多核及众核体系结构下线性代数算法研究进展

陶 袁[1],祝明发[2*]
(1. 吉林师范大学 数学学院,吉林 四平 13600; 2. 北京航空航天大学 计算机学院,北京 100191)

摘 要: 线性代数算法是科学和工程计算的核心算法,高性能计算机的硬件规模越来越大,体系结构越来越复杂,已有的包含这些算法的应用程序如何在当前体系结构的高性能计算机平台上获得较高的性能是当前研究必须解决的问题. 本文综述在当前多核和众核的异构平台上线性数值计算方面的现状、存在的问题和面临挑战; 对这些问题可能存在的问题进行系统的分析,并给出可能发展方向.

关键词: 多核 CPU; 众核 GPU; 线性代数算法
中图分类号: TP316 **文献标识码:** A **文章编号:** 1674-3873-(2015)03-0032-08

0 引言

线性代数算法在科学和工程计算中有广泛应用,在当前硬件规模越来越大和体系结构越来越复杂的情况下,许多已有的并行线性代数算法无法在当前体系结构的平台上获得较高的性能,国内外科研工作者在这方面做大量的研究工作,并取得许多科研成果. 从近几年国内外与本论文研究方向相关的顶级会议和顶级期刊发表的有关线性代数数值算法的研究看,主要包含基于存储结构性能优化的数值算法、基于通讯性能优化的数值算法和 GPU 加速的并行数值算法等三个方面.

1 基于存储结构性能优化方法

基于存储结构的性能优化方法包括: 矩阵分解、求解特征值和特征向量、求解线性方程组和稀疏矩阵乘法的性能优化方法等四个方面的内容.

1.1 矩阵分解

Rozložník M. 等[1] 研究采用部分主元(partial pivoting)策略把对称矩阵分解成三对角形式的矩阵,该分块算法能充分使用当前计算机的体系结构,实验结果显示该算法能获得较好的 linpack 性能. Druinsky A. 等[2] 研究利用树形结构稀疏模式分解矩阵,该研究提出采用兄弟占优主元策略(sibling-dominant pivoting),采用列主元重排策略最小化填充,从而达到优化性能的目的. Gunter B. C. 等[3] 和 Marques M. 等[4] 主要研究并行核外计算(out of core computing)和 QR 分解算法,其中 Gunter B. C. 等给出问题规模增加和处理器数目增加带来的算法可扩展性的问题,实验证明该算法获得 80% 的理论峰值.

Gunter B. C. 等算法和 Marques M. 等算法存在的问题: 当矩阵规模较大,需要多次访问磁盘操作时,较优的 I/O 实现效果不明显; 并且没有验证异步 I/O 模式下获得性能. Marques M. 等研究则借助于运行环境在多核处理器上实现核外计算(out of core computing) QR 分解算法,使原有的 QR 分解算法进行很小的改动就可以在该运行环境上运行,该研究的创新点在于把磁盘看成是分层内存的一部分,使磁盘对程序员而言是透明的.

Avron H. 等[5] 研究采用有序列部分主元非对称模式多面方法,该算法主要用于有小数量或中等数量处理器(论文中实验用 32 个处理器)的共享内存,该算法与已有的共享内存方面的研究采用稀疏部分主元 LU 分解算法-SuperLU_MT 相比,SuperLU_MT 算法扩展性更好,但论文实现的算法更稳定,运行速度更

收稿日期: 2015-02-25
第一作者简介: 陶 袁(1972-),男,吉林省梨树县人,现为吉林师范大学数学学院副教授,博士. 研究方向: 并行数值算法.
* 通讯作者: 祝明发(1945-),男,四川省岳池县人,教授,博士,博士生导师. 研究方向: 计算机体系结构、并行算法、高性能计算机系统和网络、人工智能.

快.

1.2 求解特征值和特征向量

Baia Y. H. 等[6-7]主要针对求解对称块三对角矩阵的特征值时,在保证精度的情况下如何提高算法的吞吐量进行研究. 主要采用并行块三对角的分治算法计算对称块三对角矩阵的特征解来保证精度,该研究采用数据与任务并行实现数据分配和负载均衡,实验结果表明: 该实现算法高效、可扩展好同时能获得指定精度的特征值.

Konda T. 等[8]研究采用双分治算法(double Divide and Conquer algorithm) 求解双对角矩阵特征值. 在奇异值互相独立的情况下,该算法相对标准 QR 分解算法和分治算法而言,计算速度、计算精度和正交性都较好; 对于聚类矩阵(Clustered Matrices) ,采用双分治算法内存使用量为 O(n) ,而标准分治算法的内存使用量为 O(n2) . 该算法存在的问题是: 舍入误差带来的精度较低.

1.3 求解线性方程组

Hirshman S. P. 等[9]研究采用循环规约算法(Cyclic Reduction Algorithm) 分解块三对角矩阵. 该算法与支持多线程的应用程序相结合使得解三对角线性方程组有较好的可扩展性. Naumov M. 等[10]研究把具有带状结构矩阵的线性方程组分成多个互相重叠独立的块的方法. 该方法提高并行度,由此提高算法的性能. Sekhara S. C. 等[11]研究采用 WZ 分解法求解矩阵形状为对称三对角线性方程组. 采用该方法提高程序的并行度,同时使结点内部的通讯时间减少 40%.

1.4 稀疏矩阵乘法的性能优化

Bender M. A. 等[12]采用缓存健忘算法(cache-oblivious) 获得最优的 I/O 复杂性等方面的研究. 该研究表明从理论上借助于排序方法可以实现缓存健忘算法最优的 I/O 实现. Jacob R. 等[13]研究表明该方法并没有考虑 CPU 的因素,而且采用该方法需要预处理过程,由此带来额外的开销,该研究从理论上看是最优的,而算法实现不一定达到最优.

Yzelman A. N. 等[14]结合超图划分的方法,研究采用压缩行存储的稀疏矩阵与稠密向量相乘的缓存行为. 通过使用缓存模拟器得到不同形状的稀疏矩阵在不同缓存替换算法表现的不同性能,为稀疏矩阵与稠密向量相乘的缓存健忘(cache-oblivious) 算法优化提供理论指导. 如图 1 所示,超图划分方法是把稀疏矩阵的同行(同列) 的非零元素转换为有边连接的超图,借助于对图的划分达到提高缓存利用率的目的.

