ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2012/06(02)-0118-07
DOI: 10.3778/j.issn.1673-9418.2012.02.003

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-51616056

大规模稀疏矩阵的主特征向量计算优化方法*

王 伟[1,2,3+], 陈建平[1,2,3], 曾国荪[1,2,3], 俞莉花[1,2,3], 谭一鸣[1,2,3]

1. 同济大学 计算机科学与技术系, 上海 200092
2. 国家高性能计算机工程技术中心 同济分中心, 上海 200092
3. 同济大学 嵌入式系统与服务计算教育部重点实验室, 上海 200092

Optimization of Parallel Principal Eigenvectors Computing for Large-Scale Sparse Matrixes*

WANG Wei[1,2,3+], CHEN Jianping[1,2,3], ZENG Guosun[1,2,3], YU Lihua[1,2,3], TAN Yiming[1,2,3]

1. Department of Computer Science and Engineering, Tongji University, Shanghai 200092, China
2. Tongji Branch, National Engineering & Technology Center of High Performance, Shanghai 200092, China
3. Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 200092, China

+ Corresponding author: E-mail: willtongji@gmail.com

WANG Wei, CHEN Jianping, ZENG Guosun, et al. Optimization of parallel principal eigenvectors computing for large-scale sparse matrixes. Journal of Frontiers of Computer Science and Technology, 2012, 6(2): 118–124.

Abstract：The principal eigenvectors computing (PEC) is a paramount operation in engineering and scientific computing. Since the general-purpose computing on graphics processing unit (GPGPU) emerges for the outstanding acceleration factors, PEC implementations on graphics processing unit (GPU) have appeared on the scene. This paper analyzes PEC performance bottleneck from the characteristic of application and GPU architecture, and therefore

*The National Natural Science Foundation of China under Grant Nos. 61103068, 61174158 (国家自然科学基金); the Joint Funds of NSFC and Microsoft Asia Research under Grant No. 60970155 (NSFC-微软亚洲研究院联合资助项目); the Doctoral Fund of Ministry of Education of China under Grant No. 20090072110035 (教育部博士点基金); the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20110072120017 (高等学校博士学科点专项科研基金); the Program of Shanghai Subject Chief Scientist under Grant No. 10XD1404400 (上海市优秀学科带头人计划项目); the Open Fund of State Key Laboratory of High-End Server & Storage Technology under Grant No. 2009HSSA06 (高效能服务器和存储技术国家重点实验室开放基金).

Received 2011-05, Accepted 2011-11.

proposes a new implementation of PEC based on a new matrix storage format, called GPU-ELL, and an optimized thread mapping strategy of GPU. It evaluates the proposed approach over ATI HD Radeon 5850 GPU, and the experimental results show its good performance with average 200 times acceleration of other existing algorithm on CPU, and 2 times of that on GPU.

Key words：general-purpose computing on graphics processing unit (GPGPU); principal eigenvectors computing (PEC); sparse matrix vector (SpMV); thread optimization

摘　要：矩阵主特征向量(principal eigenvectors computing, PEC)的求解是科学与工程计算中的一个重要问题。随着图形处理单元通用计算(general-purpose computing on graphics processing unit, GPGPU)的兴起，利用 GPU 来优化大规模稀疏矩阵的图形处理单元求解得到了广泛关注。分别从应用特征和 GPU 体系结构特征两方面分析了 PEC 运算的性能瓶颈，提出了一种面向 GPU 的稀疏矩阵存储格式——GPU-ELL 和一个针对 GPU 的线程优化映射策略，并设计了相应的 PEC 优化执行算法。在 ATI HD Radeon 5850 上的实验结果表明，相对于传统 CPU，该方案获得了最多 200 倍左右的加速，相对于已有 GPU 上的实现，也获得了 2 倍的加速。

