基于GPU的高性能稀疏矩阵向量乘及CG求解器优化

王迎瑞 任江勇 田 荣
(中国科学院计算技术研究所 北京 100190)

摘 要 以有限元/有限差分等为代表的一类数值方法,其总体矩阵常常具有"带状"、稀疏的特点。针对"带状"稀疏矩阵,提出和实现了一种高效的矩阵向量乘存储格式和算法"bDIA"。基于nVidia的GTX280系列GPU对其进行了测试,结果显示:与CUSP支持的5种常见稀疏矩阵存储格式和算法相比较,所提出的bDIA格式以及相应的spMV算法的单双精度浮点效率均可以提高1倍以上,并突破了该系列GPU在spMV计算时4%的单精度浮点效率上限和22.2%的双精度浮点效率上限;应用于共轭梯度(CG)与稳定双共轭梯度(BiCGStab)求解器,相对于DIA格式均有1.5倍左右的加速。

关键词 带状稀疏矩阵向量乘,bDIA,广义有限元,GPU,CG求解器优化
中图法分类号 TP301 **文献标识码** A

Efficient Sparse Matrix-vector Multiplication and CG Solver Optimization on GPU

WANG Ying-rui REN Jiang-yong TIAN Rong
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

Abstract Numerical methods of PDEs are mostly "compactly supported", say finite element, and finite difference methods etc. Due to the compact support, the global matrix associated with those numerical methods for scientific and engineering are sparse and very often also band shaped. We proposed and developed a high performance spMV algorithm for this specific but widely used sparse matrix type. The new algorithm, termed "bDIA (banded diagnal)", is implemented on NVIDIA GTX 285. Detailed comparisons with the five other mostly used sparse matrix formats/algorithms supported in the open source cuda linear algebra library (CUSP) show that bDIA doubles the best performance of the other five algorithms, breaking the float-point efficiency limit of 4% for single precision and 22.2% for double-precision. The conjugate gradient (CG) and the bi-conjugate gradient stabilized (BiCGStab) solvers both gain a speedup of around 1.5 using the proposed "bDIA" format/algorithm.

Keywords Banded sparse matrix-vector multiplication, bDIA, GFEM, GPU, CG solver optimization

偏微分方程的数值求解所采用的数值方法,如有限元、有限差分等,大多具有"局部紧支"的特点,而"局部紧支"导致的总体(刚度)矩阵是稀疏矩阵。流体、固体、热、电磁等众多物理问题,最后通常都表现为求解大型线性方程组。出于对内存占用的考虑,迭代法在求解大型线性方程组中占有绝对主导地位,而spMV又是迭代算法的核心,集中了约90%的计算量。因此,稀疏矩阵向量乘(spMV)被称为科学与工程计算领域的"七个小矮人"之一[1]。

在众核时代,访存受限问题是高性能系统的主要问题[2]。矩阵向量乘对$O(n^2)$的数据进行$O(n^2)$次运算,访存/计算比近于1:1。由于计算机硬件访存能力落后于计算能力,访存带宽成为矩阵向量乘运算的瓶颈。矩阵的稀疏性又导致访存不连续,使得访存瓶颈问题更加突出。所以,稀疏矩阵向量乘法是一个典型的访存密集型问题,它完全受限于硬件访存能力。提高浮点效率,即提高单位访存的计算密集度具有重要的实际意义。

1 带状稀疏矩阵向量乘简介

以有限元/有限差分等为代表的一类数值方法,其总体矩阵常常不仅稀疏,而且具有"带状"特点,如图1所示。这一类矩阵多见于结构化网格和规则求解域上的非结构化网格。图中d标记每一行中非零元素的带宽,n标记矩阵的大小,d/n可以简单地表示矩阵的稀疏性,d/n越小,矩阵越稀疏。对于一个$n\times n\times n$的结构化网格,当问题规模n很大时,$d/n\sim O(1/n)$。

目前流行的spMV存储格式与算法中,对角格式DIA主要针对结构化网格的有限元/有限差分方法,这类方法导致的总体矩阵为对角矩阵。非结构化网格相应的总体矩阵是更一般化的"带状"稀疏矩阵。本文针对矩阵的"带状"特征,基于"DIA"发展了一种新的稀疏矩阵向量乘算法"bDIA",其中"b"