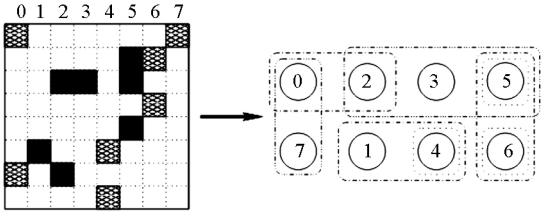

图 1 稀疏矩阵转换为超图[14]

Pichel J. C. 等[15]研究在运行时通过参数描述稀疏矩阵的结构,并对稀疏矩阵的非零元素进行重排实现矩阵变型,提高矩阵的数据局部性,从而达到提高算法在 SMP 平台上吞吐量的目的. Ternam O. 等[16]研究是第一个采用分块方法实现稀疏矩阵与稠密向量相乘,从而达到高效使用缓存的目的. Blelloch G. E. 等[17]从并行算法设计角度研究在具有共享缓存的多核平台上采用较好的分隔器可以实现具有 Cache-Oblivious 特性的稀疏矩阵与向量相乘(Sparse Matrix Vector Product,SMVP) 算法. 该方法具有深度较浅的优点. Pinar A. 等[18]研究在压缩行存储(Compressed Sparse Row,CSR) 方式的基础上,进一步采用同一行内部连续的数据分块的方式保存,提高缓存的命中率,进而达到提高性能的目的. 该研究存在的问题是采用三层循环; 保存数据采用四个数组,增加存储开销. Im E. J. 等[19]研究通过采用矩形缓存块的方式改善访问数据的时空局部性,达到减少通讯提高性能的目的.

Yzelmana A. N. 等[20]研究采用二维布局方式实现基于缓存健忘算法的稀疏矩阵与稠密向量相乘. 采用 CSR 方式,研究在二维方式上尽可能使用缓存达到减少内存开销的目的. 该研究存在的问题是: 为了获得较高的性能,采用稀疏块的方式进行划分,而且不同的块采用不同的存储结构,当划分的块较多时会带来巨大的开销.

Vuduc R. W. 等[21-23]采用分块压缩行技术(Blocked Compressed Sparse Row,BCSR) 实现稀疏矩阵与稠密向量相乘; 借助缓存的数据重用技术可以获得较高的加速比. 进一步 Vuduc R. W. 等[24]还采用多向量技术,对称结构,对角化技术和重排技术等从而增加稀疏矩阵与稠密向量相乘的数据重用达到性能优化的目的. Vuduc R. W. 等[25]设计 OSKI(Optimized Sparse Kernel Interface) 低级原语组件,实现针对不同的硬件平台自动调节稀疏矩阵计算内核,从而为上层用户隐藏底层细节.

Nishtala R. 等[26-27]和 Lee B. C. 等[28]提出采用缓存分块的方式提高数据局部性,进而减少通讯达到提

高 SMVP 性能的目的. Pichel J. C. 等[29]把在单个处理器上 SMVP 程序移植到 NUMA 结构共享内存的多处理器上,并建立模型. 在运行环境中根据矩阵的结构确定数据的局部性,并借助重排技术提高数据的局部性,由此优化 SMVP 在 SMP 上的性能. Buluc A. 等[30-32]研究解决并行稀疏矩阵与稀疏矩阵相乘问题; Buluc A. 等[33]还采用 CSB(Compressed Sparse Block) 方式,解决当前稀疏矩阵与稠密向量相乘及转置的稀疏矩阵与稠密向量相乘无法同时获得较高性能的问题,该结构设计能较好的使用缓存. Batista V. H. F. 等[34]研究采用压缩稀疏行列(Compressed Sparse row-column,CSRC) 的存储方式实现对称非零模式稀疏矩阵与稠密向量相乘; 该研究的第一个创新点是采用四种方式尽量避免对输出向量的竞争,第二个创新点是使用着色算法对线程并发处理的行进行分组; 该研究在一定程度上提高稀疏矩阵与稠密向量相乘算法的性能. 但该研究存在的问题是为避免对输出数据的竞争采用的暂时变量增加内存开销.

Liu S. F. 等[35]研究在曙光 S4800A1 上使用 OpenMP 作为并行运行环境实现 SMVP 算法,该研究比较 CSR 和 BCSR 两种存储形式和三种调度方式(静态、动态和指定方式) 获得的性能,还把这三种规划方式与非零调度方式相比较,实验结果表明采用非零调度方式在多数情况下能获得较好的性能. Blelloch G. E. 等[36]采用分段求和(segmented sum) 方法,在 Cray Y-MP C90 的向量多处理机上实现稀疏矩阵与稠密向量相乘. 该方法与通常的串行算法相比需要更少的存储空间和更方便的数据表示形式,由此获得较高的性能. Zhang N. 等[37]采用向前分段操作(forward segmented operations) 实现稀疏矩阵与稠密向量相乘,与传统的采用向后分段操作(backward segmented operations) 相比缓存利用率更高. Goumas G. 等[38]研究对当前硬件平台上影响 SMVP 性能的因素进行研究. 得出结论: 减少索引数据的工作集规模,进而减少直接的内存引用,稀疏部分填零,感知短行长度和循环开销等都会影响算法的性能. 袁娥等[39]研究采用自动优化技术提高 SMVP 算法的性能. 该研究通过采用寄存器分块算法和启发式分块选择算法,在 Pentium 4 和 AMD Athlon 平台上测试 10 个矩阵,平均加速比达到 1.69 和 1.48.

1.5 存储结构优化小结

表 1 给出基于存储优化的各种线性代数运算不同的优化方法,综合可以看出: 为减少通讯量或提高缓存命中率,采用基于压缩和分块存储仍然是当前稀疏矩阵存储结构优化的主要研究方向. 随着问题规模增加,多种方法混合逐渐成为多数算法获得较高性能、较高的计算精度或较好稳定性算法的必要途径. 根据高性能计算体系结构的发展趋势,限于 CPU 与 GPU 体系结构的差别,把这些算法移植到 GPU 上,并获得较高的实际计算速度,这些策略是否仍然适用是科研工作者需要研究的问题.

表 1 不同策略的存储结构优化比较

名称	具体方法	稳定性/精度	填充	优化策略	存在的问题
矩阵分解	部分主元策略	稳定	否/最小	通过矩阵变形,或针对特定形状矩阵	引入较小的解三角矩阵的额外开销
	核外计算	稳定	否	对稠密矩阵分块,在磁盘与内存之间设置运行环境实现 I/O 与计算并行.	I/O 速度与处理器实际计算速度不匹配
求特征值和特征向量	分块三对角求解特征值	稳定	否	使用混合的数据/任务混合并行取得数据和负载均衡	如果合并后的子问题规模差别很大会出现负载不均衡的情况
	双分治算法求解双对角特征值	精度较低	否	双分治算法求解双对角特征值,用分治算法和双分治算法混合方法求解特征向量	程序健壮性不好且精度较低
求解线性方程组	循环规约算法求解块三对角矩阵方程	精度不高	最小	并行度较高,算法可扩展性好	精度不高
	重叠法	好	部分填充	重叠法求解带状矩阵线性方程组	只适合对称正定矩阵或奇异矩阵
	WZ 分解算法	好	否	在分治算法中,当子矩阵规模降到原来两级时引入 WZ 分解算法使该算法更高效	不通用
稀疏矩阵乘法	缓存健忘算法(cache oblivious)	好	否	借助于重排技术达到 I/O 最优	重排引入额外开销
	超图划分方法	好	否	稀疏矩阵非零元素按所在行列用超图连接的边标示,并通过递归划分提高缓存命中率	转换超图的过程引入额外开销

表 1(续)

分块存储	好	否	通过分块存储提高缓存命中率	无法同时按行按列高效访问非零元素
压缩索引法	好	否	压缩索引减少通讯量	
分段计算方法	好	否	缓存或内存的利用率更高	
感知短行方法	好	部分填充	通过部分填充方法尽可能达到负载均衡	

2 基于通讯性能优化的数值算法

通讯性能优化的数值算法包括稠密矩阵的最小通讯问题研究和稀疏矩阵的通讯性能优化问题研究.