关键词：图形处理单元通用计算(GPGPU)；主特征向量计算；稀疏矩阵向量乘；线程优化

文献标识码：A　　**中图分类号**：TP301

1　引言

在众多科学与工程计算中，如计算流体力学、统计学、结构工程等领域，经常需要数值求解矩阵的某种特征，例如广义特征值或高阶多项式特征值。然而在一些实际应用中，矩阵的维数往往可以达到几万维、甚至几百万维，并且矩阵常常为稀疏矩阵。例如，万维网 Web 网页 PageRank 值的计算，可归结为求解表示网页链接结构的邻接矩阵的主特征向量的问题[1]。大规模稀疏矩阵的主特征向量求解已经成为高性能计算领域的一个典型问题，引起了越来越多的研究者的关注。

乘幂法是一种常用的计算矩阵主特征向量的迭代方法，已得到广泛应用，并且特别适用于稀疏矩阵。但对于大规模稀疏矩阵来说，直接运用该方法往往效率不高，必须针对矩阵的稀疏性，并结合计算部件的特点，来设计高效的并行算法。

目前，包括图形处理单元(graphics processing unit, GPU)、现场可编程门阵列(field programmable gate array, FPGA)、数字信号处理器(digital signal processing, DSP)等在内的许多新型加速部件都被引入到高性能计算中。其中基于 GPU 的通用计算尤为突出，利用GPU进行高性能计算已经成为国内外的一个研究热点[2]。

本文提出了一种基于GPU的大规模稀疏矩阵主特征向量(principal eigenvectors computing, PEC)并行求解方案。该方案针对矩阵的稀疏特性，并利用 GPU 的并行计算能力，可以获得较高的加速比。本文首先实现了一个基于CPU的未优化的稀疏矩阵向量乘法(sparse matrix vector, SpMV)，经过对实验结果的分析，得到了影响 SpMV 性能的瓶颈，即稀疏矩阵的存储格式和线程映射策略；其次，结合应用特征和处理部件的性能实现了对 SpMV 的并行优化执行，包括设计了一种面向 GPU 的稀疏矩阵存储格式，以及提出了一个针对 GPU 的线程优化映射策略；最后，根据本文提出的优化执行方案，利用 AMD Radeon HD 5850 GPU 对多组实际应用中的稀疏矩阵进行了实验，并与 NVIDIA 的 SpMV 库和 LAPACK 的执行性能进行了对比。

2　矩阵主特征向量问题和乘幂法

设矩阵 $\boldsymbol{A}\in C^{n\times n}$，如果对于 $\lambda\in C$，存在非零向量 $\boldsymbol{x}\in C^{n}$ 使得 $\boldsymbol{Ax}=\lambda\boldsymbol{x}$，则称 λ 是 $\boldsymbol{A}$ 的特征值，$\boldsymbol{x}$ 是

A 属于 λ 的特征向量，求解方阵的特征值和特征向量的问题称为特征问题。

乘幂法是计算矩阵主特征值和主特征向量最简单的数值方法之一，也称为向量迭代法。乘幂法的基本思想是：给定一个非零初始向量 q，构造迭代序列 $q, Aq, A^2q, A^3q,...$。在实际序列计算中，因为序列中的每个向量都可以用 A 乘以前一个向量得到，即 $A^{j+1}q=A(A^jq)$，所以不需要显示地计算 A 的幂，这可以极大地节省计算量。可以证明，几乎对任意的向量 q，序列都收敛到主特征向量。因此，乘幂法计算的核心步骤是矩阵向量的乘积运算。乘幂法的一般步骤为：选择一个非零初始向量 q；用矩阵 A 乘以向量 q 得出新向量 Aq；重复这个过程，直到绝对误差到达可容忍的范围或是达到一定的迭代次数。

