面向异构众核架构的块 Gauss-Seidel/Jacobi 预条件算法

吴立垒[1] 陈荣亮[2] 罗 力[2] 闫争争[2] 廖子菊[2] 迟利华[3] 刘 杰[1]

[1](国防科技大学并行与分布处理国家重点实验室 长沙 410073)

[2](中国科学院深圳先进技术研究院 广东 深圳 518055)

[3](湖南交通工程学院高科技研究院 湖南 湘阴 414600)

摘 要 Gauss-Seidel 算法作为线性方程组的求解器,在并行计算领域具有广泛应用,而面向异构众核架构开发其细粒度并行性一直是具有挑战性的问题.针对非结构网格问题,基于代数分块并行思路提出了面向异构众核架构的块 Gauss-Seidel/Jacobi 算法,将其作为区域分解算法的子区域求解器.面向神威太湖之光超级计算机的异构众核架构,设计并实现了该算法.为充分利用神威太湖之光国产 SW26010 芯片中每个 CPE 拥有的高速 LDM(Local Data Memory),缓解通信瓶颈,设计了多行块通信打包、计算与通信重叠性能优化策略和丢弃非关键元素的低通信复杂性数值优化方法.数值实验结果显示,相较于串行 Gauss-Seidel 算法,优化后的块 Gauss-Seidel/Jacobi 算法预处理过程加速比最高可达到 4.16 倍.以 1040 核的测试数据为基准,在处理器核数达到 33 280 时,块 Gauss-Seidel/Jacobi预条件算法的并行效率达到 61%.

关键词 非结构网格;异构众核架构;区域分解算法;块 Gauss-Seidel/Jacobi 算法;神威太湖之光

中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2019.02447

A Block Gauss-Seidel/Jacobi Preconditioner for Heterogeneous Many-Core Architecture

WU Li-Lei[1] CHEN Rong-Liang[2] LUO Li[2] YAN Zheng-Zheng[2]
LIAO Zi-Ju[2] CHI Li-Hua[3] LIU Jie[1]

[1](*Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha* 410073)

[2](*Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong* 518055)

[3](*Institute of Advanced Science and Technology, Hunan Institute of Traffic Engineering, Xiangyin, Hunan* 414600)

Abstract The Gauss-Seidel algorithm is widely used in the field of parallel computing as an iterative solver for linear system. However, it is challenging to exploit its fine-level parallelism on the emerging heterogeneous many-core architectures. For unstructured mesh problems, the missing of geometric information in matrices of unstructured mesh problems made the classical geometry-based parallel strategies fail. Instead, a block Gauss-Seidel/Jacobi algorithm for heterogeneous many-core architectures based on the algebra-based parallel strategy is proposed, which is used as the subdomain solver (preconditioner) in the DDM (Domain Decomposition Method). To balance

收稿日期:2017-12-01;在线出版日期:2018-05-15.本课题得到国家"九七三"重点基础研究发展规划项目基金(2017YFB0202104 和 2016YFB0200401)、国家自然科学基金(91530324,91430218,11401580,11701547,11602282 和 61531166003)、深圳市基础研究学科布局项目基金(JCYJ20170307165328836, JCYJ20160331193229720 和 JSGG20170824154458183)资助. **吴立垒**,硕士研究生,主要研究领域为区域分解算法和高性能计算. E-mail: wulilei15@nudt.edu.cn. **陈荣亮**,博士,副研究员,主要研究领域为偏微分方程、形状优化问题和并行计算. **罗 力**,博士,助理研究员,主要研究领域为计算流体力学、并行计算. **闫争争**,博士,助理研究员,主要研究领域为计算流体力学、并行计算. **廖子菊**,博士,助理研究员,主要研究领域为计算流体力学、并行计算. **迟利华**,博士,副教授,主要研究领域为并行算法、高性能计算. **刘 杰**,博士,研究员,主要研究领域为并行算法、高性能计算和机器学习.

