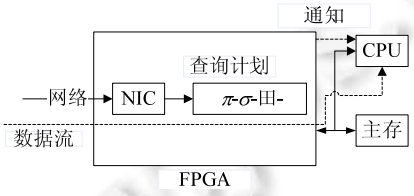


Fig.1 System architecture diagram[6]

图 1 系统架构图[6]

Glacier 支持任意的代数操作组合,因此具有较高的灵活性.但是 Glacier 上的电路只能对已知的查询进行处理,对于新的查询,需要重新生成电路.因此,提出了一种新的方法,动态重分配方法[7],使 FPGA 变得更加灵活、可扩展.该方法提供了一种基于 FPGA 的加速 SQL 查询的局部动态重分配方法,通过加速查询,使得大型数据库系统能够达到非常高的吞吐量.利用局部动态重构方法,能够更加灵活地构建架构,使得基于 FPGA 硬件来扩展新

的操作或SQL架构的花销更低.这种查询方法允许用户对其进行部分配置,这样,对于新的查询到来时不需要全部重新配置.

虽然文献[7]所提供方法的可扩展性较好,但是这种方法只解决了投影和选择操作.因此,研究人员提出了智能存储引擎 Ibex[8],它实现了除了连接以外的所有操作.它通过利用智能存储引擎,能够减少对于复杂查询操作的负载.除了性能获得提升以外,通过利用FPGA实现负载引擎还减少了能量消耗.Ibex作为一个混合引擎,利用专用的硬件实现了普通硬件引擎无法处理的评估SQL表达式的line-rate和软件任务回退.

Scofield 等人在文献[9]中指出:XTREMEDATA 作为活跃于数据仓库和分析型应用领域的新秀,它设计了dbX 专门用来分析深层次需求,以解决用户需求的应用环境.dbX 的主要目标是快速加载数据,对大规模数据进行高效查询.dbX 目的是使用 FPGA 加速软件中部分数据库函数的执行,运行时对数据执行动态重分布以及平衡结点间的负载,应用相关技术减少数据倾斜等.

Sukhwani 等人[10]提出了一种加速器,能够适用于各种不同的查询操作,利用基于FPGA的加速引擎在数据库上进行分析查询操作.加速器通过对查询进行划分来执行不同的查询操作,从而对 FPGA 可以不进行重新配置.然后,又基于 FPGA 加速了投影和排序两种数据库操作[11].相比于原始软件,提高了排序操作的速度,并行实现了投影操作,降低了投影操作的计算开销,并针对联机事务处理系统减轻了计算负载.

Yoshimi 等人[12]利用闪存存储技术,基于FPGA来加速联机分析处理中用户自定义函数的聚合操作,实现了一个加速器模型,能够将数据从闪存存储器到DRAM进行聚合.除了对数据进行适当的分配及分割,加速器模块还操作在FPGA芯片上的数据流.

2.3 基于FPGA的*K*-means算法

K-means 算法具有收敛速度快、易于实现的特点,广泛应用于信息检索、图像处理及模式识别等领域,但是在一些针对大规模数据应用中(如物理模拟等)表现不佳.Choi 等人[13]设计实现了基于 FPGA 加速计算的*K*-means 聚类计算框架,通过利用 map-reduce 模型,在多个FPGA上自主实行map和reduce函数,并在多FPGAs环境上对计算和I/O性能进行了系统级的权衡.该方法主要针对在多维的文本数据上进行大规模数据的聚类操作.他们实现的 map-reduce 编程模型,能够在多个 FPGAs 上加速运行 *K*-means 算法;并且对比之前的研究,该模型具有通用性,能够在额外的FPGA上进行大量的扩展.

图2展示了文献[13]中的一个异构计算集群.该集群包含一组同类的计算节点,每一个计算节点都具有一个FPGA加速器.除了与主机有一个基本的通信和内存获取方式,FPGAs还利用专用的通信网络进行相互连接.

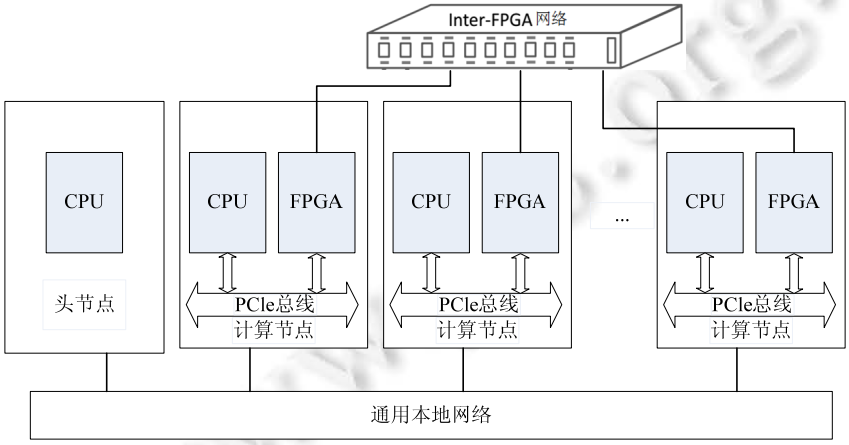


Fig.2 *K*-Means cluster structure diagram[13]

图2 *K*-Means集群架构图[13]

这种简单的集群架构具有可扩展性,并且能够对现有的系统向后兼容,因此,该集群能够对已存在的分布式

软件系统进行终端到终端的性能对比.

该模型的单节点架构图如图 3 所示,在执行 *K*-means 算法的过程中,软件和硬件系统中数据流持续从主机传输到 FPGA 上.图中 Data Manager 模块部署于每台计算节点中,负责在通用文件系统中的分区数据文件中检索数据.软件 Streamer 用于和 Data Manager 交互,当 Data Manager 有一批输入数据时,在 PCIe 的帮助下,Streamer 负责将这批数据接受并传递到 FPGA 中.PCIe 将数据从用户内存空间拷贝到内核内存空间.然后,FPGA 在内核内存空间中执行 DMA 读取操作.

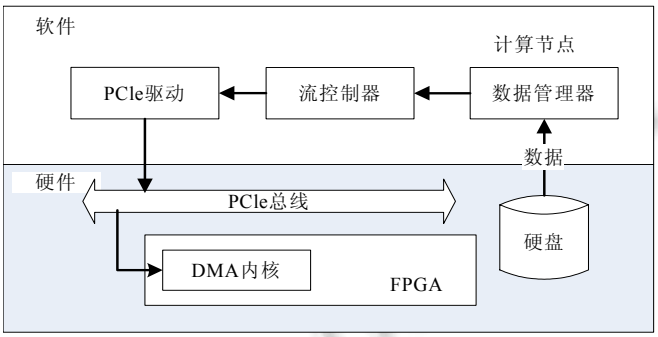


Fig.3 *K*-Means single node architecture diagram[13]

图 3 *K*-Means 单一节点架构图[13]

针对大规模的图像数据,文献[14,15]通过利用基于 FPGA 优化的 *K*-means 算法进行距离计算,加速处理了多光谱和超光谱图像数据.同样地,基于 FPGA 的 *K*-means 算法在实时图像集群[16,17]中十分有效.Winterstein 等人[18]实现了一种基于 FPGA 的变形 *K*-means 聚类算法,利用 *kd* 树型结构来剪枝搜索空间,减少计算负担,提高对图像数据的聚类速度.文献[19]提出一种基于 FPGA 参数化实现 *K*-means 聚类算法的方法,比之前的方法[20]更加有效.之前的方法[20]只能对一维数据、8 个集群点进行实现,而现在的方法[19]是基于 FPGA 动态局部重构的方法,利用 3 种新型的 *K*-means 聚类算法在多维数据上实现:第 1 种是利用不同的距离度量方法来重构算法的距离核心,第 2 种方法是利用内部配置访问通道(ICAP)来重构 *K*-means 的内核,最后一种是利用内部配置访问通道来重新配置多个 *K*-means 内核.

2.4 基于FPGA的遗传算法

遗传算法是一种基于随机特征的自适应学习策略的优化方法.由于其函数的设计不固定,例如有时函数需要进行优化,因此硬件平台必须具有可重编程性.FPGA 的可重编程性便满足了该点需求.遗传算法广泛应用于数值分析、图像处理、模式识别、非线性优化问题等领域.针对大规模的文本数据、图数据,可通过将数据转换成二进制输入数据,利用遗传算法来解决问题.遗传程序被许多研究人员用于合成组合电路[21],但是由于他们都未能解决电路 4 个或 6 个以上变量的问题,使得这些方法都存在一个严重的可扩展性问题.对此,César 等人[22]提出了一种基于 FPGA 集群的高性能可重构计算框架,针对文本数据,实现了并行遗传程序的布尔合成,使其能够扩展到 12 个变量.

针对大规模的文本数据,早期的遗传算法硬件实现的主要缺点就是缺乏参数可编程性以及缺少对多适应度函数的支持.对此,Pradeep 等人[23]设计了一种名为 IP Core 的方法,通过利用 FPGA 的高逻辑密度性能,设计了遗传算法引擎来解决这些问题.该方法能够自定义遗传算法的种群规模、生成数目、交叉率和突变率、随机数生成的种子以及适应度函数.由于 FPGA 具有动态可重构的特性,因此,François 等人[24]也将其用于无人机中,利用遗传算法对路径进行规划,利用 FPGA 能够对路径进行实时编译规划来高效地处理图数据.表 1 将传统的方法与使用 FPGA 的遗传算法进行了各个阶段的效率比较,实验结果是多次实验后的平均值,PC 环境是 CPU 为 2.8GHz 奔腾 4 处理器,FPGA 为 Virtex-II Pro 的 100MHz 时钟系统.通过表 1 可以看出,利用文献[24]基于 FPGA

的方法可以使遗传算法速度大幅提升.由于评估阶段在该文献中仍处于改进阶段,因此用 N/A 来代替,而速度提高 1.2 倍是由于排序后的种群比未排序的种群需要更少的时间来查找适应度.