到稿日期:2012-10-19 返修日期:2012-12-25 本文受国家自然科学基金项目(11072241,91130026),NSFC国家杰出青年科学基金"Exascale计算的基础研究"项目(60925009),美国橡树岭国家实验室/国家计算科学中心主任基金项目(MAT028)资助。
王迎瑞(1989—),女,硕士,主要研究领域为高性能计算机应用开发、并行计算;任江勇(1981—),男,硕士,助理研究员,主要研究领域为并行计算、混合精度计算;田 荣(1974—),男,博士,研究员,主要研究领域为数值分析、大规模可容错应用算法、多尺度模拟、混合精度计算。

指带状(banded)。

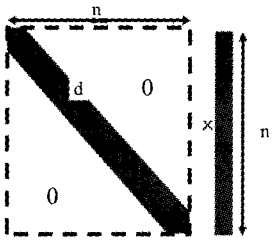


图 1　带状稀疏矩阵

2　spMV 算法性能测试现状及分析

2.1　spMV 性能测试现状

目前，常见的稀疏矩阵存储格式有对角格式 DIA、ELL-PACK 格式 ELL、行压缩格式 CSR、元素坐标格式 COO、混合格式 HYB 和块行压缩格式 BCSR 等。nVidia 对前 5 种存储格式的 spMV 在 GTX 280 系列 GPU 上的测试数据[3] 如表 1 所列。在所有数据存储格式中，spMV 实测峰值最高的数据格式为“DIA”格式，单、双精度实测峰值分别为 36.8 GFlops 和 16.7 GFlops，分别为理论峰值的 3.46%和 18.85%。

表 1　nVidia GTX280 系列 GPU 上单、双精度 spMV 性能测试[3]

存储格式	单精度实测峰值 (GFlops)	双精度实测峰值 (Gflops)	单精度效率 (%)	双精度效率 (%)
DIA	36.8	16.7	3.46	18.85
ELL	25.2	13.5	2.37	15.24
CSR(scalar)	7.6	5.0	0.71	5.64
CSR(vector)	8.8	4.1	0.83	4.63
COO	6.2	4.1	0.58	4.63

2.2　性能受限的基本原因

从测试结果可以看出，spMV 浮点效率非常低下。而这种低下的 spMV 浮点运算性能决定了整个隐格式有限元、有限差分的浮点运算效率或者说这一类科学与工程应用的整体浮点性能。分析其性能受限的基本原因，大致可分为：

(1) 矩阵元素缺乏时间局部性。在一次矩阵向量乘中矩阵元素只使用一次，不具有可重用性，这是 spMV 与 DGEMM(Linpack 测试核心算法)相比性能大幅度下降的主要原因[4,5]。

(2) 间接内存引用。为了节省内存空间和浮点运算，使非零元素连续地存储在内存中，人们通常构建额外的数据结构作为索引，然后通过索引进行间接访问，这进一步降低了内核的性能[6]。

(3) 向量的不规则访问。稠密矩阵向量乘时，对向量的访问是顺序的。而稀疏矩阵向量乘，对向量的访问是不规则的。这种不规则性使得向量的空间重用过程更为复杂[7-9]。相比于矩阵，向量元素少，因此在 CPU 上向量的不规则访问影响不大[1]，但是 GPU 全局存储不一致访问会带来访存性能量级上的下降，向量的不规则访问问题便变得重要。

(4) 向量的重复访问。spMV 计算的向量元素会被多次重复读取，考虑通过 share memory 进行重用，但是，由于受限于 share memory 的大小(16kB)，完全重用也存在困难。

(5) 非零带宽较短。许多稀疏矩阵行非零元素较少，每次循环只执行了少量有用的计算，这相当于增加了循环的开销——当内层循环的计算数目较少时，外循环的开销就变得显著[4,10]。