乘幂法求解主特征向量的计算量和存储量分别为 $O(kn^2)$和 $O(n^2)$，n 为矩阵维数，k 为迭代次数。当 n 很大时，计算量与存储量都十分巨大，特别是求解大规模稀疏矩阵中的特征向量问题，关键是对 SpMV 运算的优化。大规模稀疏矩阵的运算很不规则，原因是稀疏矩阵每行或每列的非零元素个数不同，导致循环的细粒度并行很难开发[3]。并且对矩阵数据访问的局部性不再保持，因为一次迭代中，矩阵中的每个元素只计算一次。因此，SpMV 的高性能计算需要考虑稀疏矩阵的特性和处理部件存储层次的特征。分析发现，乘幂法计算的性能瓶颈为：大规模稀疏矩阵向量乘为存储密集型应用，内存访问量大；大规模稀疏矩阵的运算很不规则。因此，SpMV 的优化执行首先需要设计一种合适的稀疏矩阵存储格式，减少计算量和存储量。此外，还需要对线程的映射策略进行优化，因为良好的映射策略可以减少线程访存的延迟时间。

3 矩阵主特征向量问题的并行性分析

目前，针对稀疏矩阵已经提出了许多不同的存储格式，每个存储格式实现的目标不同，比如为了存储的简易性、通用性、存储性能、或者适合于实现某个特殊的算法。本文重点考虑以下几种常用的存储格式：CRS(compressed row storage)、BSR(block sparse row)和 ELL(Ellpack-Itpack generalized diagonal)。这些存储格式最主要的不同点在于它们的稀疏模式，或者说是非零元素的结构特征不同，使得它们能适合于不同结构特征的稀疏矩阵[4−5]。尽管对于某个特定矩阵设计特殊的存储格式可以带来非常好的计算性能，但是设计通用的存储格式也十分重要。

本文通过对多组大规模稀疏矩阵数据进行实验，来分析 SpMV 的计算瓶颈。这些矩阵数据来自于佛罗里达大学的稀疏矩阵集[6]，且源于不同的真实应用环境，包括电路模拟、模型规约、线性规划和网络连接度分析等。这些矩阵数据不仅涉及的应用范围广，而且在矩阵的尺寸、密度和结构上也差别很大。表 1 列出了用来实验的 11 组数据。

首先分别使用 CRS、BSR 格式对表 1 中的数据进行实验，并和没有使用压缩格式的算法进行比较，每个算法所耗费的时间和性能各不相同，结果如图 1 所示。根据实验分析发现，大规模 SpMV 计算有以下特征：没有使用压缩格式的算法占用内存多，运行时间长；使用不同压缩格式的算法的执行效率和性能也不一样；SpMV 在运行时对内存访问量十分巨大，属于存储密集型应用，可以通过优化内存访问方式来提高性能。针对上述结果，本文拟针对 GPU 的计算特点对关键算核 SpMV 进行如下两点改进。

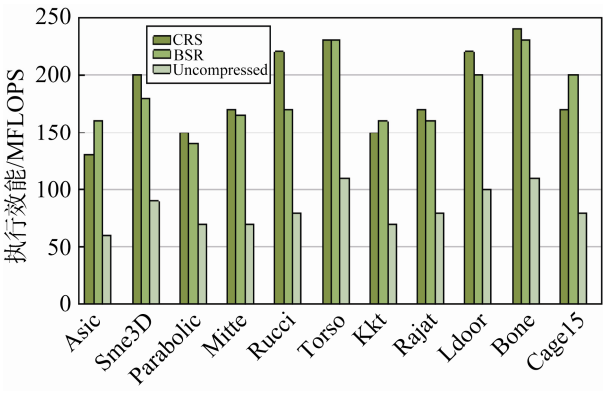
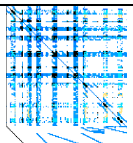
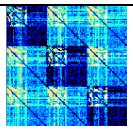
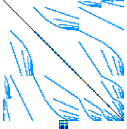
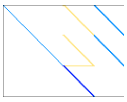
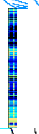
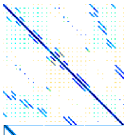
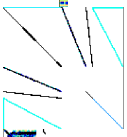
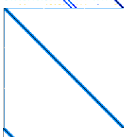
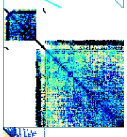
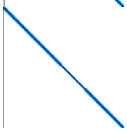
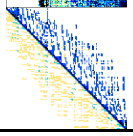