Table 1 Table of effect of genetic algorithm[24]

表 1 遗传算法效果表[24]

GA 阶段	初始基于 PC 的 GA	用 FPGA 改进 GA	速度提升比
选择和交叉	94ms	8.85μs	10 000×
突变	2ms	18μs[a]	111×
种群更新	30ms	600ns[b]	50 000×
评估	60s/50s	N/A[c]	1.2×[d]

针对在大规模文本数据上的遗传算法,Aporntewan 等人[25]提出了一种利用 Verilog 语言在 FPGA 上实现的紧凑型遗传算法,并由随机数生成器、概率寄存器、比较器、缓冲器以及适应度函数计算器这 5 种模块组成,每 3 个时钟周期迭代一次遗传算法.但是由于这种方法需要占用大量的内存与复杂的计算,因此,Krupesh 等人[26]提出了一种改进后的基于 FPGA 的紧凑型遗传算法,通过只存储种群向量来替代整个种群,使其需要更少的内存来进行更简单的计算.

在解决真实世界应用问题时,由于基于FPGA实现的遗传算法比基于CPU实现的遗传算法更高效[23],因此,许多研究人员开始采用 FPGA 来加速实现遗传算法[25−30].虽然基于 FPGA 实现的遗传算法被广泛地应用,但是需要大量定制遗传算法的硬件架构,因此,Guo 等人[31]提出了一种自动框架,基于 FPGA,能够使用户自定义硬件架构来实现通用的遗传算法.对于只基于软件实现的遗传算法,在需要实时反馈的场景下延迟过高[27],而硬件实现遗传算法的并行化效果更好[23],因此,Nambiar 等人[32]利用可重构的硬件实现遗传算法,并通过协同设计将软件和硬件结合加速算法运行.

由于人工神经网络需要知道拓扑结构和每个突触连接的初始权重,但是这两个参数的计算又过于繁重,因此,Fe 等人[33]利用遗传算法在 FPGA 嵌入式系统上加速实现了人工神经网络的训练.

Tuncer 等人[34]提出了一种基于 FPGA 的遗传算法,并将其应用于移动机器人的路径规划,将复杂的适应度函数用硬件核心实现,其他计算利用软件处理器实现.Vavouras 等人[35]主要提出了利用 Virtex-II Pro FPGA 在 XUPV2P 平台上实现遗传算法,主要优化了适应度函数.

2.5 基于FPGA的贝叶斯网络

由于 FPGA 具有重构特性,因此它在贝叶斯网络上体现的性能比软件更好,比硬件[36]更加灵活.FPGA 通过修改配置位可重编程多次,使得它更加灵活地支持更多的应用[37,38].文献[39]提出了一种在FPGA上基于粒子群优化的贝叶斯网络学习方法.利用基于 FPGA 的解决方案比软件实现的贝叶斯网络更高效,在每个集群节点上每秒生成适应度评估的数量提高约 2.6 倍.该方法没有适应度函数的限制,但是限制了条件概率表的结构.

由于最近使用拥有高维度特征空间的高吞吐率数据(例如图像、微矩阵等)场景大量增加,急需一种能够高效处理这些数据的方法.文献[40]提出了一种基于随机判别原理的贝叶斯分类器方法,并使用基于FPGA硬件的并行计算来快速处理算法的复杂计算.

基于 FPGA 实现的计算硬件架构[40]如图 4 所示,该硬件由控制模块、训练集内存、对比模块、求和模块以及预测模块组成.

- 控制模块给每个模块发出控制信号,并从训练集内存中读取数据;
- 训练集内存类似于 ROM,存储包含分类标签的训练集数据;
- 对比模块由多个比较单元组成,其个数等于子空间的个数,并且每个比较单元包含一个比较器和一个随机掩码模式发生器,比较器能够确定输入的特征是否匹配;
- 如果测试数据和训练数据的某属性发生匹配,该属性的总值在求和模块上进行增加操作;
- 预测模块利用该文献提出的计算方法来估算输入数据的分类标签.

PSNB 代表文献[40]对应的方法,表 2 中显示了在不同数据集上利用不同分类方法的得到的准确率.

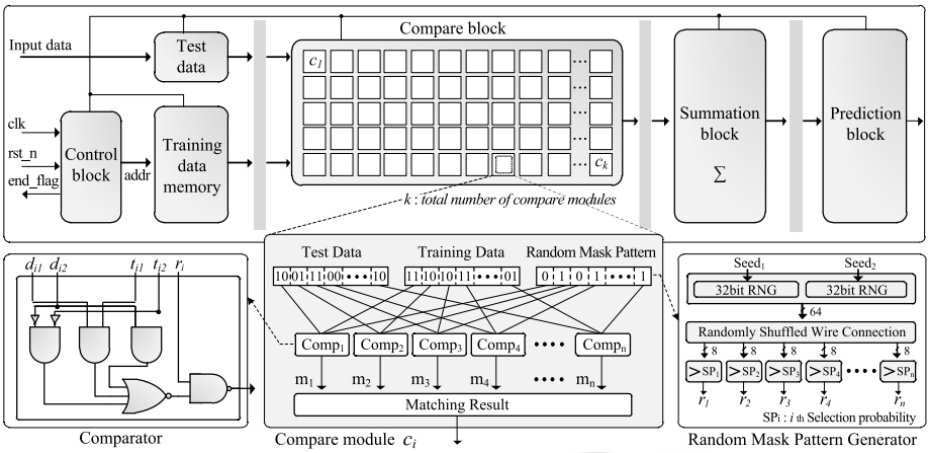


Fig.4 Computing hardware implementation diagram[40]

图 4 计算硬件实现图[40]

Table 2 Classification effect comparison table[40] (%)

表 2 分类效果对比表[40] (%)

数据集	PSNB	kNN	SVM	LDA
Leukemia	97.06	94.12	94.12	91.18
Lung cancer	72.41	71.04	70.58	61.33
Breast cancer	72.08	70.34	69.19	69.52
Scene-15	74.10	67.71	71.70	66.73

由表 2 可以看出,通过利用基于 FPGA 实现的贝叶斯算法,其分类的准确率要比 kNN,SVM 以及 LDA 的准确率高很多.

Mak[41]证明了能够使用基于 FPGA 的嵌入式异构系统芯片来做系统发育推论.Alachiotis 等人[42]基于 Mak 的研究,提出了一种基于 FPGA 加速最大似然估计的方法.他们利用 FPGA 芯片的本地内存高带宽、低延迟访问的特性,存储系统发育似然函数输入输出的似然估计向量.然而,他们的架构无法计算更高对数似然估计值,也无法执行在基于贝叶斯方法的系统发育树构建软件中用于防止条件概率向量下溢的缩放和归一化操作.因此,Stephanie 等人[43]提出了一种方法,利用 FPGA 协处理器来加速贝叶斯马尔科夫蒙特卡洛推论方法中的系统发育似然函数的处理.通过利用硬件实现系统发育似然函数,有效地替代了基于贝叶斯方法的系统发育树构建软件中归一化操作和对数似然估计计算.通过使用 FPGA 协处理器,在其上进行系统发育似然函数的映射和逻辑支持的技术开发.通过利用 FPGA 芯片上 DSP 模块和高带宽的本地内存来加速基于最大似然估计的方法,得到的性能优于传统的多核处理器.该方法设计的加速器,在基于双浮点的 Xilinx 内核生成器上,利用 Virtex-2 Pro FPGA 所达到的延迟为 38 周期,在 Virtex-6 FPGA 上所达到的延迟为 34 周期.表 3 显示了对于不同的数据集,CPU 计算所花费的时间,并将 Stephanie 等人提出的硬件方法与其进行比较.

Table 3 Bayes classification results comparison table[43]

表 3 贝叶斯分类结果对比表[43]

SW 节点平均处理时间(μs)				HW 节点平均处理时间(μs)和与 SW 节点速度比			
数据集	非 root 节点	Root 节点	平均值	HW 平均时间/节点数 @165MHz(μs)	速率比 vs. SW	HW 平均时间/节点数 @310 MHz(μs)	速率比 vs. SW
M993	16.1	43.1	17.1	6.6	2.6	3.5	4.9
M1319	20.7	54.2	22.7	9.0	2.5	4.8	4.7
M346	41.1	118.0	44.0	9.6	4.6	5.1	8.7
M1038	46.3	119.4	47.0	13.0	3.6	6.9	6.8
M1485	61.5	166.2	65.5	19.0	3.4	10.1	6.5
M4056	193.6	560.3	198.2	58.7	3.4	31.2	6.4
M3631	199.6	563.9	205.9	83.0	2.5	44.1	4.7

2.6 基于FPGA的人工神经网络

利用硬件并行实现人工神经网络,可以通过模拟、数字或混合信号设计的技术来实现.FPGA 在超大规模集成电路中具有更加成熟和灵活的数字化设计,它的高处理密度特性可用于人工神经网络的并行处理[44].

针对大规模文本数据,比较流行的人工神经网络[45]利用误差反向传播算法来训练多层感知.但是,用此方法会使训练变慢,并且在训练之前缺少明确的方法来确定网络拓扑结构.因此,文献[46]提出利用 FPGA 的可重构计算特性来解决这些问题.可重构计算由于是通用计算平台,因此每单位面积的处理性能更优.

基于 FPGA 的人工神经网络传统方法允许的最小精度是 16 位定点,该方法权衡面积来说被认为是最优的精度,使其能够在 FPGA 上获得高效的硬件性能.定点精度能够量化错误,相对应的浮点精度限制了量化错误.对此,文献[47]进一步讨论了单 FPGA 平台上在反向传播算法上使用浮点运算的可能性与灵活性.

为了确保基于 FPGA 的精密逻辑运算,在面积与精度之间的平衡很重要,合理的数值精度能够提高网络精度和收敛速度.随着精度的提高,逻辑面积(即 FPGA 资源)的花费代价也越高.浮点对比于定点来说,能够提供更高的精度,在计算时,计算结果将得到更小的量化误差,因此,其作为一种理想的数值表示法被广泛应用于通用计算平台.然而,由于 FPGA 的硬件资源限制,浮点数不如定点数那样灵活.因此,文献[47]提出了一种方法,用于优化权衡精度与面积之间的分配,来帮助 FPGA 的定点方法在通用计算中变得更加有效.在不牺牲质量性能的情况下,尽量减少硬件使用面积,并确定所能达到的最小精度.每个算法所允许的最小精度是不同的,Holt 等人[48]提出的人工神经网络的反向传播算法最小允许精度为 16 位定点计算.该精度在不牺牲人工神经网络的学习能力的情况下,最大限度地减少了硬件面积.通过基于 FPGA 实现的人工网络神经的反向传播算法,通过定点方法加强了密度优势,使由浮点精度产生的量化误差可以忽略不计.