3　高性能 spMV 的基本思想及算法实现

在上述影响 spMV 性能的 5 个因素中，矩阵元素缺乏时间局部性是矩阵向量乘的固有属性，难以通过算法进行改进；间接内存引用问题以及向量的不规则访问可以通过设计更高效的数据压缩格式提高访存连续性来缓解；向量的重复访问需要设计合理的 share memory 优化策略来提高向量重用；而对于非零带宽较短的问题，由于非零带宽与数值方法离散阶次和网格形式相关，探索稀疏性可调的数值方法可以增大非零元素带宽，增加计算流水线深度，从而提高浮点计算性能(同时达到提高数值方法插值精度的双重目的)。

3.1　带状矩阵存储格式

在目前常见的存储格式中，对角格式 DIA 主要针对结构化网格有限元/有限差分算法产生的对角矩阵。对角矩阵的特点是若干对角元素为非零，在高维结构化网格有限差分计算中，零对角和非零对角会交替出现，如图 2 所示。

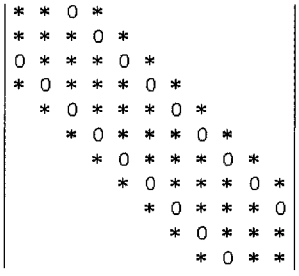


图 2　对角稀疏矩阵示意图(* 表示非零元素)

对于非结构化网格，非零元素带内的零元素的出现依赖于网格的几何拓扑和编号方式，相应的总体矩阵为更一般化的“带状”稀疏矩阵，如图 3 所示，矩阵的非零元素由“非零带宽”d 描述。“非零带宽”d 是指矩阵所有行中第一个非零元素与最后一个非零元素之间的距离(包含之间的零元素)的最大值。很明显，d 与网格的节点编号方式有关。但是，当矩阵维数 n 很大时，$d/n \sim O(1/n)$，d 仍然是相对于矩阵维数 n 的一个很小的数。

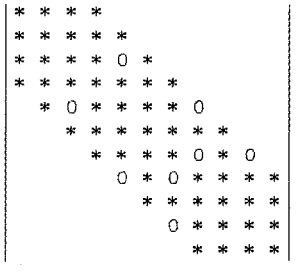
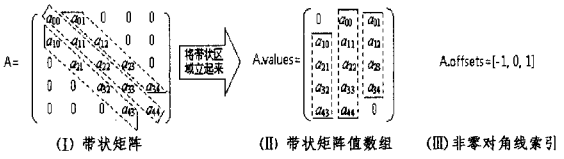


图 3　带状稀疏矩阵示意图(* 表示非零元素)

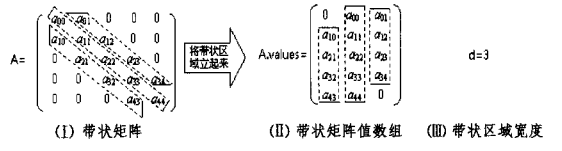
带状矩阵是对角矩阵的一般化情形，当网格为结构化网格时，带状矩阵便退化为对角矩阵。由于对应着非结构化网格这种应用更加广泛的离散形式，带状矩阵具有更广泛的普适性。

本文基于 DIA 格式，发展了更高效的面向带状稀疏矩阵的存储格式 bDIA。由于带状稀疏矩阵的结构特征，非零元素带连续存储，不必再存放非零元对角线的索引，只需要存储带宽参数即可。图 4 比较了 DIA 与 bDIA 格式的区别。DIA 格式使用一个 $n*c$ (c 表示非零对角线条数)的一维数组 *values*

按顺序存储每一条非零对角线上的元素，用长度为 c 的偏移量数组 $offset$ 表示非零元对角线相对于主对角线的偏移量[3]。bDIA 是用长度为 $n*d$（d 表示非零带宽）的一维数组 $values$ 按列存储非零元素带内的每一条对角线元素，并存储带宽参数 d。



(a) 带状稀疏矩阵的 DIA 存储格式



(b) 带状稀疏矩阵的 bDIA 存储格式

图 4　带状稀疏矩阵 DIA 与 bDIA 存储格式的比较

bDIA 的基本思想是放弃对非零元素带内的零元素的记录和索引，将非零元素带内所有元素都进行储存和计算（在非结构网格算法中非零元素带内零元素的位置的确定有时既困难又没有必要），从而实现矩阵元素的连续访问。同时，对同一非零带元素连续访问，各非零带连续排列，自然实现了对向量的连续访问。