2.1 最小通讯问题研究

最小通讯的数值算法研究也是一个渐进的过程,首先是卡耐基梅隆大学 Hong 等[40]研究最早给出单节点内部串行情况下矩阵相乘、快速傅里叶变换等输入/输出的通讯下界;之后是以色列特大维夫大学大学 Irony D. 等[41]等给出分布式内存环境下稠密矩阵相乘的通讯下界,并验证该下界是矩阵所有串行或并行操作的通讯下界; Loomis 和 Whitney 等[42]给出的从面到量(surface-to-volume)的关系式,为当前的通讯下界算法的研究提供理论依据,图 2 以矩阵乘法为例对这个问题进行具体说明,表达式为(1):

$$|V| \leqslant \sqrt{|V_A| \times |V_B| \times |V_C|} \tag{1}$$

其中$|V|$代表立方体的容量,而$|V_A|$,$|V_A|$,$|V_A|$分别代表整个立方体分别沿不同面映射后得到子集的容量.

伯克利分校 Demmel J. 等研究组在 Loomis 和 Whitney 等研究基础上为线性代数数值算法的通讯下界问题做进一步研究,并取得大量的研究成果,具体相关研究如下:

Ballard G. 等[43]研究对当前解数值线性代数的稠密矩阵算法的通讯下限研究进行综述; Ballard G. 等[44-46]等研究给出单节点内部稠密矩阵的 Strassen 算法和其他快速稠密矩阵相乘获得通讯下界的算法;同时给出稠密矩阵串行和并行带主元 LU 分解,QR 分解和 Cholesky 分解获得通讯下界的算法;并进一步给出异构结点内部稠密矩阵相乘或稠密矩阵和向量相乘获得通讯下界的算法.

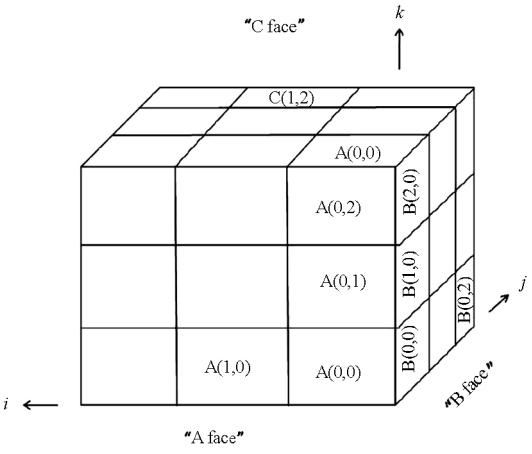

图 2 矩阵乘法几何模型[43]

Demmel J. 等[47]研究推广了通讯避免的 Krylov 子空间方法,使得采用迭代法解稀疏矩阵的线性方程组不会因为矩阵相乘级数的增加而增加通讯量,从而达到通讯最优. GRIGORI L. [48-49]等研究采用二维循环布局策略实现稠密矩阵 LU 分解的通讯避免算法. Mohiyuddin M. [50]等给出采用迭代法求解稀疏矩阵线性方程组获得最小通讯的算法,并证明该算法的稳定性. Solomonik E. [51]等研究给出并行三维布局的情况下矩阵相乘和 LU 分解等获得通讯下界的相关算法. 纽约州立大学石溪分校 Bender M. A. [12]等研究单节点内部 I/O 模式下稀疏矩阵与稠密向量乘积获得最优性能算法,并给出通讯下界.

法国巴黎南大学 Griori L. [52]等研究找到适合稀疏矩阵 Cholesky 分解满足通讯下限的算法,该算法的通讯下限不受矩阵是否按行(或按列)排列的约束,从而达到全局优化. 该算法实现用简单的方式把以前的基于稀疏矩阵的 Cholesky 分解算法和稠密矩阵的 Cholesky 分解算法有效的结合在一起.

2.2 稀疏矩阵的通讯性能优化问题研究

在分布式内存的并行计算机上借助于超图划分方法[53]减少通讯量方面的研究,该方法的性能严重依赖于矩阵的结构.

(1) 超图划分方法

Rek U. V. C. A. 等[53]在一维实现方式下通过构建超图划分模型把稀疏矩阵的划分问题转换为超图划分问题,通过合理的超图划分方法达到减少总通讯量的目的. 该方法可以使通讯量减少 63%(平均减少 30% ~38%). Vastenhouw B. 等[54]在分布式内存的并行计算机上,在二维方式下采用递归双划分(recursive bipartitioning)稀疏矩阵,每次把矩形矩阵按照非零元素的个数平均分成相同的两部分,使划分后总通

讯量最小. 实验结果表明: 采用该方法可以实现总通讯量和每个处理器最大通讯总量最小.

(2) 通讯性能优化

Bulu A. 等[55] 主要针对多核和众核体系结构的处理器采用压缩稀疏块(Compressed Sparse Block, CSB) 的数据表达方式来减少通讯量. 该研究使对称矩阵的数据通讯量减少一半; 同时在此数据表达方式基础上进一步采用寄存器分块的方式,并通过采用 bitmask 标识 0 数据的存储,使非对称矩阵的通讯量减少. Kourtis K. 等[56-57] 研究对采用压缩行存储的稀疏矩阵与稠密向量相乘,对列索引采用增量编码的方式和值采用压缩方式减少多线程应用程序对内存带宽的竞争.