Virtex-E(FPGA 的一种)可配置逻辑模块如图 5 所示,由 4 个逻辑单元(LC)和两个相似的 slice 元件组成.每个 LC 包含一个拥有 4 输入的查找表(LUT),专为逻辑运算功能进行快速超前逻辑进位,还包含一个存储元件(即触发器).

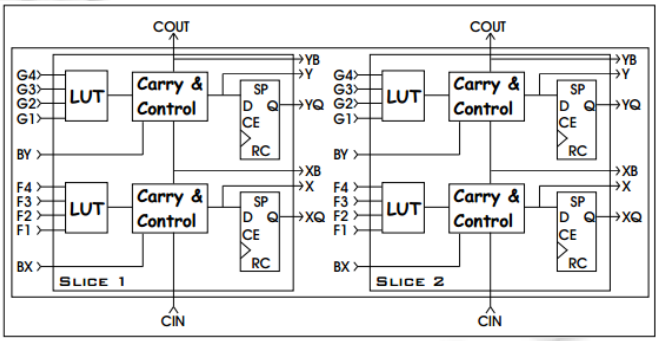


Fig.5 Virtex-E configurable logic module diagram[46]

图 5 Virtex-E 可配置逻辑模块图[46]

当把人工神经网络映射到 FPGA 中时,发现针对大规模文本数据和图像数据的计算,最主要的限制是大量资源在各个处理元件中共享问题.对此,Bonnici 等人在文献[49]中提出了一种优化乘法和资源分布式存储器的方法来解决限制问题,从而实现更高效的并行性、更快的速率、更少的资源被占用.该方法将网络层映射到一个特别的硬件结构中,使硬件资源利用率提高.利用一个可重构、参数化的神经节点作为神经实现的基础模块,并用 Verilog(HDL)作为实体模型.该方法在 FPGA 上实现人工神经网络总共设计了 3 个步骤:首先选择一个正确的 FPGA 方案;然后推导出一个高效的、高吞吐量的网络参数;最后,在硬件模型上进行实现.图 6 显示了该方法的接口设计实现的模型.由图中可以看出,寄存器的读写与 BlockRAMs 的权重有关,并且寄存器需要连接其他处理模块,使信号集与数据的输入输出相关联.INODE_in 和 INODE_out 信号与符合神经元的上层与下层相连接,Inode_dprev 与 Inode_dnext 在每层符合神经元之间形成了环状缓冲,从而高效解决了限制资源的问题.

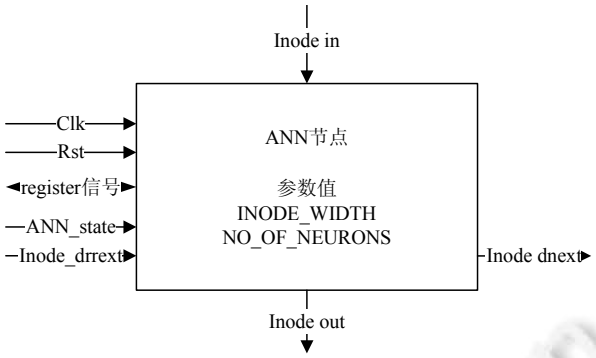


Fig.6 Interface design and implementation diagram[49]

图 6 接口设计实现图[49]

由于模拟超大规模集成电路的生产成本高,Barend 等人[50]提出了一种基于 FPGA 的加速器对人工神经网络中的大量单神经元进行仿真,使用硬件分时复用技术来扩展网络规模,最大限度地提高硬件的使用率.

表 4 对上述基于 FPGA 硬件平台的各种机器学习方向的数据密集型计算,对其优点、效率进行总结、对比.

Table 4 Comparison of the effects of various intensive computing based on FPGA

表 4 基于 FPGA 的各种密集型计算效果比较

模型	算法	优点	效率
K-means	Map-Reduce processing[13]	比用软件实现算法的速度快,降低了通信开销	高
	Tree-Based data structures[18]	支持大数据集,提高吞吐量	高
	Dynamic partial reconfiguration[19]	高能源效率,能处理较大维度的数据	中
贝叶斯	PSO[39]	精度效果较好,比用软件实现的速度快	高
	Accelerating the phylogenetic likelihood function[42]	比单核速度更快	高
	Bayesian MCMC inference methods[43]	协调多个 FPGA 共同计算自定义的并行算法	高
遗传算法	SMILE[22]	比集群实现的速度快,加强了可扩展性	高
	IP core implementation[23]	具有容错设计,收敛快	高
	UAV real-time path planning[24]	针对路径规划问题,能够实时产生结果	高
	HGA[32]	可集成于任何嵌入式系统中,易重新配置	中
人工神经网络	Analog VLSI ANN emulation[50]	提高硬件使用率,降低大规模使用成本	中
	ANN optimization[49]	在相同吞吐率下,减少空闲处理单元	高
	Floating-Point arithmetic[47]	提高了计算精度,减少计算误差	中

3 基于 GPU 的数据密集型计算

3.1 GPU概述

图形处理器(GPU)是图像应用领域的最成功的加速器.GPU 由数千个又小又高效的核心组成,这些核心的设计目的为更高效地并行处理多个任务.由于 GPU 由更多的内核组成,使其能够比 CPU 更高效地处理并行任务.GPU 具有如下几个关键的特性:比传统 CPU 具有更高的计算能力;具有高并行性,能够并行执行成百上千的线程;能够执行复杂的数学和几何计算;具有高内存带宽.

GPU 适用范围很宽,可应用于图形学、科学计算等领域.在目前的工业平台中,GPU 已成为主要的云平台并行计算设备.由于云平台最主要的功能是并行计算能力,因此,GPU 的高并行处理能力被广泛利用于云平台的并行计算中.但由于其生命周期短、能耗高、实时性较差的问题,使其在工业环境中不如 FPGA 那样被广泛使用.

3.2 基于GPU在数据库上的查询操作

针对大规模的文本数据,传统的基于 CPU 的数据库操作系统的效率很难满足用户需求,因此,He 等人[51]设

计实现了一个基于 GPU 的在内存中关系查询协处理系统(GDB),并设计了一套高度优化的数据并行操作,如分割、排序等.该系统的加速效果见表 5.由表 5 可以看出,在各种操作中,文献[51]所用的 GPU 方法比传统的 CPU 方法效率高很多.

Table 5 Database operation acceleration effect table[51]

表 5 数据库操作加速效果表[51]

初始				操作			
Primitive	CPU	GPU	速度比	Operators	CPU	GPU	速度比
Map	109	4	27.3	Selection	63	36.08	1.7
Scatter	1 312	104	12.6	Projection	20	0.86	23.3
Gather	1 000	103	9.7	OrderBy	2 500	1 000	2.5
Prefix scan	141	14	10.1	GroupBy	2 323	945	2.5
Reduce	31	11	2.8	Aggr.	32	11.62	2.8
Filter	62	37	1.7	NINLJ	528 000	75 000	7.0
Split	813	125	6.5	INLJ	4 235	649	7.0
Sort(qsort)	2 313	945	2.4	SMJ	5 030	1 946	2.6
–	–	–	–	HJ	2 550	1 327	1.9

文献[52]将原始关系代数操作映射到 GPU 上,设计了一个具有较好计算复杂性的算法结构,该算法的获取存储器模式及指令周期能够使机器利用率达到最大.Diamos 等人利用一个具有较低计算复杂性的不规则算法对输入数据集进行约减,来减少总运行时间,并将局部的结果输入一个常规的算法来使机器达到最大利用率.

在过去几年,专用的 GPU 被逐步发展成通用的计算设备及高效的并行计算模型,如 CUDA 和 OpenCL 等.Yuan 等人[53]发现,虽然过去几年在 GPU 上的查询优化处理研究效果显著,但是在大型的数据仓库生产系统中 GPU 并没有被真正地采用.对此,他们通过改变查询特点、软件技术以及 GPU 的硬件配置来解决在 GPU 上的三维数据仓库查询问题.

近年来出现了一些新型的处理器设计方案,通过将 CPU 和 GPU 集成在一个单一的芯片中来共享最后一级高级缓存.但是,通过这样耦合的 CPU-GPU 架构的主存储器带宽远远低于独立 GPU 的带宽,这样会导致目前的 GPU 查询协同处理范式受到内存延迟的影响.因此,He 等人[54]在基于主存联机分析处理数据库上提出了一种基于 CPU-GPU 架构的缓存查询协同处理范式.通过在 GPU 查询协同处理和 CPU 辅助减压时利用 CPU 辅助预取来减小缓存未命中率,提高查询处理性能.

3.3 基于GPU的*K*-means算法

K-means 算法具有收敛速度快、易于实现的特点,广泛应用于信息检索、图像处理及模式识别等领域,但是在一些针对大规模数据应用中(如物理模拟等)表现不佳.针对此种现象,一些学者利用 *k-d* 树来加快执行时间,其他学者通过改进算法,利用任务并行和数据并行来实现,但这种方法不可避免地会产生太多数据通信开销.因此,硬件平台需要具备可并行性来并行计算 *K*-means.传统的 CPU 实现该算法.由于实时处理大量数据,使得 CPU 的开销也变得繁重.因此,很多学者开始使用基于 GPU 来实现 *K*-means 算法高并行性,由于 GPU 允许用户在 pixel 级别的处理器引擎上实现碎片程序,因此能够实现 *K*-means 的并行算法.*K*-means 算法中,最需要进行并行计算的是数据对象的分配以及 *k* 个中心点的重新计算.因此,大多学者利用 GPU 来实现这两种大规模的数据密集型计算.大多数实现 *K*-means 算法都是利用单指令单数据(SISD)结构的 CPU 处理器来实现,使得该部分忽略了算法该有的并行特性.因此,Bai 等人[55]提出了基于 GPU 的单指令多数据(SIMD)架构来实现 *K*-means 算法.为了加速传统 *K*-means 算法中的计算密集型部分,将数据分配和 *k* 个中心重计算的工作由 GPU 来并行计算.算法架构图如图 7 所示.由于 GPU 对流量控制和数据缓存的能力相比 CPU 要弱一些,其更多的晶体管专为计算单元所设计,而且 GPU 与 GPU 内存之间的数据传输速率要比 CPU 与 CPU 之间的传输速率慢很多,因此,文献[55]利用 CPU 与 GPU 组合的方式来执行 *K*-means 算法.由图 7 可以看出,该架构融合了 CPU 与 GPU 各自的特性来执行 *K*-means 算法.