3.2　提高向量重用

在矩阵向量乘中，向量各分量被多次访问，具有很高的可重用性。bDIA 存储格式为按列存储，每个 thread 计算一行结果分量。考虑到带状矩阵的结构特点，矩阵元素访问的连续性导致向量元素的访问也具有连续性，而相邻结果分量在同一并行计算步对向量的访问也具有连续性。这样，每个 block 所需要用到的向量的分量很有限，共 $BLOCKSIZE+d-1$ 个连续分量，可以加载到 share memory 中进行优化，如图 5 所示。

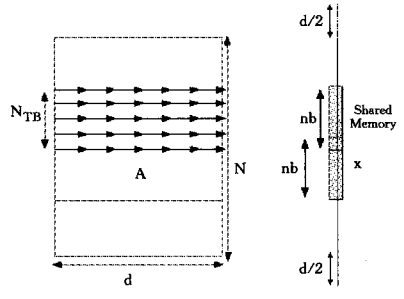
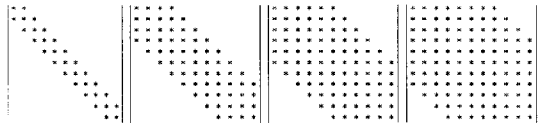


图 5　带状 spMV 的 share memory 优化

相邻的 block 用到的向量分量有一定的交错，分别是前后各 $d/2$ 个。为了实现加载 share memory 时对全局存储的一致访问，应虚拟地为向量首尾各添加 $d/2$ 个 0 进行逻辑转换，这样每个 block 所需要用到的向量分量的起始位置可以统一表示。N_{TB} 是每个 block 包含的 thread 数目，计算这 N_{TB} 个结果分量需要用到向量的 $N_{TB}+d-1$ 个分量，即 nb 个。对于第 i 个 block，它所需要的 nb 个分量起始于转换后的向量的 $N_{TB}*i$ 处，结束于 $N_{TB}*i+nb$ 处。

3.3　调节稀疏性

非零带宽是数值、几何离散格式内在的特点，一旦数值、离散格式固定，非零带宽 d 是无法改变的。为此，我们发展了一种稀疏性可调的广义有限元方法 GFEM[11,12]，即通过改变插值算法实现非零带宽 d 的调节。其基本特点是在给定的网格点数或自由度数不变（不改变内存占用）的前提下，可以通过改变插值格式来改变插值精度和总体矩阵的非零元素的宽度（见图 6）。因此，可以采用高阶插值格式使得非零带宽 d 增加。



(a) FEM　(b) GFEM r=h　(c) GFEM r=2h　(d) GFEM r=3h

图 6　稀疏性可调的有限元方法

3.4　GPU kernel 实现

Kernel 实现主要分为两个模块，首先将该 block 所需要用到的 $N_{TB}+d-1$ 个向量分量加载到 share memory 中：为了实现对全局存储的一致访问，每一轮所有 thread 连续地加载 N_{TB} 个分量，直到所有 $N_{TB}+d-1$ 个分量都读进来。接下来每个 thread 计算一行：每一轮从 values 非零元数组中取出一列矩阵对角线值，从 share memory 中取出对应的向量分量，进行乘加；然后从 values 取出下一列，同时对应的向量分量的起始位置也移动一位。该过程共进行 d 步（见图 7）。

```
Algorithm: BDIA kernel to compute y ← A·x
Input: n*n matrix A, stored in BDIA(n*d) format as
values[0...n*d-1]; vectors x as x[0...n-1], y as y[0...n-1]
Output: Modifies y
Let NTB=thread block size
Let tid=local thread ID
Let index=globle thread ID
Initialize As[NTB+d-1] in sharememory

//Load nb from x coalesced
for circle=0;index+circle*NTB<NTB+d-1;circle++ do
    if((index+circle*NTB<d/2)||(index+circle*NTB)>=n+d/2)
        As[index+circle*NTB]=0
    else
        As[index+circle*NTB]=x[index+circle*NTB-d/2]

//Compute
for each thread that index<n
    for i=0;i<d;i++ do
        value=values[index+n*i]
        vec=As[tx+i]
        result+=value*vec
    y[index]=result
```