Karakasis V. 等[58] 在 SMP 或 NUMA 平台上研究采用分块情况下,不同块形状对 SMVP 操作性能的影响; 特别是采用向量化情况下,对访存密集型操作,这种性能差异表现的更为突出. 本论文通过构建变块性能模型,为不同形状矩阵数据提供不同分块策略,从而为 SMVP 性能优化提供理论依据. Williams S. 等[59] 研究在多核平台上 SMVP 的优化方法,采用 DMA 方式可以一定程度上缓解多核平台上内存带宽不足的问题; 以及在并行环境下,不同的并行工具使用也会带来较大的性能差异(如在本论文实现中的 Pthread 和 mpi 实现相比较),两种策略都会在一定程度上影响 SMVP 在多核平台上的性能. Goumas G. 等[60] 通过分析不同形状矩阵在 SMP 和 NUMA 平台上存在的性能瓶颈,得出在当前这两种典型的体系结构硬件平台上运行 SMVP 算法获得较高性能优化方法: (1) 内存使用量过大带来的瓶颈问题,提出解决这个问题在于减少内存的通讯量; (2) 因为稀疏矩阵数据分布的随机性和访问向量的间接性导致对相乘向量的访问无法正确预测是当前平台上影响性能的另一个因素; (3) 每行很少的非零数带来循环的大量开销也是影响性能的一个方面. 针对上面三个主要问题,提出采用如下方式优化; 对单线程算法: (1) 最小化算法的工作集从而达到减少算法总循环次数是未来算法优化的一个方向; (2) 对特定形状的矩阵采用适合的存储方式也是当前当前提高 SMVP 算法性能的另一个研究方向. 对于多线程算法: (1) 采用合适的划分机制和任务分配方法; (2) 研究多线程方式下总线使用的仲裁方法; (3) 研究在 Cell 或 GPGPU 硬件平台下适合 SMVP 的编程模型等都是未来热点的研究方向.

Belgin M. 等[61] 采用基于模式表达的方法提高稀疏矩阵与稠密向量相乘的性能; 该研究发现许多矩阵通过划分成块后多数块具有相同的模式,由此把具有相同特征的块用定制的内核程序计算,由于在计算中不使用索引,大大减少了内存带宽的需求,由此提高应用程序的性能; 该方法既不检测稠密块,也不采用 0 填充的方法,使得该方法特别适合没有稠密非零的矩阵,该方法适合明确预取和向量化; 而且更利于并行. Brahme D. 等[62] 提供框架实现: 利用贪婪算法抽取变规模稠密子矩阵,以负载均衡的方式划分稀疏矩阵,同时在每个节点维护部分信息,实现计算与通讯互相重叠; 通过上述三种优化途径减少内存的通讯量并隐藏了通讯延迟. Willcock J. 等[63] 采用压缩非零元素索引的方式减少主机内存与 CPU 之间的通讯量.

2.3 通讯优化小结

表 2 给出基于不同通讯优化策略适合的体系结构、针对的矩阵类型和优缺点. 从表中可以看出,稀疏矩阵的存储优化问题是科学和科研的研究热点,但目前针对特定应用的稀疏矩阵转置避免的存储结构研究还很有限,当前的压缩稀疏块存储格式能否直接移植到 GPU 上是需要验证的科学问题; 并且在有关部分稀疏矩阵算法的通讯下界问题的研究中,能否根据稀疏矩阵的非零元素的分布计算出适合大多数稀疏矩阵算法的通讯下界也是科学和研究需要解决的问题.

表 2 通讯优化优缺点

名称	体系结构	矩阵类型	优化方法	负载均衡	存在的问题
最小通讯问题研究	分布式内存	所有矩阵	使 CPU 与内存或不同 CPU 之间通讯量最小	否	部分稀疏矩阵算法没有得到下界
超图划分方法	分布式内存	结构化方阵	把稀疏矩阵转化为超图减少总通讯量	是	增加预处理时间
压缩稀疏块	多核处理器	稀疏矩阵	采用压缩稀疏块实现转置避免高效按行/按列访问非零元素	是	针对多核 CPU 动态并行
分块	并行	稀疏矩阵	采用分块或进一步压缩方法减少通讯量	部分是	只能按行/按列高效访问非零元素

3 GPU 加速线性代数算法

Lessig C. 等[64] 研究在 GPU 上实现求解对称三对角矩阵的特征值,与 Linpack 相关实现相比获得 50

倍的加速比. 由于该研究使用 GeForce 8800 系列 GPU,所以该研究精度与 Linpack 在 CPU 上相比较低. Zhang Y. 等[65]研究在 GPU 上加速解三对角线性方程组提出采用并行循环规约(parallel cyclic reduction)和递推倍增(recursive doubling)相结合的算法,该算法与顺序 Linpack 实现相比可以获得 28 倍的加速比. Agullo E. 等[66]研究在 StarPU 平台上实现多 GPU 的 QR 分解使计算峰值达到理论上限. 而且该研究可以在 StarPU 上调度和管理 CPU 和 GPU,同时可以为上层用户提供统一的内存接口. 如图 3 所示,该研究是目前唯一一个为上层用户屏蔽 CPU 和 GPU 具体实现细节的平台.

Balland 等[67]研究给出异构平台上获得 CPU 与 GPU 最小通讯的通讯下界算法,并给出两个达到这个通讯下界的算法实现.

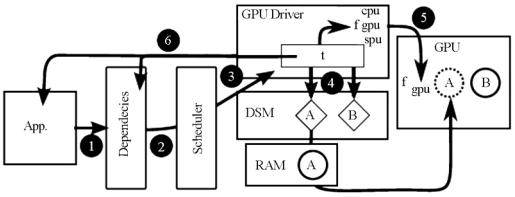

图 3 StarPU 任务规划图[67]

3.1 GPU 精度问题研究

Tomov S. 等[68]研究采用单精度与双精度的混合技术在多核及有 GPU 加速器的平台上解稠密线性方程组获得较优的性能; 同时提高 GTX 200 系列 GPU 的精度. Wolfe J. M. 等[69]首次提出部分和算法: 即在所有元素执行相加运算时部分元素先执行相加形成部分和,之后把部分和再相加形成最终和. 使用该算法根据两个相加元素之间的差距不同分别放到不同的硬件累加器上操作,达到提高硬件累加器的计算精度的目的. 作为高性能计算全新的硬件平台,科研工作者在 GPU 上进行高性能计算也进行大量研究,从矩阵相乘操作来看,主要体现在计算精度和计算峰值两个方面.

从 GPU 的计算精度看,Karl E.[2][70]从最后一位浮点数误差研究,由此产生的精度问题并给出该位的误差界(Units in Last Place); Maruyama N.[71]等在程序中植入软件的错误检测代码(Error-Checking Codes, ECC)实现内存错误检查,该研究增加应用程序一倍左右的性能开销; Anderson A. G.[72]提出 GPU 上采用 KSF(Kahan Summation Formula)求和算法提高矩阵相乘操作的精度[73-74],但该方法导致 GPU 实际计算峰值大幅下降; Larsen E. S.[75]利用倒数方法提高 GPU 浮点数精度; Goddeke G.[76-77], Baboulin M.[78]和 Strzodka R.[79]等采用混合精度的方法既能保证计算精度又保证计算峰值.

3.2 基于 GPU 的稀疏矩阵与稠密向量相乘

Choi J. W. 等[80]研究建立 GPU 上实现稀疏矩阵与稠密向量相乘算法的性能模型,并应用此模型优化算法. 该研究实现根据稀疏矩阵的不同形状自动调度不同的接口从而为上层用户提供较高的计算峰值.