Fig.1 Comparison of CRS, BSR and uncompressed format

图 1 CRS、BSR 和无压缩格式的实验结果对比

Table 1　Matrix data sets for experiment

表 1　实验用的 11 组稀疏矩阵数据集

数据名称	数据形状	维数大小/维	数据名称	数据形状	维数大小/维
Asic_320K circuit simulation		321K×321K 1 931K	Sme3Dc 3D structural		42K×42K 3 148K
Parabolic_fem computational fluid dynamics		525K×525K 3 674K	Mittelmann linear programming		1 468K×1 961K 5 382K
Rucci least squares		1 977K×109K 7 791K	Torso 2D model		116K×116K 8 516K
Kkt_power optimization		2 063K×2 063K 12 771K	Rajat31 circuit simulation		4 690K×4 690K 20 316K
Ldoor structural		952K×952K 42 493K	Bone010 model reduction		986K×986K 47 851K
Cage15 directed weighted graph		5 154K×5 154K 99 199K			

(1) 存储格式

SpMV 中具有的计算不规则性使得对它进行优化具有很大挑战。这些不规则性源于以下两种情况：数据访问位置并非固定不变和细粒度循环并行没有被开发。因此需要设计一种新的面向 GPU 的数据压缩格式来存储稀疏矩阵，使它能够最大限度地提高计算性能。

(2) 优化 SpMV 算核的存储访问方式

矩阵向量相乘是一个存储密集型的应用，矩阵中的每个元素从内存中取出后只计算一次。因此每个浮点数操作中的内存访问负载很高。当矩阵是稀疏矩阵时，由于间接的和不规则的内存访问，使得它会引起更为复杂的存储负载。平均来说，当稀疏矩阵和向量相乘时，访问一个非零元素需要包含两次以上的存储操作，从而加重了存储负载。

4　基于 GPU 的 PEC 高效实现方案

4.1　面向 GPU 的稀疏矩阵的存储格式

本文提出的 GPU-ELL 是对 ELL 存储格式的扩展，包含 A[](float)和 j[](integer)两个数组，维数为 $N\times MaxEntriesbyRows$。此外，还包含了一个整型数组rl[]，维数为N，其目的是存储每一行的真实长度，其中并不包含 0 元素。同时，数组是以列优先的方式存储的。GPU-ELL 数据结构采取了如下设计思想：

(1) 合并存储器访问，采用列优先的方式来储存矩阵中的元素。线程通过索引 x 可以找到第 x 行的元素：$A[x+i\times N]$且$\{0\le i<rl[x]\}$。其中，i 是列号，$rl[x]$是 x 行的非零元素的个数。线程 x 和线程 x+1 访问连续的储存地址，满足了合并存储器访问的条件。

(2) 不同块之间的线程可以异步执行。每一个块中的线程可以异步执行，每一个线程只计算向量 $\boldsymbol{u}$ 的一个元素，并且在向量 $\boldsymbol{u}$ 中的不同元素之间不存在数据依赖。

(3) 降低等待时间和减少一个warp 中线程之间的不平衡。每一个 warp 的计算量不一样，只有 warp 相关行的长度大不相同时，才会造成较长的等待时间。

4.2 优化的线程映射策略

GPU 的体系结构包含了很多 SM(streaming multi-processor), 每一个 SM 包含了很多 SP(streaming processor)。GPU 中有不同层次的存储器, 包括片外全局存储、片外局部存储、片上共享存储、片上 cache 存储器、片上 cache 纹理存储器和片上寄存器。

全局存储器的容量大但是访问延迟也很大。共享存储位于 SM 中, 并且以 Bank 的形式进行组织, 当一个 Bank 同时有多个访问请求时, 会造成冲突, 且每一个 SM 中有很多寄存器。常数存储器和纹理存储器位于全局存储其中一部分只读区域, 它们都含有只读片上缓存。访问常数 cache 的速度很快, 但是它只有一个端口, 因此当多个 SP 需要 cache 中的同一个值时性能较好。虽然纹理 cache 的访问延迟要大于常数 cache, 但是当对存储的读操作不规则时, 并不能造成很大的性能损失。另外, 当访问的数据有 2D 空间局部性时性能较好。