between the parallel and numerical efficiencies, we combined the inherent high parallelism of Jacobi method and the good numerical converge rate of Gauss-Seidel method together to form the new algorithm. The block Gauss-Seidel/Jacobi algorithm can be regarded as small Gauss-Seidel iterations on the intra-thread level and a global Jacobi iteration on the inter-thread level. The proposed algorithm presents scalable parallelism with minimum inter-thread communication. Moreover, the unknowns were ordered in a node-by-node manner, meaning that all the N unknowns belongs to a mesh node are coupled to form a $N\times N$ block. Such ordering of unknowns does not affect the numerical performance but improves the convergence and scalability of our solver. For the problem solving the incompressible Navier-Stokes equations, the subdomain solver takes a 4×4 node block as a minimum unit. The Sunway TaihuLight supercomputer, consisted of 40 960 homegrown SW26010 many-core processors with totally over 10 million cores, is exactly based on heterogeneous many-core architecture. In this paper, the block Gauss-Seidel/Jacobi algorithm was implemented and optimized based on SW26010 many-core architecture. It is known that communication is the bottleneck of heterogeneous many-core architecture. Therefore, a series of low-communication-complexity optimization strategies were designed to achieve a better numerical performance. To reduce the communication cost, we designed implementation-based optimization strategies such as multiple lines packed copying and computation-communication overlapping techniques using the small but fast LDM (Local Data Memory) on many-core processors. Moreover, a copying reduced variant of the block Gauss-Seidel/Jacobi algorithm is also proposed to alleviate the memory bandwidth bottleneck. Unlike the optimization strategies based on accurate computation, the copying reduced version leads to inaccurate preconditioner by neglecting portion of data that need to be communicated and synchronized. Omitting the less important data does not affect the numerical efficiency apparently but achieves much better parallel efficiency. In this work, we adopt the RAS (Restricted Additive Schwarz) method for inter-node parallel algorithm and the block Gauss-Seidel/Jacobi method for the inter- and intra-thread parallelism. The aerodynamic simulations of a high-speed train model and a car model on unstructured mesh were tested on the Sunway TaihuLight. Numerical results show that the proposed block Gauss-Seidel/Jacobi algorithm delivers at most a 4.16x speedup comparing to the sequential version. For the parallel efficiency, our algorithm achieves a 61% parallel efficiency as the number of processors increases from 1040 to 33 280. Moreover, the proposed algorithms are not limited to be used as the subdomain solver in DDM, they can also be adopted as the iterative solver for linear systems and smoothers in the multigrid method.

Keywords unstructured mesh; heterogeneous many-core architecture; domain decomposition method; block Gauss-Seidel/Jacobi method; Sunway TaihuLight supercomputer

1 引 言

随着计算机计算能力的发展和流体模拟手段的成熟,人们对流体模拟的适应性和精度提出了越来越高的要求.非结构网格弥补了结构网格不能在任意形状、任意连通区域内进行网格剖分的缺陷,具有较强的几何适应性.与此同时,在复杂流体处理方面,非结构网格能对流体某些特定区域进行充分细的局部剖分得到精度较高的结果,因而在科学与工程计算中得到了越来越广泛的应用.然而,基于非结构网格的离散系统相对复杂,特别是离散系统对应的刚度矩阵的稠密度和带宽一般比结构网格对应的值大,往往导致较大的矩阵条件数,使得离散系统的求解更加困难[1],研究适用于非结构网格问题求解的高性能算法成为一种机遇和挑战.

Newton-Krylov-Schwarz(NKS)算法[2]是一套求解非线性方程组的高性能算法,主要包含 3 个部

分：求解非线性方程组的非精确 Newton 法、求解每个 Newton 迭代步中线性 Jacobian 系统的 Krylov 子空间迭代法和加速 Krylov 子空间迭代法收敛速度的 Schwarz 型预处理方法. NKS 算法中计算量最大、耗时最长的步骤是线性 Jacobian 系统的求解. 其中，Schwarz 型预处理方法由于其采用的“分而治之”思想，天然地适合构造面向大规模超级计算机的并行算法. 本文提出的预处理算法并行化方法是基于传统 Schwarz 算法改进后的限制型加性 Schwarz 方法[3]，其数学表达式如下：

$$M_{RAS}^{-1} = \sum_{i=1}^{np} (R_i^0)^{\mathrm{T}} M_i^{-1} R_i^{\delta} \tag{1}$$

式(1)中，M_i^{-1} 为子区域的预处理算子(即子区域求解器)，R_i^{δ} 和 $(R_i^0)^{\mathrm{T}}$ 分别为从全局区域到子区域的限制算子和延拓算子. 由于子区域求解的矩阵规模较小，且求解结果在 Krylov 迭代法中频繁使用，传统上常使用直接法(如 LU 分解法)求出其精确解. 但是，直接法的计算量通常较大，且 M_i^{-1} 作为预处理算子本身可以是非精确的，在实际应用中也常使用不完全 LU 分解法(ILU)或者迭代法(如 Gauss-Seidel 算法)作为子区域求解器. 在使用预处理算子后，求解 Jacobian 系统所需的 Krylov 算法迭代步数大大降低，相应地，子区域求解器就成为了求解整个非线性系统最为耗时的部分.