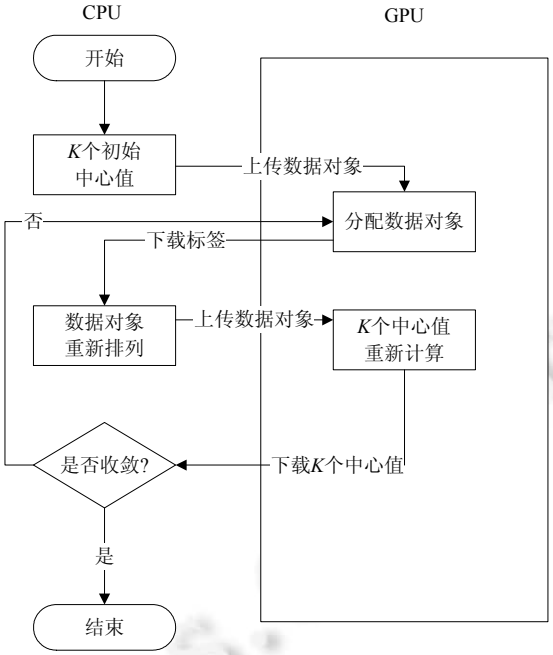


Fig.7 *K*-Means architecture diagram based on GPU[55]

图 7 基于 GPU 的 *K*-means 架构图[55]

针对大规模的高维文本数据,You 等人[56]发现,数据维度对基于 GPU 并行计算 *K*-means 有很大的影响.对此,他们针对高维度和低维度分别提出了不同的计算策略,使 GPU 的性能发挥到最大:对于低维度数据集,利用 GPU 芯片上的寄存器来减少数据访问延迟;对于高维度数据集,设计了一种新的模拟矩阵乘法,并利用 GPU 芯片上的寄存器以及共享芯片上的内存来提高从计算到内存获取的效率.

随着数据量的指数增长,单个 GPU 实现 *K*-means 聚类算法在不久的将来无法再适用于大规模数据集.因此,Kijsipongse 等人[57]设计并实现了一个基于 GPU 集群的大规模 *K*-means 并行集群.他们采用动态负载均衡的方法,在集群中对不同的 GPU 分配不同的工作负载.通过利用大量并行的 GPUs,提高 *K*-means 聚类中花费大量时间部分的执行速度.利用软件的分布式优点,在节点间通过简单的通信和合作来共享内存.

为了解决 *K*-means 对初始聚类中心和异常值敏感以及对数据多次遍历、计算导致计算任务繁重的缺点,Yan 等人[58]提出了基于 CUDA(一种基于 GPU 处理器实现的统一计算架构)利用最小化 d 维超球面内点数目来排除异常点,然后,可以通过密度的方法调整阈值来获得 k 个聚类中心.通过利用 GPU 的计算能力,使得该算法的效率得到提高.

由于 *K*-means 算法随着数据量的急剧增加运行效率变得越来越低,Zheng HX 等人[59]利用 GPU 的并行特性,基于 Hadoop 框架实现了 *K*-means 算法.由于现今的数据集变得越来越大,因此需要高性能的方法来解决这个问题.由于 *K*-means 算法是一个很容易被划分为多个独立任务的算法,而且 GPU 相比于 CPU 可以被看作是一个具有高计算能力、高存储带宽的多个并行处理器,因此,通过利用 GPU 来克服高计算需求,利用 Hadoop 来解决主存限制.

3.4 基于GPU的遗传算法

遗传算法广泛应用于数值分析、图像处理、模式识别、非线性优化问题等领域.针对大规模的文本、图像等类型数据,将其转换成二进制数据后进行遗传算法的计算.通常利用并行分布式系统来计算遗传算法,但由于每次迭代时都包含多个复杂的计算,使其在分布式中计算开销很大,通信和管理变得复杂.因此,处理器平台应能够对算法进行并行计算支持,GPU 具有良好的并行性.尽管 GPU 的单指令多线程架构通常不适用于传统的并

行遗传算法,但由于 GPU 具有较高的并行计算的性能,而且遗传算法具有良好的并行性,因此,研究人员们希望能够最大限度地利用 GPU 高效实现遗传算法的并行计算.细粒度并行遗传算法针对 GPU 的单指令多线程架构能够很好地匹配.Mohamed 等人[60]针对这种问题进行了相应的研究,将传统的并行计算模型应用于 GPU,并提出了一些优化方法来最大化通用 GPU 的使用效率.通过提高内存占用率并尽量减少对全局存储器的访问,来减少总执行时间.

Stefano 等人[61]提出了一种根据多核 CPU 的串行实现演化而来的模型,该模型基于 CPU/CUDA 来实现一个多功能的遗传算法.由于遗传算法具有并行的特性,利用通用 GPU 计算模式在模型的内部训练特点,该模型在处理性能和可扩展性方面进行了高度优化.该模型受机器学习监督模式启发,能够处理基于大量数据集的回归和分类两种科学问题.

由于 GPU 的内存空间受限,导致种群规模所占用的内存空间对算法的性能造成了影响.Martín 等人[62]针对此种情况对遗传算法进行了数据形式变化,分别对布尔型数据、非布尔型数据(将其转变成多比特数据)这两种情况进行了二进制遗传算法实现.

针对大规模的图数据,Kai 等人[63]将基于 GPU 的并行遗传算法用于根据实时路况生成智能行车路线.由于该计算量非常大,所以需要利用 GPU 来进行并行计算[64].通过利用 GPU 的共享内存来提高数据访问速度,从而解决大规模数据上计算的延时问题.

3.5 基于GPU的贝叶斯算法

贝叶斯算法广泛应用于统计分析、文本分类等领域.随着数据量的增加,贝叶斯算法对数据或文本的分类处理变得困难,因此需要使用并行化的手段来提高算法效率,所以需要硬件平台具备并行执行计算的能力.GPU 具有良好的并行计算能力,并且对比于利用 CPU 实现的降维策略、索引策略[65]、并行多处理单元等解决方案,GPU 能够提供更高的并行度、更低的能耗.而且,GPU 的高精度计算性能能够使贝叶斯算法的结果准确度更好.对此,Felipe 等人[66]提出了基于 GPU 的朴素贝叶斯并行方法来用于文本分类.在贝叶斯网络学习中,经常利用马尔科夫链蒙特卡洛方法(MCMC)来模糊推导贝叶斯网络.但是由于 MCMC 对于超过 15~20 个节点的复杂计算无法高效地执行,对此,Yu 等人[67]提出了使用通用处理器和通用 GPU 来加快贝叶斯网络学习的算法.他们利用 GPU 的并行性,在 60 个节点上并行实现了贝叶斯网络学习.Lei 等人[68]基于 GPU,利用朴素贝叶斯邻近算法解决了图像分类问题.Tomonari 等人[69]提出了在 GPU 上的、基于并行网格的递归贝叶斯估计方法,解决了实时自主导航问题.

虽然贝叶斯推论的几个并行算法已经实施在基于 CPU 的集群、多核 CPU 以及小型 CPU 和 GPU 的集群,但是没有一个方法能够实现同时充分利用 CPU 和 GPU 来计算.在文献[70–72]中,一种快速并行的贝叶斯系统发育推理被提出.基于此种方法,Chai 等人[73]提出了一种新型混合并行算法,实现了结合 MPI,OpenMP 以及 CUDA 编程的贝叶斯系统发育推理.他们利用 CPU 内核和 GPU,并平等地给这两种硬件平台分配工作,让它们同时进行计算.

Pratas 等人[74]实现了在系统发育树构建软件中系统发育似然函数的细粒度并行化,并将其运行在多核 CPU、cell BE 以及通用 GPU 这些多种不同的集群架构上.Zhou 等人[75]提出了一种在系统发育树构建软件上对 DNA 序列数据提高 GPU 效率的方法,有效地减少了数据在 CPU 和 GPU 之间的传输负载.

Suchard 等人[76]开发了一个软件库 BEAGLE,用于执行系统发育似然函数计算.BEAGLE 利用基于 GPU 实现的 CUDA 来编程,利用单核 CPU 来进行 SSE(streaming SIMD extensions)以及通过多核 CPU 中的 OpenMP 来编程[77].当前的系统发育树构建软件通过 SSE 能够快速计算似然函数,并能够兼容 BEAGLE 库,并能够将似然函数的计算交给 GPUs 进行[78].

针对大规模的图像数据,Perkins 等人[79]提出了一个基于 GPU 实现的无线电干涉仪测量公式,用于支持贝叶斯推理的无线电观测技术.Chantas 等人[80]提出了一种基于 GPU 加速实现的贝叶斯推理变形的方法,用于将图像重新构造成具有高分辨率的图像.Ferreira 等人[81]提出了用于实时实现的贝叶斯框架,它基于 GPU,利用 CUDA 来实现机器人多感官知觉.Wahib 等人[82]通过利用贝叶斯优化算法和一元启发式算法来搜索预连接片

段的组合,并将配合基受体连接的能量最小化,并通过 GPU 来加速计算每个片段组合的可能性.

3.6 基于GPU的图计算

近年来,由于大规模图数据的广泛存在和图计算的难解性,GPU 被广泛用于图计算中.针对较大规模的视频数据,文献[83]通过利用 GPU 的并行性,提出了一种基于稀疏局部特征和超图匹配[84]的方法来识别视频中的行为动作.在 GPU 硬件平台上,从时域数据的特性中提取一个连续的、快速的图匹配算法.传统方法中,通过图匹配或点集匹配图的方法,图和超图经常用于识别复杂的和非刚性模式的机器视觉,并利用近似方法解决最小化离散能量函数问题.在文献[83]中,通过在 GPU 上进行最优分配的并行计算,得到准确值来取代传统方案中近似问题的解决方案.该方案中,尽量使图结构简单化、正规化,从而推导出更有效的递归最小化算法.

从在线社会网络和其他真实世界的图数据能够得到大规模的有用信息,针对图中所包含的数据可以进行分析,从而有效地用于广告、安全以及提高网络用户的整体体验.然而,这些图数据分析需要大量的计算,因此,文献[84]提出了一种方法,利用大规模并行、大量多线程的 GPU 来解决此问题.由于真实世界的数据被认为是稀疏的,且具有不规则的数据依赖关系,研究人员利用有效的技术[85]存储预处理中的图数据,采用 CPU-GPU 异构的方式来实现图数据计算.文献[84]将问题转换成识别缺失边的问题,此边若加入图中,将导致图中三角形数最大限度的增加.或识别一条现有边,该边的去除将会导致图中的三角形数目最大限度的减少.通过该方法的转化,来计算图中关联数据.