GPU 计算时, 线程映射策略必须保证有充足的线程来隐藏全局存储的访问延迟。最佳的全局存储访问模式是当有连续的线程访问连续的元素时, 使用硬件优化的合并访问模式。因此, 需要有足够多的线程来对每一行的元素进行处理, 另外也需要实现一个连续性的映射, 从而实现最优的存储访问模式。本文提出的线程映射策略给每行映射多个线程, 使得连续的线程访问连续的非零元素, 可计算出非零元素的部分乘积。映射到一行的多个线程可以通过部分乘积的并行求和规约来计算输出向量元素。这些部分乘积的值被存入共享存储器中, 它们只被一个线程块中的线程所访问。

5 实验和结果分析

5.1 实验环境

本文的实验环境采用 AMD Radeon HD 5850 GPU 进行搭建。该 GPU 有 1 044 GFLOPS 的计算能力。如果乘加指令每个时钟周期执行两个浮点操作, 则可以使潜在的性能加倍。SpMV 算法代码正好有很多实例通过编译器产生乘加指令, 因此可以使性能大于 1 044 GFLOPS。

主机的配置为 Intel Pentium IV 3.00 GHz, 操作系统为 Microsoft Windows XP 32 位操作系统, 开发工具为 Microsoft Visual Studio 2008 Professional 版本。实现的语言为 OpenCL 1.0, 开发环境为 AMD ATI Stream SDK V2.2, 并安装了 ATI Stream SDK V2.2。

5.2 实验结果分析

利用本文的优化方法求解稀疏矩阵的主特征向量, 将表 1 中的稀疏矩阵作为实验数据, 并将实验结果和 NVIDIA 的 SpMV Library 和 LAPACK 中的 SpMV 进行了对比。

5.2.1 PEC 的加速比分析

首先比较了不同矩阵数据下 GPU 和 CPU 实现的性能(采用相同的压缩格式), 并根据 AMD 编译器产生的代码来计算 GPU 的 FLOPS 性能指标。FLOPS 指“每秒所执行的浮点运算次数”, 被用来估算处理部件(如 CPU 和 GPU)的执行效能。一个 GFLOPS 等于每秒 10^9 次的浮点运算。本文方法在 AMD 5850 GPU 上进行实现的最后运行结果如图 2 所示。

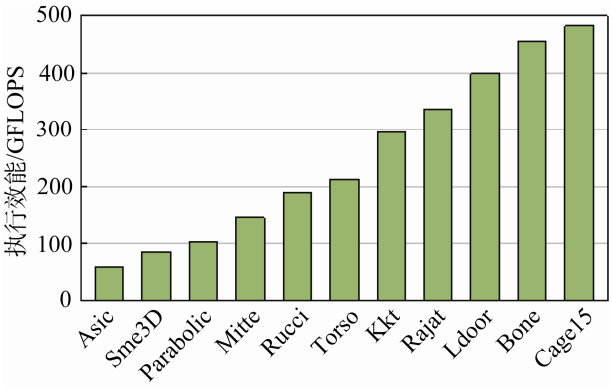


Fig.2 Performance on different data sets

图 2 在不同数据集上的性能

从图 2 中可以看到, 随着数据量的增大, GPU 的峰值也相应增加, 达到了 482 GFLOPS。相对于 CPU 来说, 通过计算可以得知, 最多可以获得 200 倍以上的加速度。在同样的设置下, 比较了 CPU 和 GPU 在不同矩阵数据集上完成 SpMV 所使用的时间, 如表 2 所示。从表中可以得到同样的结论, 随着数据规模的增大, 加速的效果越来越明显, 充分显示了本文方法适用于大规模问题。