计算科学中算法的发展也离不开硬件进步的推动. 近些年来，半导体工艺、“功耗墙”等问题使得处理器主频的增长趋于极限，各种异构众核架构开始纷纷兴起，常见的有 CPU+GPU，CPU+MIC，CPU+FPGA 等. 而目前 Top500 排名第 1 位的神威太湖之光超级计算机系统[4]采用的就是异构众核架构，其计算芯片 SW26010 为每个结点内部提供了另一层并行性. 一方面，如何用好这一层的并行性，开发出对应的线程级并行算法，是充分发挥神威太湖之光超级计算机片上众核计算能力的关键. 另一方面，如前文所述，NKS 算法中的子区域预处理算子(也称子区域求解器)的构造也正是求解非线性系统的性能瓶颈. 这两方面因素使得对子区域求解器的并行化成为必然选择. 本文考虑将 NKS 算法中的子区域问题交由计算性能较强的加速卡进行并行计算. 但是，由于子区域预处理算子(通常使用 ILU、Gauss-Seidel 算法)的自身结构不易并行化，特别是对于非结构网格问题，传统针对结构网格采取的基于几何的并行方式较难实现[5]，使得这一工作面临许多挑战. 面向异构众核架构，提出并实现了基于代数的块 Gauss-Seidel/Jacobi 算法. 针对 SW26010 众核处理器，在并行方式和数值方法方面提出了多种性能优化策略，取得了良好的并行加速效果和算法可扩展性.

对于面向异构众核架构的预处理算法并行化工作，并行效率和数值效率是两个均需考虑而又互相矛盾的方面. 一方面，ILU 和 Gauss-Seidel 预处理算法本身结构不易并行，要从代数上获得较好的并行性就必然涉及到数值方法上的解耦，有可能会丢失一定的数值精度；另一方面，数值精度又直接影响预处理的效果，精度太低会使得 Krylov 算法迭代步数下降不明显，预处理失去意义. 目前，针对这一领域的研究大多是面向 CPU＋GPU 架构. Chow 和 Patel[6]针对传统基于 Gaussian 消去法的 ILU 预处理算法进行改进，提出了一种新型的基于不动点迭代法的 ILU 算法，使得元素级并行成为可能. 数值实验证明，只需要 2～3 次迭代就可以构造出极为有效的 ILU 预处理算子. Yang 等人[7]基于 Chow 提出的细粒度并行迭代 ILU 算法，面向申威 SW26010 众核处理器提出并实现了一种基于几何的并行流水线 ILU 算法(GP-ILU)作为区域分解后子区域的求解器. GP-ILU 子区域求解器实现了线程内和线程间两层流水线并行，对比传统的块 PILU[6]算法，只需要一次迭代就能取得良好的预处理效果. 数值实验的结果表明，在每个核组(CG)均使用 64 个从核(CPEs)计算时，GP-ILU 相较于串行 ILU 加速比最高达到 4.8 倍. Heuveline 等人[8-9]针对跨平台有限元软件包 HiFlow^3，利用多色排序和矩阵重排的思想提出了对 ILU(p)和块 Gauss-Seidel 型预处理算法的并行方法. 数值实验显示，其并行版本在多核 CPU 和 GPU 上都具有较好的并行效率和可扩展性. Cotronis 等人[10]面向 GPU 众核架构，利用红黑排序的思想实现了细粒度并行的局部松弛修正 SOR 算法(LMSOR)，避免了传统 SOR 算法在自适应选取 ω 的最佳值时存在的全局通信问题. Di 等人[11]基于 GPGPU 众核架构，提出了一种较传统的红黑排序 SOR 算法允许更多单指令多数据并行的 MLSOR 算法. 数值实验表明，在通用 GPU S1070 与 S2050 上，MLSOR 算法性能优于红黑排序 SOR(RBSOR)算法，且总体上，随着问题规模或 GPU 核数的增加，MLSOR 相较于 RBSOR 的性能优势也相应增加.

综上可知，一方面，针对 ILU 型或 Gauss-Seidel 型预处理算法的并行化工作大多面向同构架构，面

向异构架构的并行与优化工作不多，且大多都集中在 CPU＋GPU 异构众核架构. 面向神威太湖之光超级计算机系统，由于其在 2016 年 6 月才正式发布，目前能查阅到的文献仅有 Yang 等人[7]提出的基于几何的并行流水线 ILU 算法(GP-ILU). 另一方面，目前针对预处理算法的并行化工作大都基于结构化网格，基于非结构网格的研究工作较少.

针对这一研究现状，本文提出了面向异构众核架构的并行预处理算法. 该算法结合了基于分布式系统的粗粒度并行区域分解算法和基于异构众核处理器的细粒度并行 Gauss-Seidel 迭代算法. 其中本文着重讨论了作为区域分解算法中子区域求解器的块 Gauss-Seidel/Jacobi 算法. 对于许多应用，通信是国产申威异构众核芯片 SW26010 的性能瓶颈. 其设计者专门为每个 CPE(Computing Processing Element)加上了 64 KB 的用户编程可控 SPM 作为高速缓冲 LDM，以缓解通信瓶颈. 为充分利用 SW26010 芯片中的高速 LDM，设计了基于并行实现的多行块通信打包、计算与通信重叠性能优化策略和低通信复杂性数值优化方法等策略. 其中，多行块通信打包和计算与通信重叠并不改变算法的数值性能，而低通信复杂性块 Gauss-Seidel/Jacobi 算法会造成算法数值性能有较大的降低. 最终，使用高速列车气动模拟这一真实算例的测试结果显示，块 Gauss-Seidel/Jacobi 算法与串行 Gauss-Seidel 算法预处理效果相同，其预处理过程的加速比最高达到 4.16 倍. 以 1040 核的测试数据为基准，当处理器核数达到 33280 时，算法的并行效率为 61%.