Bell N. 和 Garland[81]设计一种在 GPU 上高效计算稀疏矩阵与稠密向量相乘的方法. 这种方法的关键思想用一个线程 warp 计算稀疏矩阵的一行与向量相乘,这种实现能够在并行性和负载均衡之间更好的平衡,同时计算乘积后的规约过程可以在 warp 内部实现. 该研究很大程度上提高基于 GPU 的稀疏矩阵与稠密向量相乘算法的性能,该源代码在 CUSP[81]包中提供,后来整合到 NVIDIA 公司的 CUSPARSE 函数包[82]中. 该研究采用混合方法(COO + ELL,Coordinate format(COO)适合压缩行或列存储格式,ELLPACK format(ELL)适合向量和单指令流多数据流体系结构)对非结构化稀疏矩阵与稠密向量相乘获得高性能.

Choi[80]等研究一种在 GPUs 上基于模型的自动 SMVP 优化策略. 他们采用分块压缩的行和分块 ELLPACK(BELLPACK)方式在 NVIDIA T10P 多核处理器(C1060)上单精度吞吐量达到 29.0GFLOPS 和双精度吞吐量达到 15.7GFLOPS. Deng 等[83]采用数据重排技术在 GPU 上加速 SMVP 计算,他们采用的技术特别适合超稀疏矩阵. Su 等[84]设计全新的稀疏矩阵格式—鸡尾酒格式(Cocktail format)及 OpenCL[85]实现—clSpMV. 这个格式能够在程序运行环境中动态分析稀疏矩阵并推荐一种在 GPU 适合的运行格式. Tang 等[86]研究通过压缩行/列索引的方式减少与 GPU 全局内存的通讯,Fu 等[87]研究针对非结构化网格提出采用并行有限元解的方法并获得 87 倍加速比. 然而,所有与 GPU 上 SMVP 的研究工作因为无法在通常的稀疏矩阵存储格式保证高效按行或按列访问非零元素,所以无法同时为 GPU 上 SMVP 和 SMTVP 提供较高的吞吐量. Shengen Yan 等[88]研究提供 GPU 上 SMVP 框架,该研究设计基于 COO 格式的 GPU 上高效的存储格式,使用该存储格式可以提高缓存的命中率,同时使负载更均衡,性能优于 NVIDIA 公司函数库 CUSPARSE 及 clSpMV 的结果.

3.3 GPU 加速线性代数算法小结

表 3 给出当前 GPU 加速线性代数算法实现方法及优缺点,从近几年相关领域的顶级会议和期刊的研究成果看,有关 GPU 的线性代数算法的加速方面的研究一直是科学和研究的热点,但目前仍然有几个方向是近几年需要科研工作者关注的热点研究: (1)用更通用的方法求解线性方程组方面,(2)整合 CPU 与

GPU 统一平台运行环境的异构处理器的调度算法,(3) 基于 CUDA 的 GPU 线性代数算法的加速研究也很有限,(4) 基于 OpenCL 针对不同厂商的 CPU 和 GPU 均获得较高实际计算速度算法研究等.

表 3 GPU 加速线性代数算法比较

名称	优化方法	同步方法	优点	缺点
解三对角线性方程组	并行循环规约方法	多次加载 kernel	并行循环规约方法(parallel cycle reducing) 与递归倍增(recursive doubling) 有效结合提高算法性能	能够用该方法解块三对角线性方程组问题
StarPU	CPU 和 GPU 的统一平台	依据算法	为上层用户提供统一的接口	任务规划采用静态方式,会出现负载不均衡问题
CPU 与 GPU 任务分配	通讯下界算法	依据算法	给出 CPU 与 GPU 之间任务分配的通讯下界算法	只给出达到该下界的部分线性代数算法
GPU 精度问题	混合方法	依据算法	获得较高的精度	用双精度收集结果,受 GPU 双精度浮点数性能的限制
	硬件实现部分和算法	依据算法	性能较高	需要额外的硬件支持
	KSF 求和算法	依据算法	精度较高	实际计算速度较低
稀疏矩阵与稠密向量相乘	面向吞吐量算法实现	warp 切换	借助 warp 切换避免显示的同步操作	不支持分块存储
	ClSpMV 框架	无	根据稀疏矩阵非零元素分布选择合适的数据结构,采用 OpenCL 实现,跨不同 GPU 平台	该研究没有 CPU 结果
	yaSpMV 框架	消除全局同步	分块保存 COO 格式稀疏矩阵提高缓存利用率,消除负载不均衡,消除全局同步,用位压缩提高带宽利用率,实际计算速度最高	没有给出分析矩阵带来的额外开销

4 结论

本文综述了近几年在多核及众核平台上线性代数算法方面的研究进展. 从近几年发表的学术论文看,通讯优化问题一直是体系结构方面算法优化的主题,对多核 CPU 平台是处理器与主机内存、同一个结点内部不同 CPU 之间或不同的结点处理器之间的优化; 对 CPU 和 GPU 组成的异构平台,通讯优化主要针对减少 CPU 与 GPU 之间的通讯及 GPU 内部处理器核与全局内存之间的通讯优化. 所以未来的研究方向除本文每节阐述的研究方向外,还有如何针对 GPU 体系结构,设计新算法或优化已有的算法减少处理器核与全局内存的通讯及 CPU 与 GPU 之间的通讯也是未来主要的研究方向.

参 考 文 献

[1] M. Rozložník, G. Shklarski, S. Toledo, et al. Partitioned Triangular Tridiagonalization [J]. ACM Transactions on Mathematical Software, 2011, 37(4): 1 ~ 17.

[2] A. Druinsky, S. Toledo. Factoring matrices with a tree-structured sparsity pattern [J]. Linear Algebra and its Applications, 2011, 435(5): 1099 ~ 1110.

[3] B. C. Gunter, R. A. V. D. Geijn. Parallel Out-of-Core Computation and Updating of the QR Factorization [J]. ACM Transactions on Mathematical Software, 2005, 31(1): 60 ~ 78.

[4] M. Marqués, G. Quintana-Ortí, E. S. Quintana-Ortí, et al. Out-of-Core Computation of the QR Factorization on Multi-core Processors [C]. Euro-Par 2009 Parallel Processing, Lecture Notes in Computer Science, Springer Berlin Heidelberg, Germany, 2009.

[5] H. Avron, G. Shklarski, S. Toledo, et al. Parallel Unsymmetric Pattern Multifrontal Sparse LU with Column Preordering [J]. ACM Transactions on Mathematical Software, 2008, 34(2): 41 ~ 71.