Table 2　Comparison of running time on GPU and CPU

表 2　GPU 与 CPU 运行时间的比较

数据	大小/维	运行时间/s	
		CPU	GPU
Asic_320k	1 931K	5.854	0.020 8
Sme3Dc	3 148K	10.657	0.060 4
Parabolic_fem	3 674K	13.765	0.085 2
Mittelmann	5 382K	18.765	0.165 9
Rucci	7 791K	20.765	0.449 4
Torso	8 516K	28.521	0.609 4
Kkt_power	12 771K	41.864	1.068 4
Rajat31	20 316K	83.765	2.273 1
Ldoor	42 493K	>300	6.053 2
Bone010	47 851K	>300	7.156 8
Cage15	99 199K	>600	12.830 2

5.2.2　不同存储格式在 GPU 上的性能比较

对三种主要存储格式与本文提出的存储格式在 GPU 上的性能进行了比较。从图 3 中可以看出，本文的 GPU-ELL 压缩格式在所有的 11 个数据集上的性能是最好的，原因是 GPU-ELL 压缩格式充分考虑了 GPU 的体系结构和访存特征。

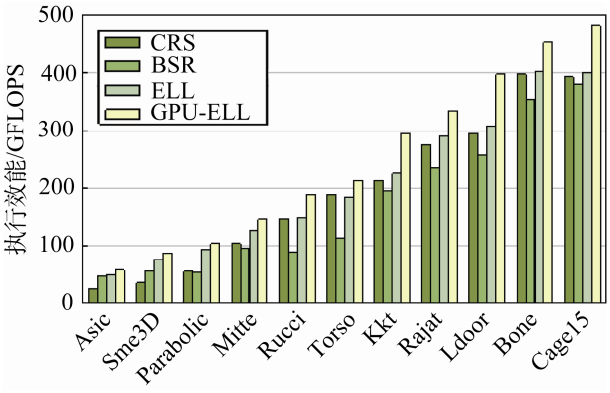


Fig.3　Performance comparison on different storage formats

图 3　不同存储格式上的性能比较

5.2.3　与已有 GPU 实现的比较

将本文方法分别同已有的 NVIDIA SpMV 和 LAPACK SpMV 进行比较，如图 4 所示。从图 4 中可以看出，相对于已有的 NVIDIA SpMV 和 LAPACK SpMV 来说，本文方法在不同数据集上性能均是最好的，充分体现了本文的优化线程映射策略和 GPU-ELL 稀疏矩阵存储格式的优越性。

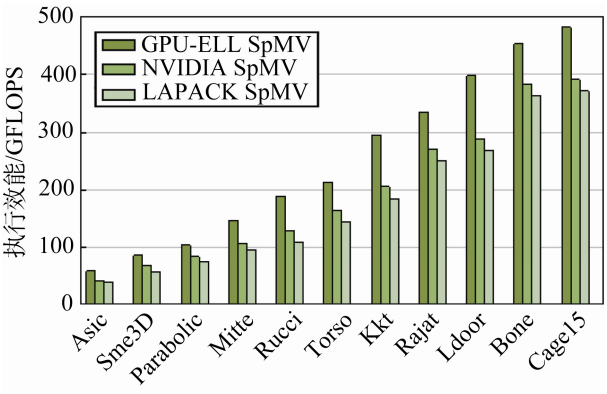


Fig.4　Performance comparison of existing GPUs

图 4　与已有 GPU 实现的性能比较

6　相关工作

近年来，关于稀疏矩阵向量乘法(SpMV)的优化工作成为国内外的一个研究热点。Vazquez 等人[5]针对 GPU 的体系结构提出了一种新的稀疏矩阵存储格式 ELLPACK-R，并将 ELLPACK-R 的执行性能和其他存储格式的性能进行了比较。Baskaran 等人[7]分析了在 NVIDIA GPU 上使用 CUDA (compute unified device architecture)编程模型开发 SpMV 高性能执行的性能瓶颈，并提出了相应的优化策略。Bell 等人[8]阐述了几种使用 CUDA 来优化执行 SpMV 的方案。这些方案通过开发细粒度并行来有效利用 GPU 的计算能力。

7　总结

本文分别从应用特征和GPU体系结构特征两方面分析了 PEC 运算的性能瓶颈，从而提出了一种新的稀疏矩阵存储格式——GPU-ELL 和一个针对 GPU 的线程优化映射策略，并设计了相应的 PEC 优化执行算法，在 AMD 的 HD Radeon 5850 上进行了实现。实验结果表明，相对于其他基于 CPU 的 PEC 算法，获得了最多 200 倍的加速；相对于已有 GPU 上的实现，也获得了 2 倍的加速。

References:

[1] Langville A N, Meyer C D. A survey of eigenvector methods for Web information retrieval[J]. The SIAM Review, 2005, 47(1): 135–161.