本文将探讨面向异构众核架构的基本块 Gauss-Seidel/Jacobi 预处理算法及其优化版本. 文中的第 2 节，介绍节点间的区域分解算法和节点内的块 Gauss-Seidel/Jacobi 算法；第 3 节，阐述缓解通信瓶颈的主要思路，并据此提出低通信复杂性块 Gauss-Seidel/Jacobi 算法；第 4 节，给出使用高速列车外流场模拟这一真实算例的数值结果，并对其加以分析；最后，在第 5 节对前文加以总结，同时指出本文工作的广泛适用性.

2 基本的块 Gauss-Seidel/Jacobi 算法

预处理算子通过将线性方程组 $Mz=r$ 转换为另一个易于求解的同解方程，达到改善迭代法收敛性的目的. 本文采用的预处理算子包含了节点间进程级并行的区域分解算法和节点内线程级并行的块 Gauss-Seidel/Jacobi 子区域预处理算法，本文算法未考虑指令级并行优化.

假设有 np 个进程，使用基于图划分的策略将有限元网格划分为 np 个子区域(ParMETIS①)，其中，图上的每个顶点表示一个网格单元. 假设全局网格为 Ω_h，子区域网格为 $\Omega_{h,n}$(其中 n 为 1 至 np 的任意整数)，则有 $\Omega_h=\Omega_{h,1}\cup\cdots\cup\Omega_{h,np}$，其中，对于 $i\neq j$，总有 $\Omega_{h,i}\cap\Omega_{h,j}=\varnothing$ 成立.

在节点间并行方面，本文采用的区域分解算法是限制型加性 Schwarz 算法，其数学表达式如式(1)所示. 假设异构架构中每个 CPU 核对应 1 块加速卡，共有 np 个这样的组合，则算法将整体求解区域 Ω 划分为 np 个互不重叠的子区域，每个子区域向外拓展 δ 层，最终得到 np 个重叠的子区域. 如图 1 所示，Ω_i 对应进程 i，类似地，每个子区域均对应 1 个 MPI 进程. 其中，每个进程的 CPU 核负责子区域求解器的初始化和线程调度工作，加速卡则负责迭代地并行计算. 若异构架构中 CPU 核有盈余，CPU 也可参与并行计算. 每个子区域问题同时求解且互不影响，最终完成并行预处理过程.

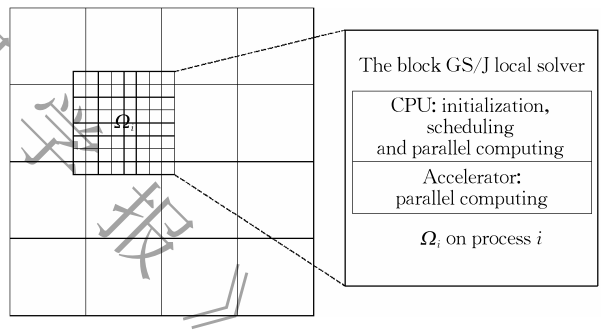

图 1 面向异构众核框架的区域分解算法示意图(其中 Ω_i 为第 i 个子区域，由一个众核处理器核组求解)

在节点内并行方面，常用的子区域求解器有 ILU 型和 Gauss-Seidel 型预处理算法，本文重点研究求解非结构网格中子区域问题的 Gauss-Seidel 算法. 在非结构网格问题中，矩阵行的相邻不代表几何上网格点的相邻，并不能单独从矩阵中获得足够的物理、几何信息，这使得传统基于几何的并行方式失效. 所以，针对 Gauss-Seidel 算法的并行思路要从基于代数的角度入手.

2.1 SOR、Gauss-Seidel 及 Jacobi 算法

SOR 算法的全称为逐次超松弛迭代法(Successive Over Relaxation method)[12]，其最初是用于求

① ParMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library, http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview, 2013.3.30

解线性方程组 $Mz=r$，而这里主要研究求解 Schwarz 算法划分后的子区域问题对应的 SOR 算法. 基本的 SOR 算法分量形式可以表示为：

$$z_i^{(k+1)}=\frac{\omega}{m_{ii}}\left(-\sum_{j=1}^{i-1}m_{ij}z_j^{(k+1)}-\sum_{j=i+1}^{n}m_{ij}z_j^{(k)}+r_i\right)+(1-\omega)z_i^{(k)},\ (i=1,2,\cdots,n;\ k=0,1,2,\cdots)$$