[6] Y. H. Bai, R. C. Ward. A Parallel Symmetric Block-Tridiagonal Divide-and-Conquer Algorithm [J]. ACM Transactions on Mathematical Software, 2007, 33(4): 25/1-25/23.

[7] Y. H. Baia, R. C. Wardc. Parallel block tridiagonalization of real symmetric matrices [J]. Journal of Parallel and Distributed Computing, 2008, 68(5): 703 ~ 715.

[8] T. Konda, Y. Nakamura. A new algorithm for singular value decomposition and its parallelization [J]. Parallel Computing, 2009, 35(6): 331 ~ 344.

[9] S. P. Hirshman, K. S. Perumalla, V. E. Lynch, et al. BCYCLIC: A parallel block tridiagonal matrix cyclic solver [J]. Journal of Computational Physics, 2010, 229(18): 6392 ~ 6404.

[10] M. Naumov, A. H. Sameh. A tearing based hybrid parallel banded linear system solver [J]. Journal of Computational and Applied Mathematics, 2009, 226(2): 306 ~ 318.

[11] S. C. S. Rao,Sarita. Parallel solution of large symmetric tridiagonal linear systems [J] . Parallel Computing,2008,34(3):177 ~ 197.
[12] M. A. Bender,G. S. Brodal,R. Fagerberg,et al. Optimal Sparse Matrix Dense Vector Multiplication in the I/O Model [C] . SPAA'07,San Diego, USA,2007.
[13] R. Jacob,M. Schnupp. Experimental Performance of I/O optimal Sparse Matrix Dense Vector Multiplication Algorithms within Main Memory [C] . Para 2010,Reykjavik,Iceland,2010.
[14] A. N. Yzelman,R. H. Bisseling. Cache Oblivious Sparse Matrix Vector Multiplication By Using Sparse Matrix Partitioning Methods [J] . Society for Industrial and Applied Mathematics,2009,31(4):3128 ~ 3154.
[15] J. C. Pichel, D. B. Heras, J. C. Cabaleiro, et al. Rivera. Improving the Locality of the Sparse Matrix Vector Product on Shared Memory Multiprocessors [C] . EUROMICRO-PDP'04,Coruna,Spain, 2004.
[16] O. Ternam,W. Jalby. Characterizing the behavior of sparse algorithms on caches [C] . SC'92,Minneapolis,USA,1992.
[17] G. E. Blelloch,P. B. Gibbons,H. V. Simhadri,et al. Low Depth Cache Oblivious Algorithms [C] . SPAA'10,Santorini,Greece,2010.
[18] A. Pinar,M. T. Heath. Improving Performance of Sparse Matrix Vector Multiplication [C] . SC 1999,Portland,USA,1999.
[19] E. J. Im,K. Yelick,R. Vuduc,et al. Sparsity: Optimization Framework for Sparse Matrix Kernels [J] . International Journal of High Performance Computing Applications,2004 18(1):135 ~ 158.
[20] A. N. Yzelmana,R. H. Bisselinga. Two dimensional cache oblivious sparse matrix vector multiplication [J] . Parallel Computing,2011,37(12):806 ~ 819.
[21] R. W. Vuduc,A. Gyulassy,J. W. Demmel,et al. Memory Hierarchy Optimizations and Performance Bounds for Sparse A^TAx [C] . ICCS 2003, Krakow,Poland,2003,2659:705 ~ 714.
[22] R. W. Vuduc,H. J. Moon. Fast sparse matrix-vector multiplication by exploiting variable block structure [C] . HPCC'05,Sorrento,Italy,2005.
[23] R. W. Vuduc, J. W. Demmel, K. A. Yelick, et al. Performance Optimizations and Bounds for Sparse Matrix-Vector Multiply [C] . SC'02, Baltimore,USA,2002.
[24] R. W. Vuduc,J. W. Demmel. Automatic Performance Tuning of Sparse Matrix Kernels [D] . California: University of California,Berkeley,2003.
[25] R. W. Vuduc,J. W. Demmel,K. A. Yelick,et al. OSKI: A library of automatically tuned sparse matrix kernels [C] . Proceedings of SciDAC 2005, San,Francisc,USA,2005.
[26] R. W. Nishtala, R. W. Vuduc, J. W. Demmel, et al. Yelick. Performance Modeling and Analysis of Cache Blocking in Sparse Matrix Vector Multiply [R] . California: University of California,Berkeley,2004.
[27] R. Nishtala,R. W. Vuduc,J. W. Demmel,et al. When cache blocking of sparse matrix vector multiply works and why [J] . Applicable Algebra in Engineering,Communication and Computing,2007,18(3):297 ~ 311.
[28] B. C. Lee,R. W. Vuduc,J. W. Demmel,et al. Yelick. Performance Models for Evaluation and Automatic Tuning of Symmetric Sparse Matrix-Vector Multiply [C] . ICPP'04,Quebec,Canada,2004.
[29] J. C. Pichel,D. B. Heras,J. C. Cabaleiro,et al. Improving the Locality of the Sparse Matrix-Vector Product on Shared Memory Multiprocessors [C] . EUROMICRO-PDP'04,A Coruna,Spain,2002.
[30] A. Buluç,J. R. Gilbert. On the Representation and Multiplication of Hypersparse Matrices [C] . IPDPS'08,Florida,USA,2008.
[31] A. Buluç,J. R. Gilbert. Highly Parallel Sparse Matrix-Matrix Multiplication [J] . Computing Research Repository,2010,abs/1006:2183 ~ 2421.
[32] A. Buluç,J. R. Gilbert. Parallel Sparse Matrix-Matrix Multiplication And Indexing: Implementation And Experiments [J] . Society for Industrial and Applied Mathematics,2011,34(4):170 ~ 191.
[33] A. Buluç,J. T. Fineman,M. Frigo,et al. Parallel Sparse Matrix-Vector and Matrix-Transpose-Vector Multiplication Using Compressed Sparse Blocks [C] . SPAA'09,Calgary,Canada,2009.
[34] V. H. F. Batista,G. O. A. Jr,F. L. B. Ribeiro,et al. Parallel structurally-symmetric sparse matrix-vector products on multi-core processors [J] . Computing Research Repository,2010,abs/1003.0:1 ~ 17.
[35] S. Liu,Y. Zhang,X. Z. Sun,et al. Performance Evaluation of Multithreaded Sparse Matrix-Vector Multiplication using OpenMP [C] . 11th IEEE International Conference on High Performance Computing and Communications,Seoul,Korea,2009.
[36] G. E. Blelloch, M. A. Heroux, M. Zagha, et al. Segmented Operations for Sparse Matrix Computation on Vector Multiprocessors [R] . CMU: Pennsylvania,1993.
[37] N. Zhang. A Novel Parallel Scan for Multi-Core Processors and Its Application in Sparse Matrix-Vector Multiplication [J] . IEEE Transactions on Parallel And Distributed Systems,2012,23(3):397 ~ 404.
[38] G. Goumas,K. Kourtis,N. Anastopoulos,et al. Understanding the Performance of Sparse Matrix-Vector Multiplication [C] . Proceedings of the 16th Euromicro Conference on Parallel,Distributed and Network-Based Processing,Toulouse,France,2008.
[39] 袁　娥,张云泉,刘芳芳,等. SpMV 的自动性能优化实现技术及其应用研究 [J] . 计算机研究与发展,2009,46(7):1117 ~ 1126.
[40] J. W. Hong,H. T. Kung. I/O Complexity: The Red-Blue Pebble Game [C] . In STOC'81: proceedings of the thirteenth annual ACM symposium on Theory of Computing,New York,USA,1981:326 ~ 333.
[41] D. Ironya,S. Toledoa,A. Tiskinb,et al. Communication lower bounds for distributed-memory matrix multiplication [J] . Journal of Parallel and Distributed Computing,2004,64(9):1017 ~ 1026.
[42] L. H. Loomis,H. Whitney. An inequality related to the isoperimetric inequality [J] . Bulletin of The American Mathematical Society,1949,55 (1949):961 ~ 962.
[43] G. Ballard, J. Demmel, O. Holtz, et al. Minimizing Communication In Numerical Linear Algebra [J] . SIAM Journal on Matrix Analysis and Applications,2011,32(3):866 ~ 901.