[2] Owens J, Luebke D, Govindaraju N, et al. A survey of general-purpose computation on graphics hardware[J]. Computer Graphics Forum, 2007, 26(1): 80–113.

[3] Kurzak J, Alvaro W, Dongarra J. Optimizing matrix multiplication for a short-vector SIMD architecture-CELL processor[J]. Parallel Computing, 2009, 35(3): 138–150.

[4] Kotakemori H, Hasegawa H, Kajiyama T, et al. Performance evaluation of parallel sparse matrix-vector products on SGI Altix3700[C]//Proceedings of the 1st International Workshop on OpenMP (IWOMP), Eugene, OR, USA, June 2005. Berlin, Heidelberg: Springer-Verlag, 2005: 153–163.

[5] Vazquez F, Garzon E M, Martinez J A, et al. The sparse matrix vector product on GPUs[R]. University of Almeria, 2009.

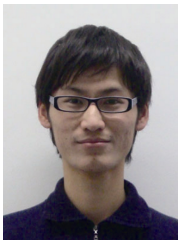
[6] Davis A T. The University of Florida sparse matrix collection[EB/OL]. (1994)[2011-04]. http://www.cise.ufl.edu/research/sparse/matrices/.

[7] Baskaran M, Bordawekar R. Optimizing sparse matrix-vector multiplication on GPUs RC24704[R]. IBM, 2008.

[8] Bell N, Garland M. Efficient sparse matrix-vector multiplication on CUDA NVR-2008-004[R]. NVIDIA, 2008.

WANG Wei was born in 1979. He received his Ph.D. degree from Tongji University. Now he is a lecturer at Department of Computer Science and Technology, Tongji University. His research interests include parallel distributed computing, trustworthy computing and cloud computing.

王伟(1979—), 男, 同济大学博士, 现为同济大学计算机科学与技术系讲师, 主要研究领域为并行分布式计算, 可信计算, 云计算。



CHEN Jianping was born in 1987. He is a master candidate at Tongji University. His research interests include parallel computing and cloud computing.

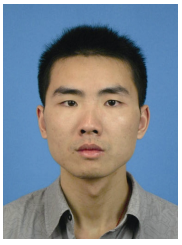
陈建平(1987—), 男, 上海人, 同济大学硕士研究生, 主要研究领域为并行计算, 云计算。

ZENG Guosun was born in 1964. He received his Ph.D. degree from Shanghai Jiaotong University. Now he is a professor and Ph.D. supervisor at Tongji University, and the senior member of IEEE and CCF. His research interests include heterogeneous parallel computing, software design and verification of trustworthy network and cloud computing.

曾国荪(1964—), 男, 上海交通大学博士, 现为同济大学教授、博士生导师, IEEE 和 CCF 高级会员, 上海市优秀学科带头人, “863 项目”信息领域专家评委成员, 主要研究领域为异构并行计算, 可信网络软件设计与验证, 云计算。

YU Lihua was born in 1986. She is a master candidate at Tongji University. Her research interests include parallel computing and green computing.

俞莉花(1986—), 女, 江苏海门人, 同济大学硕士研究生, 主要研究领域为并行计算, 绿色计算。



TAN Yiming was born in 1982. He is a Ph.D. candidate at Tongji University. His research interests include parallel computing and energy analysis.

谭一鸣(1982—), 男, 河南洛阳人, 同济大学博士研究生, 主要研究领域为并行计算, 能耗分析。