其中，i 表示当前计算的第 i 个分量(这里假设一共有 n 个分量)，k 表示当前迭代的步数，$z_i^{(k+1)}$ 表示第 $k+1$ 次迭代得到的更新向量 z 第 i 个分量，m_{ij} 表示矩阵 $\boldsymbol{M}$ 的第 i 行，第 j 列元素，r_i 表示线性方程组中向量 $\boldsymbol{r}$ 的第 i 个分量，ω 表示权值，若 ω 的数值选取得当，可使收敛速度有明显改善. SOR 算法实际上是对 Gauss-Seidel 算法的一种修正，在其每个迭代步中，使用 Gauss-Seidel 算法计算出的结果与上一步迭代步的结果进行加权平均，从而得到最终结果. 当 $\omega=1$ 时，SOR 迭代法退化成 Gauss-Seidel 迭代法. 相较于 SOR 算法，Gauss-Seidel 算法的求解精度可能会有所降低，但由于预处理算子可以是非精确的，本文考虑将 Gauss-Seidel 算法作为预处理算子. 由于 Gauss-Seidel 算法在计算 $z^{(k+1)}$ 的第 i 个分量时，需要利用已经计算出的最新分量 $z_j^{(k+1)}$ $(j=1,2,\cdots,i-1)$，使得各个分量的计算任务之间具有依赖性，算法难以并行化. Jacobi 算法与 Gauss-Seidel 算法同属于基于矩阵分裂的迭代法，在 Jacobi 算法中，计算 $z_i^{(k+1)}$ 时不使用变量的最新信息，这使得算法收敛速度有所降低，但却使得各个分量的计算任务可以并行执行. 因此考虑将 Gauss-Seidel 和 Jacobi 算法的优点结合起来，一方面易于并行，另一方面收敛速度较快，提出基于代数的块 Gauss-Seidel/Jacobi 算法.

2.2　非结构网格的矩阵模式

非结构网格问题对应的系数矩阵 $\boldsymbol{M}$ 的非零元结构通常较为不规则. 由于非结构网格问题的矩阵非零元分布模式大都相似，这里以高速列车外流场模拟这一真实算例为例，其非结构网格对应的全局矩阵和使用区域分解算法划分得到的子区域矩阵非零元分布模式如图 2、3 所示. 其中，图 3 的 4×4 小方块对应图 2 的每个小黑点，表示在每个网格节点中包含 3 个速度分量和 1 个压力分量，可以看出，不管是全局矩阵还是子区域矩阵，其非零元分布均较为稀疏且不规则.

从这样的矩阵结构中难以得到足够的几何信息，这使得传统基于几何的并行方式失效，因而本文考虑采用基于代数的并行方式. 如上文所述，对 Gauss-Seidel 预处理算法采取的并行策略是利用 Jacobi 算法易于并行的特点，结合 Gauss-Seidel 算法收敛速度较快的优势，在线程间使用块状 Jacobi 算法，而线程内部则使用 Gauss-Seidel 算法，最终构成块 Gauss-Seidel/Jacobi 算法作为并行子区域求解器. 由于这里采用的是全耦合算法，其以节点-节点(node-by-node)方式对未知量进行排序，即将一个网格节点上的所有未知量排列在一起，形成一个紧凑的小块，便于使用块状的 Gauss-Seidel 算法对其进行求解. 为了利用好物理信息来加速预处理效果，本文在对子区域求解器并行时，都是以节点块作为最小单位.