GPGPU(通用计算图形处理单元)允许用户在当前个人电脑的图形硬件上进行并行计算.文献[86]基于GPU,针对最优图着色问题提出两种并行的并发读专一写(CREW)的 PRAM 算法:第 1 种方法是在 GPU 上计算顶点独立集,并对其分配颜色;第 2 种方法是针对边传递图的图结构,优化顶点独立集的计算,并对每个标准化的独立点集分配颜色.在分析社交网络中有影响力的人、分析电网、研究蛋白质的相互作用等领域中,中介性核心性算法被广泛用于图数据分析.由于其计算复杂,使其在大规模的图数据中精确计算变得不可行.而 GPU 的应用结构、高内存吞吐量以及并行性,使其适用于该种不规则的架构以及非结构化的应用.文献[87]实现了一种动态中介性核心性算法.传统算法中,图中点的子集中点数很少时,网络的更新对中心值的影响很大.因此,文献[87]通过有效映射单位工作线程来提高算法效率.

3.7 基于GPU的神经网络算法

在医学、天文等多个领域中,都需要针对某种问题对大规模的数据进行快速的分类.因此,需要加速神经网络的训练和测试的执行过程.然而针对神经网络的复杂计算,传统的硬件平台执行效率很低,因此,需要硬件平台拥有并行化的能力来使算法并行执行.文献[88]针对神经网络的复杂计算,利用 GPU 的并行编程和计算能力来加速神经网络的执行过程.神经网络的数学计算主要包括矩阵乘法运算,GPU 可以有效地使用每个分块,通过共享内存来执行矩阵乘法运算.除了矩阵运算,每个隐藏神经元激活函数的计算也可并行执行.因此,可以创建和矩阵数量相等的线程,每个线程可以在 GPU 处理器中独立地计算.

针对大规模的图像数据,检测其中的人类对象,需要在光谱图像中进行.通过提取行人的红外图像,进行特征描述,制定特征集来代表候选人,然后进行二进制分类特征的提取.文献[89]提出了一种行人检测的红外图像算法,通过自适应模糊 C 均值聚类和卷积神经网络的方法在 GPU 上实现.利用自适应模糊 C 均值聚类分割红外图像和检索候选行人,利用卷积神经网络同时学习相关的功能和执行二进制分类.该算法通过利用一种基于GPU 的深度学习框架[90]来实现.

文献[91]通过利用人工神经网络来实现频谱感知技术.通过利用 GPU,实现人工神经网络的并行计算,从而提高其性能.通过调整计算在 GPU 上的分布来影响整个算法的性能,并通过优化数据传输来减少执行时间.此外,利用细粒度并行处理,选择合适的参数,用于在GPU上处理大量的数据.文献[92]提出了一种基于皮层的通过利用 GPU 加速实现的神经网络模型用于视觉导航,并应用于物理机器人探索真实世界的环境.该模型由基于速率的运动能量模型和基于皮层的脉冲神经网络模型组成.该模型生成一个光流代表皮层来决定运动不连续的物体的位置,并结合这些代表目标位置的信号,引导机器人绕过障碍物向目标移动.

深度学习模型用于从原始数据学习高层次的特征表示,涉及图像、音频以及其他形式的复杂数据.也存在一些软件框架用于设计以及训练深层神经网络,如 Caffe[90]等.这些框架可以在同一台机器上使用多个 GPU,但是不能在多个分布式机器上利用 GPU,因为机器间的可用带宽限制了分布式 GPU 的性能.文献[93]提出了一种分布式的 GPU 方法,利用现有的深度学习框架进行机器间通信的扩展.通过一个 3 层的混合架构来支持 CPU、GPU 的分布式配置,通过分布式自由等待反向传播算法来提高 GPU 的利用率以及通信平衡,通过专门的感知架构通信协议来减小通信开销.利用以上手段来提高深层神经网络基于集群的训练速度.文献[94]提出一种新的分布式方案,通过深度学习来减少训练时间、通信开销并对计算模型并行实现.文献[95]基于麦克斯韦 GPU,提出一种算法 maxDNN,对卷积核进行有效的深度学习计算,利用典型的深度学习网络体系结构,使该算法的计算效率得到有效的提升.卷积神经网络的主要算法是多通道滤波的二维卷积,现今的 GPU 计算卷积的能力比 CPU 要快很多.fbcunn 是一种基于 GPU 实现的频繁域卷积,并在许多卷积形状上有明显的速度优势[96].

表 6 对上述基于 GPU 硬件平台的各种机器学习方向的数据密集型计算,对其优点、效率进行总结、对比.

Table 6 Comparison of the effects of various intensive computing based on GPU

表 6 基于 GPU 的各种密集型计算效果比较

模型	算法	优点	效率
K-means	GKcc[55]	加快复杂计算速度	中
	Hadoop-GPU framework[59]	利用 GPU 建立 Hadoop 集群,可扩展且具有良好的容错性	高
	Dynamic load balancing[57]	动态分配工作量来均衡负责,提高算法性能	高
	GPU-based *K*-means[56]	高计算内存访问效率	高
贝叶斯	Montblanc[79]	提高单精度、双精度的计算效率	高
	oMC^3[73]	结合 CPU 内核与 GPU 计算能力来加速计算效率	中
	GPU-NB[66]	减少内存消耗,多线程并行执行算法	中
	PQ[68]	通过将算法并行分解执行来提高图像分类效果	高
遗传算法	GAME[61]	对 GPU 的内存管理、调度及数据传输进行优化	高
	PGA[63]	解决复杂计算,提高活动安排问题的计算效果	高
	Bitwise operations[62]	用比特或布尔类型存储数据,从而加快算法执行效率	高
神经网络	ANN-ED[91]	通过均衡计算在 GPU 的分布以及优化数据传输,减少算法执行时间	中
	CMeans-CNN[89]	在相同吞吐率下,针对检测精度的基线算法有较好的效果	高
	Rapid automated classification[88]	提高了算法结果的精度以及算法的执行速度	高

4 基于其他平台的数据密集型计算

4.1 基于DPU的数据库操作

DPU(database processing unit)为数据库处理单元,Q100 是 DPU 的一个应用实例.Q100 数据库处理单元[97]使用特定的硬件改进了分析型数据库应用的能耗效率.这一加速器被称为 DPU,DPU 类似于 GPU(其中,GPU 应用于处理图形应用,而 DPU 应用于处理分析型数据库负载).文献[97]在理论概念层次上提出了 Q100 的设计方案,且相对于单线程和多线程的数据库管理系统软件,它能够保证一到两个数量级的加速.与以往工作不同的是,Q100 无需在硬件上编译查询,数据处理单元使用特定域电路来加速分析查询.Q100 提出了一个指令集体系结构,它能够利用专门的硬件实现特定的 SQL 运算符.该系统架构能够为模块化的应用提供进一步的扩展.同时,Q100 系统结构为正则表达式匹配或压缩/解压缩引擎等其他的流加速器设计提供了理论依据.由此可见,DPU 为加速器的设计提供了广阔的前景,开创了将体系结构与数据库结合起来应用的新方法.Wu 等人在文献[98]中指出,Q100 具有异构的 ASIC 集合,能够以非常少的能源消耗来快速地处理关系型表和列.对于 TPC-H 查询,相当于数据库管理系统软件,Q100 所产生的能量消耗会低 3 个数量级,且性能方面能提高 70 倍.

4.2 基于异构多核处理器的并行操作

多核处理器是将多个核心处理器集成在一个芯片上,对外进行功能服务.文献[99]基于异构多核处理器的分布式特性,实现了一种并行聚类算法(P-means)来解决大数据聚类的问题.该论文基于实现异构多核处理器的

StarSs 框架,通过 6 个步骤实现了聚类算法.每一步骤建立一个能够单独并行执行的任务,使整个算法能够并行执行.该算法通过在连续时间内利用 Amdahl 定律来比较理论计算和执行时间的测量值,对算法的实现进行评估,从而采用基于 StarSs 框架的并行计算.算法通过利用 30 个簇、60 万个包含 60 维度的数据点在异构多核处理器上进行聚类,大大提高了算法的并行效率.

4.3 基于同构多核处理器的并行操作

Hydra 作为一个典型的同构多核处理器,多被用于并行处理大规模数据.文献[100]利用 Hydra 来并行处理大规模的生物信息数据.普遍的大规模生物信息实验,利用科学工作流管理系统(SWFMS)来协调工作流程,控制监控整个执行过程.但是由于缺少在高性能计算(HPC)下的并行执行能力,使得 SWFMS 非常耗时.最近,Hydra 提出了解决 SWFMS 和 HPC 之间无法紧密关联的方案,通过为研究学者们提供一个透明的方法来并行执行工作流,同时能够捕获分布式文件系统中数据的存储位置.该文通过分析生物信息领域中分布式的使用,扩展了 Hydra 中间件,对生物信息的工作流提供分布式处理.

4.4 基于NDP的数据库操作

随着内存大小的持续增长,使得现代数据系统能够在内存中处理大规模的数据集.然而随着内存容量的增加,使得其与内存延迟大小成反比,这导致在内存中处理大规模数据使得两者无法相匹配,数据在内存中灵活移动变为了内存的性能瓶颈[101].数据库系统研究人员已经提出了一些解决方案[102−106],以尽量减少数据移动,并尽量访问硬盘中的数据.然而对于一个给定的查询,所有与查询结果相关的数据必须移动到内存中,因此,提高大规模数据集移动的效率变得很关键.在文献[1]中提出了 JAFAR 方法,基于 NDP 加速器来加速内存存储被选择的行列数据.JAFAR 实现了选择操作,并只允许改进后的查询列数据存储在内存中.该论文中,通过模拟硬件与软件结合,改进了指定查询所选择的列存储操作.