[44] G. Ballard, J. Demmel, O. Holtz, et al. Graph Expansion and Communication Costs of Fast Matrix Multiplication [C]. SPAA'11, California, USA, 2011.
[45] G. Ballard, J. Demmel, O. Holtz, et al. Communication-Optimal Parallel and Sequential Cholesky Decomposition [C]. SPAA'09, Calgary, Canada, 2009.
[46] G. Ballard, J. Demmel, A. Gearhart, et al. Communication Bounds for Heterogeneous Architectures [C]. SPAA'11, California, USA, 2011.
[47] J. Demmel, M. Hoemmen, M. Mohiyuddin, et al. Avoiding Communication in Sparse Matrix Computations [C]. IPDPS 2008, Florida, USA, 2008.
[48] L. Grigori, J. W. Demmel, H. Xiang, et al. Communication Avoiding Gaussian Elemination [C]. SC'08, Austin, USA, 2008.
[49] L. Grigori, J. W. Demmel, H. Xiang, et al. CALU: A Communication Optimal LU Factorization Algorithm [J]. SIAM Journal on Matrix Analysis and Applications, 2011, 32(4): 1317 ~ 1350.
[50] M. Mohiyuddin, M. Hoemmen, J. Demmel, et al. Minimizing Communication in Sparse Matrix Solvers [C]. SC09, Portland, USA, 2009.
[51] E. Solomonik, J. Demmel. Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms [C]. Euro-Par 2011 Parallel Processing, Lecture Notes in Computer Science, Springer Berlin Heidelberg, Germany, 2011.
[52] L. Grigori, P. Y. David, J. W. Demmel, et al. Brief Announcement: Lower Bounds on Communication for Sparse Cholesky Factorization of a Model Problem [C]. SPAA'10, Santorini, Greece, 2010.
[53] U. V. Catalyurek, C. Aykanat. Hypergraph-Partitioning-Based Decomposition for Parallel Sparse-Matrix Vector Multiplication [J]. IEEE Transaction on Parallel and Distributed System, 1999, 10(7): 673 ~ 693.
[54] B. Vastenhouw, R. H. Bisseling. A Two-Dimensional Data Distribution Method for Parallel Sparse Matrix-Vector Multiplication [J]. SIAM Review, 2005, 47(1): 67 ~ 95.
[55] A. Buluç, S. Williams, L. Oliker, et al. Reduced-Bandwidth Multithreaded Algorithms for Sparse Matrix-Vector Multiplication [C]. IPDPS'2011, Anchorage, USA, 2011.
[56] K. Kourtis, G. Goumas, N. Koziris, et al. Improving the Performance of Multithreaded Sparse Matrix-Vector Multiplication using Index and Value Compression [C]. 37th International Conference on Parallel Processing, Portland, USA, 2008.
[57] K. Kourtis, G. Goumas, N. Koziris, et al. Exploiting Compression Opportunities to Improve SpM × V Performance on Shared Memory Systems [J]. ACM Transactions on Architecture and Code Optimization, 2010, 7(3): 1 ~ 31.
[58] V. Karakasis, G. Goumas, N. Koziris, et al. Exploring the Effect of Block Shapes on the Performance of Sparse Kernels [C]. IPDPS'09, Rome, Italy, 2009. [59] S. Williams, Leonid Oliker, Richard Vuduc, et al. Optimization of Sparse Matrix-Vector Multiplication on Emerging Multicore Platforms [C]. SC'07, Reno, USA, 2007.
[60] G. Goumas, K. Kourtis, N. Anastopoulos, et al. Performance evaluation of the sparse matrix-vector multiplication on modern architectures [J]. Journal of SuperComputing, 2009, 50(1): 36 ~ 77.
[61] M. Belgin, G. Back, C. J. Ribbens, et al. Pattern-based Sparse Matrix Representation for Memory-Efficient SMVM Kernels [C]. Proceedings of the 23rd international Conference on Supercomputing, Yorktown Heights, USA, 2009.
[62] D. D. Brahme, B. R. Mishra, A. Barve, et al. Parallel Sparse Matrix Vector Multiplication using Greedy Extraction of Boxes [C]. 2010 International Conference on High Performance Computing, Panaji, Goa, India, 2010.
[63] J. Willcock, A. Lumsdaine. Accelerating Sparse Matrix Computations via Data Compression [C]. ICS '06, Cairns, Australia, 2006.
[64] C. Lessig, P. Bientinesi. On Parallelizing the MRRR Algorithm for Data-Parallel Coprocessors [C]. PPAM 2010, Wroclaw, Poland, 2010.
[65] Y. Zhang, J. Cohen, J. D. Owens, et al. Fast Tridiagonal Solvers on the GPU [C]. PPoPP 2010, Bangalore, India, 2010.
[66] E. Agullo, C. Augonnet, J. Dongarray, et al. QR Factorization on a Multicore Node Enhanced with Multiple GPU Accelerators [C]. IPDPS 2011, Anchorage, USA, 2011.
[67] C. Augonnet, S. Thibaulty, R. Namyst, et al. StarPU: a Runtime System for Scheduling Tasks over Accelerator-Based Multicore Machines [J/OL]. [2015-7-14] https://hal.inria.fr/inria-00467677/document.
[68] S. Tomov, R. Nath, H. Ltaief, et al. Dense Linear Algebra Solvers for Multicore with GPU Accelerators [C]. Proc. of IPDPS'10, Atlanta, USA, 2010. [69] J. M. Wolfe. Reducing truncation errors by programming [J]. Communications of the ACM, 1964 7(6): 355 ~ 356.
[70] K. E. Hillesland, A. Lastra. GPU Floating-Point Paranoia [C]. proceedings of GP2, Los Angeles, USA, 2004.
[71] N. Maruyama, A. Nukada, S. Matsuoka, et al. Software-Based ECC for GPUs [C]. Sysposium on Application Accelerators in High Performance Computing, Urbana-Chapaign, Illinois, USA, 2009.
[72] A. G. Anderson, W. A. Goddard, P. Schroder, et al. Quantum Monte Carlo on Graphical Processing Units [J]. Computer Physics Communications, 2007, 177: 298 ~ 306.
[73] D. Goldberg. What every computer scientist should know about floating-point arithmetic [J]. ACM Computing Surveys, 1991, 23(1): 5 ~ 48.
[74] W. Kahan. Pracniques: further remarks on reducing truncation errors [J]. Communications of the ACM, 1965, 8(1): 40, 48.
[75] E. S. Larsen, D. Mcallister. Fast Matrix Multiplies using Graphics Hardware [C]. SC2001, Denver, USA, 2001.
[76] D. Goddeke, R. Strzoka, S. Turek, et al. Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations [J]. International Journal of Parallel Emergent and Distributed Systems, 2007, 22(4): 221 ~ 256.
[77] D. Goddeke, R. Strzoka, S. Turek, et al. Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations(Part 2: Double Precision GPUs) [J/OL]. [2013-7-15]. http://gpgpu.org/2008/07/14/performance-and-accuracy-of-hardware-oriented-native-emulated-and-mixed-precision-solvers-in-fem-simulations-part-2-double-precision-gpus.
[78] M. Baboulin, A. Buttari, J. Dongarra, et al. Accelerating scientific computations with mixed precision algorithms [J]. Computer Physics Communications, 2009, 180(12): 2526 ~ 2533.