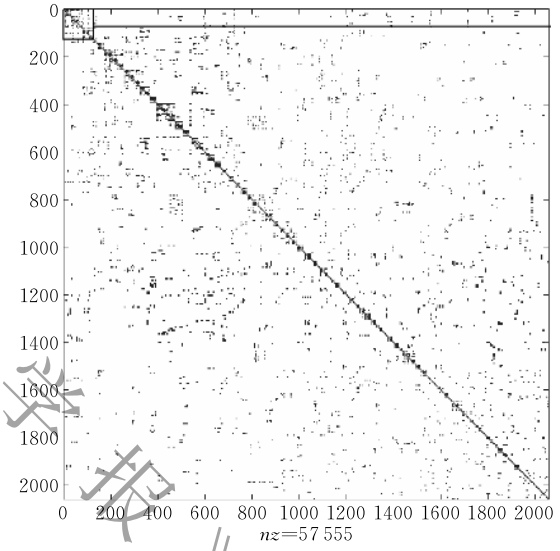

图 2　全局矩阵非零元分布

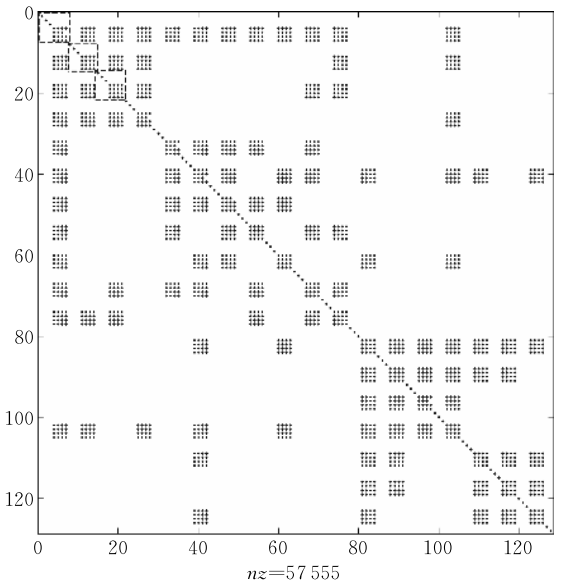

图 3　子区域矩阵非零元分布

2.3 基本的块 Gauss-Seidel/Jacobi 算法

由于传统串行 Gauss-Seidel 算法的某一未知分量计算需利用当前迭代步更新的所有分量，计算任务间具有严格的依赖性，其与异构众核架构的高并行度存在匹配性问题. 而预处理算子在实际应用中可为非精确的，考虑放宽一定数值精度以换取算法的并行性，即将 Gauss-Seidel 算法与 Jacobi 算法结合，以充分利用异构众核架构的高并行性. 通过降低每次迭代的开销时间，达到最终整体提高计算效率的目的. 该并行化思路采用基于代数思想，不需要网格的几何信息，能够有效用于复杂的非结构网格问题.

面向异构众核架构，提出的基本块 Gauss-Seidel/Jacobi预处理算法框架如算法 1 所示.

算法 1. 基本的块 Gauss-Seidel/Jacobi 算法.

输入：非零元对角块 $\boldsymbol{D}_{ii}$、向量 $\boldsymbol{r}_i$、上三角矩阵 $\boldsymbol{L}_{ij}^{out}$、$\boldsymbol{L}_{ij}^{in}$、下三角矩阵 $\boldsymbol{U}_{ij}^{out}$、$\boldsymbol{U}_{ij}^{in}$

输出：第 k 次迭代得到的解 $\boldsymbol{z}_i^{k+1}$

//步骤 1. 使用 n 个线程并行求解对角块矩阵的逆

1. For $i=start,\cdots,end$
2. 拷贝 $\boldsymbol{D}_{ii}$ 至异构芯片存储器
3. 计算 $\boldsymbol{D}_{ii}^{-1}$
4. 拷贝 $\boldsymbol{D}_{ii}^{-1}$ 至主存
5. End for

//步骤 2. 使用 n 个线程并行计算初始值

6. For $i=start,\cdots,end$
7. 拷贝 $\boldsymbol{D}_{ii}$ 和 $\boldsymbol{r}_i$ 至异构芯片存储器
8. $\boldsymbol{z}_i^0=\boldsymbol{D}_{ii}^{-1}\boldsymbol{r}_i$
9. 拷贝 $\boldsymbol{z}_i^0$ 至主存
10. End for

//步骤 3. 使用 n 个线程并行迭代更新值

11. For $k=0,\cdots,K$
12. For $i=start,\cdots,end$
13. 拷贝 $\boldsymbol{D}_{ii}^{-1}$，$\boldsymbol{r}_i$，$\boldsymbol{L}_{ij}^{out}$，$\boldsymbol{L}_{ij}^{in}$，$\boldsymbol{D}_{ii}$，$\boldsymbol{U}_{ij}^{in}$，$\boldsymbol{U}_{ij}^{out}$ 至异构芯片存储器
14. $$\boldsymbol{z}_i^{k+1}=\boldsymbol{z}_i^k+\boldsymbol{D}_{ii}^{-1}\Big(\boldsymbol{r}_i-\sum_{j=1}^{start-1}\boldsymbol{L}_{ij}^{out}\boldsymbol{z}_j^k-\sum_{j=start}^{i-1}\boldsymbol{L}_{ij}^{in}\boldsymbol{z}_j^{k+1}-\boldsymbol{D}_{ii}\boldsymbol{z}_j^k-\sum_{j=i+1}^{end}\boldsymbol{U}_{ij}^{in}\boldsymbol{z}_j^k-\sum_{j=end+1}^{m}\boldsymbol{U}_{ij}^{out}\boldsymbol{z}_j^k\Big)$$
15. 拷贝 $\boldsymbol{z}_i^{k+1}$ 至主存
16. End for
17. End for

其中，基本的块 Gauss-Seidel/Jacobi 算法主要包含两个部分：计算初值和迭代更新解. 假设在构造子区域的预处理算子时，待求解的线性方程组为 $\boldsymbol{Mr}=\boldsymbol{z}$，其中矩阵 $\boldsymbol{M}$ 共有 m 个行块，线程数为 n，则分配到每个线程上的行块数为 m/n. 算法 1 中使用 $start$ 和 end 分别表示分配到当前线程的起始行块号和结束行块号，$\boldsymbol{L}^{out}$，$\boldsymbol{L}^{in}$ 表示 $\boldsymbol{M}$ 的下三角块矩阵，$\boldsymbol{D}$ 表示 $\boldsymbol{M}$ 的对角块矩阵，$\boldsymbol{U}^{out}$，$\boldsymbol{U}^{in}$ 表示 $\boldsymbol{M}$ 的上三角块矩阵，out 表示当前矩阵行上非零元块对应的向量元素位于其它线程，in 则表示当前非零元块对应的向量元素位于当前线程.