5 结论和展望

本文对面向新处理器的数据密集型计算方法进行了综述.从本文的综述可以看出:因为 FPGA 具有动态编译性,利用 FPGA 进行数据库上的查询操作效果比较好,能够通过利用不同的代数操作组合方式来执行不同的查询要求.FPGA 的可重构性使得它能够无限的重新配置成大量的特定电路,因此利用重新配置电路的方法来解决例如贝叶斯网络这类动态结构的计算,其表现比普通的硬件更加灵活.FPGA 与 GPU 对比于 CPU,他们的本地内存具有高带宽、低延迟访问的特性,因此对于频繁访问内存数据的计算,能够加快算法的数据访问速度.GPU 对于数据库查询利用多线程执行的方法来加速查询操作.对于具有高并行性的算法,例如遗传算法、*K*-means 算法等,可以利用 GPU 的高并行性,在 GPU 中的多个计算单元并行执行这类算法.GPU 对于高计算复杂性的数学计算和几何计算,利用其高浮点计算的能力,能够快速、高精度地进行处理.因此,对于具有高精度、高复杂性、高并行性的算法,可以利用 GPU 来并行实现;对于具有动态结构、高并行性的算法,适用于 FPGA 进行硬件并行加速处理.在工业上,面对大规模使用处理器的情况下,由于 GPU 耗能高,因此多数利用 FPGA 来进行工业设计与实现.

根据上述讨论,在基于新存储器的数据密集型计算有如下可以进一步研究的问题:

(1) 针对多级存储器的协同优化,目前只有单独研究新型处理机和新型存储结构的工作.根据实验,处理机和存储访问效率都有可能成为瓶颈,因而可以进一步研究新型处理机和新型存储结构的结合,针对多级存储器进行协同优化,可以整合多级存储和多处理机的优势,提高系统性能.

(2) 针对迭代计算的优化,在图计算、机器学习等重要算法中,设计大量迭代计算(如 *K*-means,PageRank 等).但是迭代计算由于其轮数不可控,涉及到共享信息等特点,往往成为并行计算的难题.因而进一步可以考虑利用新型处理机的计算能力,进一步优化迭代计算.

(3) 针对网络过滤/分析任务的优化,在当前有一类任务是在线分析、过滤,通过仅扫描数据一遍完成过滤和分析,而并不保存数据.这类计算任务是一类重要的数据密集型计算任务,当前的主要解决方案是

通过抽样完成,这显然会降低过滤和分析的精度.因而可以考虑设计基于新型处理机的分布式在线过滤/分析方法,以提高过滤/分析的精度和效率.

References:

[1] Xi S, Babarinsa O, Athanassoulis M, Idreos S. Beyond the wall: Near-Data processing for databases. In: Proc. of the Int'l Workshop on Data Management on New Hardware. 2015. [doi: 10.1145/2771937.2771945]

[2] Aingaran K, Smentek D, Wicki T, Jairath S, Konstadinidis G, Leung S, Loewenstein P, McAllister C, Phillips S, Radovic Z, Sivaramakrishnan R. M7: Oracle's next-generation sparc processor. IEEE Micro, 2015,2:36−45. [doi: 10.1109/MM.2015.35]

[3] Choi SH, Park N, Song YH, Lee SW. ASiPEC: An application specific instruction-set processor for high performance entropy coding. In: Proc. of the Ubiquitous Computing Application and Wireless Sensor. Springer-Verlag, 2015. 67−75. [doi: 10.1007/978-94-017-9618-7_7]

[4] Francisco P. The Netezza data appliance architecture: A platform for high performance data warehousing and analytics. IBM Redbooks, 2011.

[5] Becher A, Bauer F, Ziener D, Teich J. Energy-Aware SQL query acceleration through FPGA-based dynamic partial reconfiguration. In: Proc. of 2014 the 24th Int'l Conf. on Field Programmable Logic and Applications (FPL). IEEE, 2014. 1−8. [doi: 10.1109/FPL.2014.6927502]

[6] Mueller R, Teubner J, Alonso G. Glacier: A query-to-hardware compiler. In: Proc. of the 2010 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2010. 1159−1162. [doi: 10.1145/1807167.1807307]

[7] Dennl C, Ziener D, Teich J. On-the-Fly composition of FPGA-based SQL query accelerators using a partially reconfigurable module library. In: Proc. of the Annual IEEE Symp. on Field-Programmable Custom Computing Machines. IEEE, 2012. 45−52. [doi: 10.1109/FCCM.2012.18]

[8] Woods L, István Z, Alonso G. Ibex: An intelligent storage engine with support for advanced SQL offloading. Proc. of the VLDB Endowment, 2014,7(11):963−974. [doi: 10.14778/2732967.2732972]

[9] Scofield TC, Delmerico JA, Chaudhary V, Valente G. Xtremedata dbx: An FPGA-based data warehouse appliance. Computing in Science & Engineering, 2010,12(4):66−73. [doi: 10.1109/MCSE.2010.93]

[10] Sukhwani B, Min H, Thoennes M, Dube P, Iyer B, Brezzo B, Dillenberger D, Asaad S. Database analytics acceleration using FPGAs. In: Proc. of the 21st Int'l Conf. on Parallel Architectures and Compilation Techniques. ACM Press, 2012. 411−420. [doi: 10.1145/2370816.2370874]

[11] Sukhwani B, Thoennes M, Min H, Dube P, Brezzo B, Asaad S, Dillenberger D. Large payload streaming database sort and projection on FPGAs. In: Proc. of 2013 the 25th Int'l Symp. on Computer Architecture and High Performance Computing (SBAC-PAD). IEEE, 2013. 25−32. [doi: 10.1109/SBAC- PAD.2013.21]

[12] Yoshimi M, Kudo R, Oge Y, Terada Y, Irie H, Yoshinaga T. Accelerating OLAP workload on interconnected FPGAs with flash storage. In: Proc. of 2014 the 2nd Int'l Symp. on Computing and Networking (CANDAR). IEEE, 2014. 440−446. [doi: 10.1109/CANDAR.2014.87]

[13] Choi YM, So HKH. Map-Reduce processing of *k*-means algorithm with FPGA-accelerated computer cluster. In: Proc. of 2014 IEEE the 25th Int'l Conf. on Application-Specific Systems, Architectures and Processors (ASAP). IEEE, 2014. 9−16. [doi: 10.1109/ASAP.2014.6868624]

[14] Estlick M, Leeser M, Theiler J, Szymanski JJ. Algorithmic transformations in the implementation of *K*-means clustering on reconfigurable hardware. In: Proc. of the 2001 ACM/SIGDA ninth Int'l Symp. on Field Programmable Gate Arrays. ACM Press, 2001. 103−110. [doi: 10.1145/360276.360311]

[15] Leeser M, Theiler J, Estlick M, Szymanski JJ. Design tradeoffs in a hardware implementation of the *k*-means clustering algorithm. In: Proc. of the 2000 IEEE Sensor Array and Multichannel Signal Processing Workshop. IEEE, 2000. 520−524. [doi: 10.1109/SAM.2000.878063]

[16] Saegusa T, Maruyama T. An FPGA implementation of *k*-means clustering for color images based on *Kd*-tree. In: Proc. of the Int'l Conf. on Field Programmable Logic and Applications (FPL 2006). IEEE, 2006. 1−6. [doi: 10.1109/FPL.2006.311268]

[17] Saegusa T, Maruyama T. An FPGA implementation of real-time *K*-means clustering for color images. Journal of Real-Time Image Processing, 2007,2(4):309−318. [doi: 10.1007/s11554-007-0055-8]

[18] Winterstein F, Bayliss S, Constantinides G. FPGA-Based *K*-means clustering using tree-based data structures. In: Proc. of 2013 the 23rd Int'l Conf. on Field Programmable Logic and Applications (FPL). IEEE, 2013. 1−6. [doi: 10.1109/FPL.2013.6645501]

[19] Hussain HM, Benkrid K, Ebrahim A, Erdogan AT, Seker H. Novel dynamic partial reconfiguration implementation of *k*-means clustering on FPGAs: Comparative results with GPPs and GPUs. Int'l Journal of Reconfigurable Computing, 2012,2012:1-15. [doi: 10.1155/2012/135926]

[20] Hussain HM, Benkrid K, Seker H, Erdogan AT. FPGA implementation of *K*-means algorithm for bioinformatics application: An accelerated approach to clustering Microarray data. In: Proc. of the 2011 NASA/ESA Conf. on Adaptive Hardware and Systems (AHS). IEEE, 2011. 248−255. [doi: 10.1109/AHS.2011.5963944]

[21] Koza J, Bennett F, Andre D, Keane M. Genetic programming III: Darwinian invention and problem solving. In: Proc. of the Evolutionary Computation. 1999. 451−453.

[22] Pedraza C, Castillo J, Martínez JI, Huerta P, Bosque JL, Cano J. Genetic algorithm for Boolean minimization in an FPGA cluster. The Journal of Supercomputing, 2011,58(2):244−252. [doi: 10.1007/s11227-010-0401-7]

[23] Fernando PR, Katkoori S, Keymeulen D, Sankaran H, Stoica A, Zebulum R, Rajeshuni R. Customizable FPGA IP core implementation of a general-purpose genetic algorithm engine. IEEE Trans. on Evolutionary Computation, 2010,14(1):133−149. [doi: 10.1109/TEVC.2009.2025032]

[24] Allaire FCJ, Tarbouchi M, Labonté G, Fusina G. FPGA implementation of genetic algorithm for UAV real-time path planning. In: Proc. of the Unmanned Aircraft Systems. Springer-Verlag, 2009. 495−510. [doi: 10.1007/s10846-008-9276-8]

[25] Aporntewan C, Chongstitvatana P. A hardware implementation of the compact genetic algorithm. In: Proc. of the 2001 IEEE Congress Evolutionary Computation. 2001. 624−629. [doi: 10.1109/CEC.2001.934449]

[26] Vavouras M, Papadimitriou K, Papaefstathiou I. High-Speed FPGA-based implementations of a genetic algorithm. In: Proc. of the Int'l Symp. on Systems, Architectures, Modeling, and Simulation (SAMOS 2009). IEEE, 2009. 9−16. [doi: 10.1109/ICSAMOS.2009.5289236]

[27] Scott SD, Samal A, Seth S. HGA: A hardware-based genetic algorithm. In: Proc. of 1995 ACM the 3rd Int'l Symp. on Field-Programmable Gate Arrays. ACM Press, 1995. 53−59. [doi: 10.1109/FPGA.1995.241945]

[28] Shackleford B, Snider G, Carter RJ, Okushi E, Yasuda M, Seo K, Yasuura H. A high-performance, pipelined, FPGA-based genetic algorithm machine. Genetic Programming and Evolvable Machines, 2001,2(1):33−60. [doi: 10.1023/A:1010018632078]

[29] Guo L, Thomas DB, Luk W. Customisable architectures for the set covering problem. ACM SIGARCH Computer Architecture News, 2014,41(5):101−106. [doi: 10.1145/2641361.2641378]

[30] Unlt GIP. Hardware Implementation of Genetic Algorithms Using FPGA. 2004.