图 7　带状 spMV 的 GPU kernel 实现

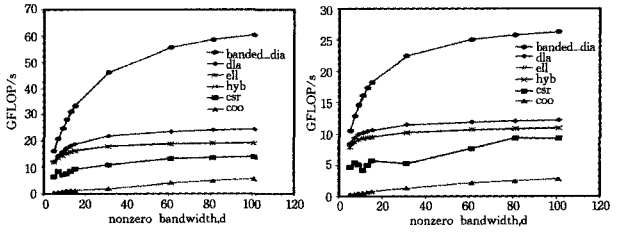
该 kernel 采用了本文提出的 bDIA 存储格式，提高了对矩阵非零元及向量的访存连续性；利用 share memory 实现向量重用；以 values 数组和非零带宽 d 为输入，结合稀疏性可调的广义有限元方法可以扩大非零元带宽 d，以增加计算流水线的深度。

bDIA 格式最终在开源的 CUDA 线性代数包 CUSP 的基础上实现。CUSP 支持目前流行的 5 种稀疏矩阵格式。这样，bDIA 就可以很方便地与现有其他稀疏矩阵的存储格式进行对比性测试。

4　实验与分析

实验平台是 GeForce GTX 285 GPU。对矩阵规模 $n=$

15600，d 从 3 到 101 变化的一系列带状稀疏矩阵进行测试，bDIA 与现有 5 种常用格式的性能相比较，单双精度测试结果如图 8(a)和图 8(b)所示。



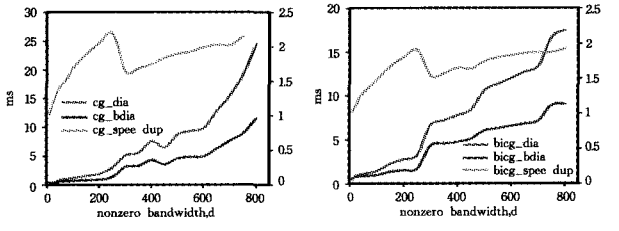
(a) 单精度　　(b)双精度

图 8　面向带状稀疏矩阵的高性能 spMV 算法(hyb 与 ell 重合)

图中横轴为非零带宽 d，代表矩阵稀疏性，d 越大，表示矩阵越相对稠密；纵轴为浮点性能 GFlop/s。从图中可以看出，随着非零带宽 d 的增大，所有存储格式的浮点性能均呈现近似对数的增长曲线。这反映了在一定范围内调节矩阵稀疏性，有利于提高 spMV 的浮点效率，发挥 GPU 的高浮点能力。对于不同的非零带宽，bDIA 相对于其他 5 中格式都有明显的性能提升。几种格式的性能表现为 bDIA＞DIA＞HYB≈ELL＞CSR＞COO，bDIA 有明显优势。在性能稳定后，bDIA 相对于其他 5 种常见格式中最好的结果有 1 倍的提高。

nVidia GTX 280 系列 GPU 全局存储访存带宽为159GB/s，而浮点运算能力理论峰值为单精度1062.72GFlops，双精度88.6GFlops，访存能力与计算能力的比值为单精度 1:27，双精度 1:4.5。因此，该系列 GPU 单精度浮点效率一般认为不超过 4%[3]，双精度浮点效率不超过22.2%。而 bDIA 通过设计向量的规则化访问以及高度重用，使得访存需求从 $n*d+n*d$ 降低为约 $n*d+n$，而计算次数仍为 $2*n*d$。优化后的算法访存/计算比接近 1∶2。n=15600，d=101 时，实测得的 bDIA 格式的 spMV 性能为单精度60.45GFlops(5.69%)，双精度 26.27GFlops(29.65%)。由于计算访存比的改善，bDIA 突破了目前 spMV 算法的浮点效率上限。