在传统的串行算法中，一般取初值 $\boldsymbol{z}_i^0=\boldsymbol{D}_{ii}^{-1}(\boldsymbol{r}_i-\boldsymbol{L}_{ij}\boldsymbol{z}_j)$. 但因为式子中存在 $\boldsymbol{z}_j$，这样的初值计算过程对整个 $\boldsymbol{z}$ 向量有依赖，无法并行计算. 所以在算法设计时直接将包含 $\boldsymbol{z}_j$ 的项略去，取 $\boldsymbol{z}_i^0=\boldsymbol{D}_{ii}^{-1}\boldsymbol{r}_i$. 迭代更新解的过程对应算法 1 的步骤 3，其中，每个线程内部都可看作一个小的 Gauss-Seidel 迭代过程，即迭代求解 $\boldsymbol{z}_i^{k+1}=\boldsymbol{z}_i^k+\boldsymbol{D}_{ii}^{-1}(\boldsymbol{r}_i-\boldsymbol{M}_{ij}\boldsymbol{z}_j)$；而各个线程之间可看作是一次大的 Jacobi 迭代过程. 算法 1 运行到第 14 行时，每个线程的数据分配情况如图 4 所示. 首先将 M，$\boldsymbol{z}$ 和 $\boldsymbol{r}$ 从行块号 $start$ 到 end（p 号线程对应的）部分分配给 p 号线程. 除此之外，为了完成迭代更新，还需把向量 $\boldsymbol{z}$ 的其它部分分配给 p 号线程. 由于在第 k 步的迭代中，每个线程均对向量 $\boldsymbol{z}$ 的某一部分进行更新，为了避免同步开销，对于不在 p 号线程上的 $\boldsymbol{z}$ 向量数据，都只使用第 k 步迭代得到的，而在 p 号线程上的 $\boldsymbol{z}$ 向量数据则使用已更新的结果.

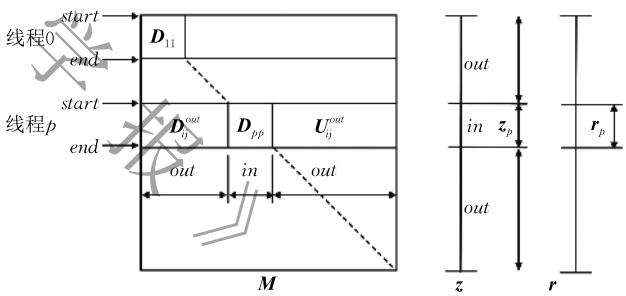

图 4 每个线程的数据分配情况

2.4 块 Gauss-Seidel/Jacobi 迭代算法收敛性证明

下面通过数学推导证明当线性方程组的系数矩阵 $\boldsymbol{M}$ 为严格对角占优或不可约的弱对角占优时，使用块 Gauss-Seidel/Jacobi 算法求解线性方程组 $\boldsymbol{Mr}=\boldsymbol{z}$ 将会收敛. 首先给出几个收敛性证明的相关定理.

定理 1[13]（一阶定常迭代法收敛性判别定理）. 对任意初始向量 $\boldsymbol{x}^{(0)}$ 和向量 $\boldsymbol{g}$，由一阶定常迭代法：

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{B}\boldsymbol{x}^{(k)}+\boldsymbol{g},\ k=0,1,2,\cdots \tag{2}$$

产生的向量序列 $\{\boldsymbol{x}^{(k)}\}$ 收敛的充分必要条件为 $\rho(\boldsymbol{M})<1$，其中 ρ 表示矩阵的谱半径.

定理 2[13]. 当矩阵 $\boldsymbol{M}=(a_{ij})_{n\times n}\in\mathbb{R}^{n\times n}(\mathbb{C}^{n\times n})$ 为严格对角占优或为不可约的弱对角占优矩阵时，则 $a_{ii}\neq0(i=1,2,\cdots,n)$，且 $\boldsymbol{M}$ 为非奇异矩阵.

由定理 2 可知，子区域对应的线性方程组 $\boldsymbol{Mr}=\boldsymbol{z}$ 的可使用块 Jacobi 和块 Gauss-Seidel 算法，相应地，也可使用块 Gauss-Seidel/Jacobi 算法.