[31] Guo L, Thomas DB, Luk W. Automated framework for general-purpose genetic algorithms in FPGAs. In: Proc. of the Applications of Evolutionary Computation. Berlin, Heidelberg: Springer-Verlag, 2014. 714−725. [doi: 10.1007/978-3-662-45523-4_58]

[32] Nambiar VP, Balakrishnan S, Khalil-Hani M, Marsono MN. HW/SW co-design of reconfigurable hardware-based genetic algorithm in FPGAs applicable to a variety of problems. Computing, 2013,95(9):863−896. [doi: 10.1007/s00607-013-0305-5]

[33] Fe J, Aliaga RJ, Gadea R. Experimental platform for accelerate the training of anns with genetic algorithm and embedded system on FPGA. In: Proc. of the Natural and Artificial Computation in Engineering and Medical Applications. Berlin, Heidelberg: Springer-Verlag, 2013. 413−420. [doi: 10.1007/978-3-642-38622-0_43]

[34] Tuncer A, Yildirim M, Erkan K. A hybrid implementation of genetic algorithm for path planning of mobile robots on FPGA. In: Proc. of the Computer and Information Sciences III. London: Springer-Verlag, 2013. 459−465. [doi: 10.1007/978-1-4471-4594-3_47]

[35] Vavouras M, Papadimitriou K, Papaefstathiou I. Implementation of a genetic algorithm on a virtex-II Pro FPGA. In: Proc. of the FPGA. 2009. 287−287. [doi: 10.1145/1508128.1508206]

[36] Vuillemin JE, Bertin P, Roncin D, Shand M, Touati HH, Boucard P. Programmable active memories: reconfigurable systems come of age. IEEE Trans. on VLSI System, 1996,4(1):56−69. [doi: 10.1109/92.486081]

[37] Compton K, Hauck S. Reconfigurable computing: A survey of systems and software. ACM Computing Surveys, 2002,34(2):171−210. [doi: 10.1145/508352.508353]

[38] Hauck S, DeHon A. Reconfigurable computing: The theory and practice of FPGA-based computation. In: Proc. of the Systems on Silicon. Morgan Kaufmann Publishers, 2007.

[39] Hibbard MJ, Peskin ER, Sahin F. FPGA implementation of particle swarm optimization for Bayesian network learning. Computers & Electrical Engineering, 2013,39(8):2454−2468. [doi: 10.1016/j.compeleceng.2013.07.018]

[40] Choi SW, Lee CH. A FPGA-based parallel semi-naive Bayes classifier implementation. IEICE Electronics Express, 2013,10(19):20130673−20130673. [doi: 10.1587/elex.10.20130673]

[41] Mak TST, Lam KP. Embedded computation of maximum-likelihood phylogeny inference using platform FPGA. In: Proc. of the IEEE Computational Systems Bioinformatics Conf. Table of Contents, 2004. 512−514. [doi: 10.1109/CSB.2004.1332479]

[42] Alachiotis N, Sotiriades E, Dollas A, Stamatakis A. Exploring FPGAs for accelerating the phylogenetic likelihood function. In: Proc. of the 8th IEEE Int'l Workshop on High Performance Computational Biology (HiCOMB 2009). 2009. [doi: 10.1109/IPDPS.2009.5160929]

[43] Zierke S, Bakos JD. FPGA acceleration of the phylogenetic likelihood function for Bayesian MCMC inference methods. BMC Bioinformatics, 2010,11(1):184−195. [doi: 10.1186/1471-2105-11-184]

[44] Anguita D, Boni A, Ridella S. A digital architecture for support vector machines: Theory, algorithm, and FPGA implementation. IEEE Trans. on Neural Network, 2003,14(5):993−1009. [doi: 10.1109/TNN.2003.816033]

[45] Rumelhart DE, McClelland JL, PDP Research Group. Parallel Distrubuted Processing: Explorations in the Microstructure of Cognition. Vol.1: Foundations, Cambridge: MIT Press, 1986.

[46] Dehon A. The density advantage of configurable computing. IEEE Computer, 2000,33(5):41−49. [doi: 10.1109/2.839320]

[47] Nichols KR, Moussa MA, Areibi SM. Feasibility of floating-point arithmetic in FPGA based artificial neural networks. In: Proc. of the CAINE. 2002.

[48] Holt JL, Baker TE. Back propagation simulations using limited precision calculations. In: Proc. of the IJCNN-91-Seattle Int'l Joint Conf. on Neural Networks. IEEE, 1991. 121−126. [doi: 10.1109/IJCNN.1991.155324]

[49] Bonnici M, Gatt EJ, Micallef J, Grech I. Artificial neural network optimization for FPGA. In: Proc. of the 13th IEEE Int'l Conf. on Electronics, Circuits and Systems (ICECS 2006). IEEE, 2006. 1340−1343. [doi: 10.1109/ICECS.2006.379730]

[50] van Liempd B, Herrera D, Figueroa M. An FPGA-based accelerator for analog VLSI artificial neural network emulation. In: Proc. of the 13th Euromicro Conf. on Digital System Design: Architectures, Methods and Tools (DSD). IEEE, 2010. 771−778. [doi: 10.1109/DSD.2010.20]

[51] He B, Lu M, Yang K, Fang R, Govindaraju NK, Luo Q, Sander PV. Relational query coprocessing on graphics processors. ACM Trans. on Database Systems, 2009,34(4):21:1−21:39. [doi: 10.1145/1620585.1620588]

[52] Diamos GF, Wu H, Lele A, Wang J. Efficient relational Algebra Algorithms and Data Structures for GPU. 2012.

[53] Yuan Y, Lee R, Zhang X. The yin and yang of processing data warehousing queries on GPU devices. Proc. of the VLDB Endowment, 2013,6(10):817−828. [doi: 10.14778/2536206.2536210]

[54] He J, Zhang S, He B. In-Cache query co-processing on coupled CPU-GPU architectures. Proc. of the VLDB Endowment, 2014,8(4):329−340. [doi: 10.14778/2735496.2735497]

[55] Bai HT, He LL, Ouyang DT, Li ZS, Li H. *K*-Means on commodity GPUs with CUDA. In: Proc. of the 2009 WRI World Congress on Computer Science and Information Engineering. IEEE, 2009. 651−655. [doi: 10.1109/CSIE.2009.491]

[56] Li Y, Zhao K, Chu X, Liu J. Speeding up *k*-means algorithm by GPUs. In: Proc. of the 2010 IEEE 10th Int'l Conf. on Computer and Information Technology (CIT). IEEE, 2010. 115−122. [doi: 10.1109/CIT.2010.60]

[57] Kijsipongse E. Dynamic load balancing on GPU clusters for large-scale *K*-means clustering. In: Proc. of the 2012 Int'l Joint Conf. on. Computer Science and Software Engineering (JCSSE). IEEE, 2012. 346−350. [doi: 10.1109/JCSSE.2012.6261977]

[58] Yan B, Zhang Y, Yang Z, Su H, Zheng H. DVT-PKM: An improved GPU based parallel *K*-means algorithm. In: Proc. of the Intelligent Computing Methodologies. Springer Int'l Publishing, 2014. 591−601. [doi: 10.1007/978-3-319-09339-0_60]

[59] Zheng HX, Wu JM. Accelerate *K*-means algorithm by using GPU in the hadoop framework. In: Proc. of the Web-Age Information Management. Springer Int'l Publishing, 2014. 177−186. [doi: 10.1007/978-3-319-11538-2_17]

[60] Wahib M, Munawar A, Munetomo M, Akama K. Optimization of parallel genetic algorithms for nVidia GPUs. In: Proc. of the 2011 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2011. 803−811. [doi: 10.1109/CEC.2011.5949701]

[61] Cavuoti S, Garofalo M, Brescia M, Longo G, Ventre G. Genetic algorithm modeling with GPU parallel computing technology. In: Proc. of the Neural Nets and Surroundings. Berlin, Heidelberg: Springer-Verlag, 2013. 29−39. [doi: 10.1007/978-3-642-35467-0_4]

[62] Pedemonte M, Alba E, Luna F. Bitwise operations for GPU implementation of genetic algorithms. In: Proc. of the 13th Annual Conf. Companion on Genetic and Evolutionary Computation. ACM Press, 2011. 439−446. [doi: 10.1145/2001858.2002031]

[63] Wang K, Shen Z. A GPU-based parallel genetic algorithm for generating daily activity plans. IEEE Trans. on Intelligent Transportation Systems, 2012,13(3):1474−1480. [doi: 10.1109/TITS.2012.2205147]

[64] Shen Z, Wang K, Zhu F. Agent-Based traffic simulation and traffic signal timing optimization with GPU. In: Proc. of 2011 the 14th Int'l IEEE Conf. on Intelligent Transportation Systems (ITSC). IEEE, 2011. 145−150. [doi: 10.1109/ITSC.2011.6083080]

[65] Christen P. A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. on Knowledge and Data Engineering, 2012,24(9):1537−1555. [doi: 10.1109/TKDE.2011.127]

[66] Viegas F, Andrade G, Almeida J, Rocha L, Gonçalves M, Ferreira R. GPU-NB: A fast CUDA-based implementation of Naïve Bayes. In: Proc. of 2013 the 25th Int'l Symp. on Computer Architecture and High Performance Computing (SBAC-PAD). IEEE, 2013. 168−175. [doi: 10.1109/SBAC-PAD.2013.16]

[67] Wang Y, Qian W, Zhang S, Yuan B. A novel learning algorithm for Bayesian network and its efficient implementation on GPU. arXiv preprint arXiv:1210.5128, 2012.