(下转第 47 页)

[6] Kaya D. The exact and numerical solitary-wave solutions for generalized modified Boussinesq equation [J]. Phys. Lett. A,2006,348:244 ~ 250.
[7] Parker A. On exact solutions of the regularied long-wave equation: A direct approach to partially integrable Equation,1. solitary wave and solutions [J]. J. Math. Phys.,1995,36:3498 ~ 3505.
[8] Li J. B.,Zhang L. J. Bifurcations of traveling wave solutions in generalized Pochhammer-Chree equation [J]. Chaos,Solitons and Fractals,2002,14:581 ~ 593.
[9] Rafei M.,Ganji D. D. Mohammadi Daniali H. R. Pashaei H. Application of homotopy perturbation method to the RLM and generalized modified Boussinesq equationsf [J]. Phys. Lett. A,2007,364:1 ~ 6.
[10] 马松华,方建平. 联立薛定谔系统新精确解及其所描述的孤子脉冲和时间孤子 [J]. 物理学报,2006,55(11):5611 ~ 5616.
[11] X. Q. Liu,S. Jiang. The sech-tanh method and its applications [J]. Phys. Lett.,A,2002,298:253 ~ 258.
[12] 李画眉,林　机,许友生,等. 两组新的广义 Ito 方程组的多种行波解 [J]. 物理学报,2004,53(2):349 ~ 355.
[13] Senyue Lou et al. Exact solitary wave in a convecting fluid [J]. Phys. A: Math. Gen.,1991,24:587 ~ 590.

Study on the Exact Travelling Wave Solutions of the Nonlinear Coupled KdV Equations

JING Shu-jie,ZHAO Jian-wei,WANG Shi-lei

(Mathematics and information science academy,Henan Polytechnic University,Jiaozuo 454000,China)

Abstract: When auxiliary functions method is used to solve the nonlinear coupling KdV equations, solving nonlinear partial differential equations of the problem will be transformed into solving algebraic equations. By the further application of the Maple software get ten exact travelling wave solutions, which contain hyperbolic function, Jacobi elliptic function, trigonometric function, rational function and so on. At last, some figures of the obtained solutions was given using the Maple software.

Key words: auxiliary functions; nonlinear coupled KdV equations; exact travelling wave solution

(责任编辑: 梁怀学)

(上接第 40 页)

[79] R. Strzodka,D. Goddeke. Mixed Precision Methods on GPUs [C]. NVISION,SAN JOSE,2008.
[80] J. W. Choi,A. Singh,R. W. Vuduc,et al. Model-driven Autotuning of Sparse Matrix-Vector Multiply on GPUs [C]. PPoPP 2010,Bangalore,India,2010.
[81] B. Nathan,G. Michael. Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors [C]. SC'09,Portland,USA,2009.
[82] Nvidia. CUDA Toolkit 5.0 CUSPARSE Library [EB/OL]. (2012-05-06) [2013-05-20]. https://developer.nvidia.com/cuda-toolkit.
[83] Y. D. Deng,B. D. Wang,S. Mu,et al. Taming irregular EDA applications on GPUs [C]. 2009 International Conference on Computer-Aided Design,San Jose,USA,2009.
[84] B. Y. Su,K. Keutzer. clSpMV: A Cross-Platform OpenCL SpMV Framework on GPUs [C]. ICS' 2012,Venice,Italy,2012.
[85] Khronos Opencl Working Group. The OpenCL Specification [S/OL]. [2015-7-15]. https://www.khronos.org/registry/cl/specs/opencl-1.1.pdf.
[86] W. T. Tang,W. J. Tan,R. Rayy,et al. Accelerating Sparse Matrix-Vector Multiplication on GPUs using Bit-Representation-Optimized Schemes [C]. SC2013,Denver,USA,2013.
[87] 高文. 服务器聚集系统中高可用性分析与设计方法 [D]. 北京: 中国科学院研究生院(计算技术研究所),2000.
[88] S. G. Yan,Chao Li,Yunquan Zhang,et al. yaSpMV: Yet Another SpMV Framework on GPUs [C]. PPoPP'14,Orlando,USA,2014.

Research on Linear Algebra Algorithm on Multi-core and Many-core Architecture Platform

TAO Yuan[1]*,ZHU Ming-fa*[2]

(1. College of Mathematics,Jilin Normal University,Siping 136000,China;
2. School of Computer Science and Engineerig,Beihang University,Beijing 100191,China)

Abstract: Linear Alebra Algorithms are the key of science and engineer computing. How to get higher performance for the existing Linear Algebra Algorithms on the platform is the problem to be solved, because the scale of the hardware is much larger and the architecture is much more complexity. In the paper, the current situation, the problem and challenge of the Linear Algebra Algorithm on the heterogeneous architecture platform between multi-core CPU and many-core GPU were reviewed. The problem was systematically analysed and the direction in future work was given.

Key words: Multi-core CPU; Many-core GPU; Linear Algebra Algorithm

(责任编辑: 梁怀学)