上述测试为 kernel 级运算，在应用级运算中，共轭梯度(CG)与稳定双共轭梯度(BiCGStab)是典型的迭代求解器，二者均以 spMV 为核心计算(约占 90%的计算量)，其性能很大程度上受限于 spMV 的浮点效率。应用级实现存在调用切换等额外开销，其效率一般低于 kernel 级实现，但相应测试结果真正体现了算法在实际应用中的有效性。测试选取 n=30000、d 变化的一组矩阵，结果如图 9 所示。



(a) CG 求解器　　(b) BiCGStab 求解器

图 9　CG/BiCGStab 求解器中 bDIA 与 DIA 格式的比较

图中横轴为非零带宽 d，左侧纵轴表示运算时间，右侧纵轴表示加速比。蓝色曲线是 DIA 格式在 CG/BiCGStab 求解器中的运算时间，红色曲线为 bDIA 格式的情况，绿色曲线为 bDIA 格式相对于 DIA 格式的加速比。从图中可知，应用 bDIA 格式进行求解，两种求解器的运算时间均大幅减少。测试结果验证了本文发展的高性能 spMV 算法对迭代求解线性方程组类应用的性能均有较大提高。

结束语　访存受限是基于 GPU 的 spMV 算法的性能瓶颈所在。本文提出和实现了带状稀疏矩阵的 bDIA 存储格式和相应的 spMV 算法，由于计算访存比的改善，bDIA 突破了目前 spMV 算法的浮点效率上限。数值实验结果表明，bDIA 的性能相比于其他流行的格式/算法的最佳性能有约 1 倍的提升，并且它在 CG、BiCGStab 迭代法求解线性方程组的应用中表现出色。同时，在(非零)带宽相对较小的一定范围内，各种存储格式随着带宽增长，其浮点运算性能明显提高，利用这一特点，我们可以通过应用算法如我们提出的、目前仍在发展中的稀疏性可调的广义有限元方法[11,12]调节矩阵稀疏性，以适应硬件计算能力。考虑到 spMV 在科学与工程计算中的重要地位，本文发展的高性能 spMV 应具有较为广泛的实用价值。

参 考 文 献

[1] Goumas G I, Kourtis K, Anastopoulos N, et al. Performance evaluation of the sparse matrix-vector multiplication on modern architectures[J]. The Journal of Supercomputing, 2009, 50(1): 36-77

[2] 范东睿. 众核发展趋势浅析[J]. 中国科学院计算技术研究所创新求实月刊, 2008, 72(1): 8-11

[3] Bell N, Garland M. Efficient Sparse Matrix-Vector Multiplication on CUDA [EB/OL]. http://www.nvidia.com/object/nvidia_research_pub_001.html, 2008

[4] Buttari A, Eijkhout V, Langou J, et al. Performance optimization and modeling of blocked sparse kernels[R]. Technical Report, 2007, 21(4): 467-484

[5] Mellor-Crummey J, Garvin J. Optimizing sparse matrix-vector product computations using unroll and jam[J]. Int J High Perform Comput Appl, 2004, 18(2): 225

[6] Pinar A, Heath MT. Improving performance of sparse matrix-vector multiplication[C]// Supercomputing' 99. Portland, OR, November 1999

[7] Geus R, Röllin S. Towards a fast parallel sparse matrix-vector multiplication[C]// Parallel Computing: Fundamentals and Applications, International Conference ParCo. Imperial College Press, 1999: 308-315

[8] Im E. Optimizing the performance of sparse matrix-vector multiplication[D]. University of California, Berkeley, 2000

[9] Pichel J C, Heras D B, Cabaleiro J C, et al. Improving the locality of the sparse matrixvector product on shared memory multiprocessors[C]// Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing. 2004: 66-71

[10] White J, Sadayappan P. On improving the performance of sparse matrix-vector multiplication[C]// 4th International Conference on High Performance Computing (HiPC '97). 1997

[11] Tian Rong. Generalized FEM without extra degrees of freedom [J]. International Journal for Numerical Methods in Engineering, 2012 (submitted for publication)

[12] 田荣. 面向百亿亿级计算协同设计的思考[J]. 中国科学院计算技术研究所信息技术快报, 2012, 10(3): 50-63