[68] Zhu L, Jin H, Zheng R, Feng X. Effective naive Bayes nearest neighbor based image classification on GPU. The Journal of Supercomputing, 2014,68(2):820−848. [doi: 10.1007/s11227-013-1068-7]

[69] Furukawa T, Lavis B, Durrant-Whyte HF. Parallel grid-based recursive Bayesian estimation using GPU for real-time autonomous navigation. In: Proc. of the 2010 IEEE Int'l Conf. on Robotics and Automation (ICRA). IEEE, 2010. 316−321. [doi: 10.1109/ROBOT.2010.5509396]

[70] Feng X, Cameron KW, Buell DA. PBPI: A high performance implementation of Bayesian phylogenetic inference. In: Proc. of the 2006 ACM/IEEE Conf. on Supercomputing. New York: IEEE, 2006. [doi: 10.1109/SC.2006.47]

[71] Feng X, Buell DA, Rose JR, Waddell PJ. Parallel algorithms for Bayesian phylogenetic inference. Journal of Parallel Distrib Computer, 2003,63(7-8):707−718. [doi: 10.1016/S0743-7315(03)00079-0]

[72] Feng X, Cameron KW, Sosa CP, Smith B. Building the tree of life on terascale systems. In: Proc. of the 21st Int'l Parallel and Distributed Processing Symp. Long Beach, 2007. 1−10. [doi: 10.1109/IPDPS.2007.370214]

[73] Chai J, Su H, Wen M, Cai X, Wu N, Zhang C. Resource-Efficient utilization of CPU/GPU-based heterogeneous supercomputers for Bayesian phylogenetic inference. The Journal of Supercomputing, 2013,66(1):364−380. [doi: 10.1007/s11227-013-0911-1]

[74] Pratas F, Trancoso P, Stamatakis A, Sousa L. Fine-Grain parallelism using multi-core, Cell/BE, and GPU systems: Accelerating the phylogenetic likelihood function. In: Proc. of the Int'l Conf. on Parallel Processing (ICPP 2009). IEEE, 2009. 9−17. [doi: 10.1109/ICPP.2009.30]

[75] Zhou J, Liu X, Stones DS, Xie Q, Wang G. MrBayes on a graphics processing unit. Bioinformatics, 2011,27(9):1255−1261. [doi: 10.1093/bioinformatics/btr140]

[76] Suchard MA, Rambaut A. Many-Core algorithms for statistical phylogenetics. Bioinformatics, 2009,25(11):1370−1376. [doi: 10.1093/bioinformatics/btp244]

[77] Ayres DL, Darling A, Zwickl DJ, Beerli P, Holder MT, Lewis PO, Huelsenbeck JP, Ronquist F, Swofford DL, Cummings MP, Rambaut A, Suchard MA. BEAGLE: An application programming interface and high-performance computing library for statistical phylogenetics. Oxford University Press, 2011. [doi: 10.1093/sysbio/syr100]

[78] Ronquist F, Teslenko M, van der Mark P, Ayres DL, Darling A, Höhna S, Larget B, Liu L, Suchard MA, Huelsenbeck JP. MrBayes 3.2: Efficient Bayesian phylogenetic inference and model choice across a large model space. Systematic Biology, 2012,61(3):539−542. [doi: 10.1093/sysbio/sys029]

[79] Perkins S, Maraism P, Zwart J, Natarajan I, Tasse C, Smirnov O. Montblanc: GPU accelerated radio interferometer measurement equations in support of Bayesian inference for radio observations. arXiv preprint arXiv:1501.07719, 2015.

[80] Chantas G. Variational Bayesian image super-resolution with GPU acceleration. In: Proc. of the Artificial Neural Networks (ICANN 2010). Berlin, Heidelberg: Springer-Verlag, 2010. 480−489. [doi: 10.1007/978-3-642-15819-3_64]

[81] Ferreira JF, Lobo J, Dias J. Bayesian real-time perception algorithms on GPU. Journal of Real-Time Image Processing, 2011,6(3):171−186. [doi: 10.1007/s11554-010-0156-7]

[82] Wahib M, Munawar A, Munetomo M, Akama K. A Bayesian optimization algorithm for De Novo ligand design based docking running over GPU. In: Proc. of the 2010 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2010. 1−8. [doi: 10.1109/CEC.2010.5586531]

[83] Lombardi E, Wolf C, Celiktutan O, Sankur B. Activity recognition from videos with parallel hypergraph matching on GPUs. arXiv preprint arXiv:1505.00581, 2015.

[84] Çeliktutan O, Wolf C, Sankur B, Lombardi E. Fast exact hyper-graph matching with dynamic programming for spatio-temporal data. Journal of Mathematical Imaging and Vision, 2015,51(1):1−21. [doi: 10.1007/s10851-014-0503-6]

[85] Chatterjee A, Radhakrishnan S, Antonio JK. On analyzing large graphs using GPUs. In: Proc. of 2013 IEEE the 27th Int'l Conf. on Parallel and Distributed Processing Symp. on Workshops & Ph.D. Forum (IPDPSW). IEEE, 2013. 751−760. [doi: 10.1109/IPDPSW.2013.235]

[86] Sengupta S. Parallel graph coloring algorithms on the GPU using OpenCL. In: Proc. of the 2014 Int'l Conf. on Computing for Sustainable Global Development (INDIACom). IEEE, 2014. 353−357. [doi: 10.1109/IndiaCom.2014.6828158]

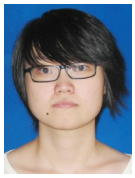
[87] McLaughlin A, Bader D. Revisiting edge and node parallelism for dynamic GPU graph analytics. In: Proc. of the 2014 IEEE Int'l Conf. on Parallel & Distributed Processing Symp. on Workshops (IPDPSW). IEEE, 2014. 1396−1406. [doi: 10.1109/IPDPSW.2014.157]

[88] Peker M, Şen B, Gürüler H. Rapid automated classification of anesthetic depth levels using GPU based parallelization of neural networks. Journal of Medical Systems, 2015,39(2):1−11. [doi: 10.1007/s10916-014-0182-2]

[89] John V, Mita S, Liu Z, Qi B. Pedestrian detection in thermal images using adaptive fuzzy *C*-means clustering and convolutional neural networks. In: Proc. of 2015 the 14th IAPR Int'l Conf. on Machine Vision Applications (MVA). IEEE, 2015. 246−249. [doi: 10.1109/MVA.2015.7153177]

[90] Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T. Caffe: Convolutional architecture for fast feature embedding. In: Proc. of the ACM Int'l Conf. on Multimedia. ACM Press, 2014. 675−678. [doi: 10.1145/2647868.2654889]

[91] El Hafid Y, Elrharras A, Guennoun K, Amri A, Wahbi M. Accelerating the detection of spectral bands by ANN-ED on a GPU. Computer and Information Science, 2015,8(1):95−107.
[92] Beyeler M, Oros N, Dutt N, Krichmar JL. A GPU-accelerated cortical neural network model for visually guided robot navigation. Neural Networks, 2015,72:75−87. [doi: 10.1016/j.neunet.2015.09.005]
[93] Zhang H, Hu Z, Wei J, Xie P, Kim G, Ho Q, Xing E. Poseidon: A system architecture for efficient GPU-based deep learning on multiple machines. arXiv preprint arXiv:1512.06216, 2015.
[94] Wang J, Cheng L. DistDL: A distributed deep learning service schema with GPU accelerating. In: Proc. of the Web Technologies and Applications. Springer Int'l Publishing, 2015. 793−804. [doi: 10.1007/978-3-319-25255-1_65]
[95] Lavin A. maxDNN: An efficient convolution kernel for deep learning with maxwell GPUs. arXiv preprint arXiv:1501.06633, 2015.
[96] Vasilache N, Johnson J, Mathieu M, Chintala S, Piantino S, LeCun Y. Fast convolutional nets with fbfft: A GPU performance evaluation. arXiv preprint arXiv:1412.7580, 2014.
[97] Wu L, Lottarini A, Paine T, Kim MA, Ross KA. The Q100 Database Processing Unit. 2015.
[98] Wu L, Lottarini A, Paine TK, Kim MA, Ross KA. Q100: The architecture and design of a database processing unit. ACM SIGPLAN Notices, 2014,49(4):255−268. [doi: 10.1145/2644865.2541961]
[99] Foina AG, Planas J, Badia RM, Ramirez-Fernandez FJ. *P*-Means, a parallel clustering algorithm for a heterogeneous multi-processor environment. In: Proc. of the 2011 Int'l Conf. on High Performance Computing and Simulation (HPCS). IEEE, 2011. 239−248. [doi: 10.1109/HPC Sim.2011.5999830]
[100] Coutinho F, Ogasawara E, De Oliveira D, Braganholo V, Lima AA, Dávila AM, Mattoso M. Data parallelism in bioinformatics workflows using Hydra. In: Proc. of the 19th ACM Int'l Symp. on High Performance Distributed Computing. ACM Press, 2010. 507−515. [doi: 10.1145/1851476.1851550]
[101] Balasubramonian R, Chang J, Manning T, Moreno JH, Murphy R, Nair R, Swanson S. Near-Data processing: Insights from a MICRO-46 workshop. IEEE Micro, 2014,34(4):36−42. [doi: 10.1109/MM.2014.55]
[102] Abadi D, Boncz P, Harizopoulos S, Idreos S, Madden S. The Design and Implementation of Modern Column-Oriented Database Systems. Now, 2013.
[103] Balkesen C, Teubner J, Alonso G, Özsu MT. Main-Memory hash joins on multi-core CPUs: Tuning to the underlying hardware. In: Proc. of the 2013 IEEE 29th Int'l Conf. on Data Engineering (ICDE). IEEE, 2013. 362−373. [doi: 10.1109/ICDE.2013.6544839]
[104] Petraki E, Idreos S, Manegold S. Holistic indexing in main-memory column-stores. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2015. 1153−1166. [doi: 10.1145/2723372.2723719]
[105] Pirk H, Petraki E, Idreos S, Manegold S, Kersten M. Database cracking: Fancy scan, not poor man's sort! In: Proc. of the 10th Int'l Workshop on Data Management on New Hardware. ACM Press, 2014. [doi: 10.1145/2619228.2619232]
[106] Porobic D, Liarou E, Tozun P, Ailamaki A. ATraPos: Adaptive transaction processing on hardware Islands. In: Proc. of 2014 IEEE the 30th Int'l Conf. on Data Engineering (ICDE). IEEE, 2014. 688−699. [doi: 10.1109/ICDE.2014.6816692]



王鹤澎(1992－),女,山东省郓城人,硕士生,主要研究领域为海量数据计算.

孔欣欣(1994－),女,硕士,主要研究领域为数据库,大数据,数据质量.

王宏志(1978－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为大数据管理,数据质量管理,XML 数据管理.

李建中(1950－),男,博士,教授,博士生导师,CCF 会士,主要研究领域为无线传感器网络,物联网,数据库,海量数据管理.

李佳宁(1992－),男,硕士,主要研究领域为数据库,大数据,数据质量.

高宏(1966－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为无线传感器网络,物联网,海量数据管理,数据挖掘.