当矩阵 $\boldsymbol{M}$ 为不可约的弱对角占优矩阵时，由定理 1 可知，只需证明 $\rho(\boldsymbol{B}_{\mathrm{GSJ}})<1$（$\boldsymbol{B}_{\mathrm{GSJ}}$ 为块 Gauss-Seidel/Jacobi 算法的迭代矩阵），其中 $\boldsymbol{B}_{\mathrm{GSJ}}=(\boldsymbol{D}-\boldsymbol{L}_{\mathrm{GSJ}})^{-1}\boldsymbol{U}$，则可知块 Gauss-Seidel/Jacobi 算法对于线性方程组 $\boldsymbol{Mr}=\boldsymbol{z}$ 收敛.

采用反证法证明其收敛性. 假设块 Gauss-Seidel/Jacobi 算法不收敛，由定理 1 可知 $\rho(\boldsymbol{B}_{\mathrm{GSJ}})\geqslant 1$，即存在 $\boldsymbol{B}_{\mathrm{GSJ}}$ 的特征值 λ 使得式子 $|\lambda|\geqslant 1$ 成立，则 $\det(\lambda\boldsymbol{I}-(\boldsymbol{D}-\boldsymbol{L}_{\mathrm{GSJ}})^{-1}\boldsymbol{U}_{\mathrm{GSJ}})=0$，即 $\det(\boldsymbol{D}-\boldsymbol{L}_{\mathrm{GSJ}}-\lambda^{-1}\boldsymbol{U}_{\mathrm{GSJ}})=0$.

另一方面，由于不可约的弱对角占优矩阵 $\boldsymbol{M}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$ 与矩阵 $\boldsymbol{D}-\boldsymbol{L}_{\mathrm{GSJ}}-\lambda^{-1}\boldsymbol{U}_{\mathrm{GSJ}}$ 的零元和非零元位置相同（块 Gauss-Seidel/Jacobi 算法只缩放了矩阵 $\boldsymbol{M}$ 的非零元，不改变非零元的位置），且 $|\lambda|\geqslant 1$，即 $|1/\lambda|\leqslant 1$，则 $\boldsymbol{D}-\boldsymbol{L}_{\mathrm{GSJ}}-\lambda^{-1}\boldsymbol{U}_{\mathrm{GSJ}}$ 也为不可约的弱对角占优矩阵. 因此，由定理 2 可知，矩阵 $\boldsymbol{D}-\boldsymbol{L}_{\mathrm{GSJ}}-\lambda^{-1}\boldsymbol{U}_{\mathrm{GSJ}}$ 也为非奇异矩阵，即 $\det(\boldsymbol{D}-\boldsymbol{L}_{\mathrm{GSJ}}-\lambda^{-1}\boldsymbol{U}_{\mathrm{GSJ}})\neq 0$. 这与上式 $\det(\boldsymbol{D}-\boldsymbol{L}_{\mathrm{GSJ}}-\lambda^{-1}\boldsymbol{U}_{\mathrm{GSJ}})=0$ 相矛盾，故假设不成立，所以块 Gauss-Seidel/Jacobi 算法收敛.

类似地，当矩阵 $\boldsymbol{M}$ 为严格对角占优矩阵时，也可采用同样方法证明块 Gauss-Seidel/Jacobi 算法收敛. 下文的数值实验从实验角度同样证明了块 Gauss-Seidel/Jacobi 算法的收敛性.

3　低通信复杂性块 Gauss-Seidel/Jacobi 算法

针对异构众核架构存在的通信瓶颈问题，本节设计了多种针对异构芯片存储器（例如编程可控便笺式存储器 SPM）的优化策略，且在数值实验中取得了较好的效果.

3.1　多行块通信打包

基本的块 Gauss-Seidel/Jacobi 算法每个线程在计算某一行块时，需要将这一行的数据传到每个加速器核，其实质是传输 1 行数据就计算 1 行. 这样的方式使得计算每个行块都需要重新调用一次主存和异构芯片存储器间的通信操作，而每次唤起和结束设备通信操作均需耗费大量时间，所以设计在调用通信操作时考虑将多行块通信打包，使多次通信合并为一次，达到缩短通信时间的目的. 由于块 Gauss-Seidel/Jacobi 算法计算初值的过程中涉及的数据传输较少，且计算相对简单，在这里只讨论迭代更新解的过程. 其数据传输过程如图 5 所示，其中，带有颜色的方块表示每个非零元块. 假设此时 1 号线程即将计算第 5 个行块（深色条纹），调用通信操作一次传输 3 个行块的非零元数据到异构芯片存储器中，则图中深色（包括深色和深色条纹）的方块表示需要传输的数据，对于矩阵 $\boldsymbol{M}$ 和向量 $\boldsymbol{r}$，传输第 5 到 7 号行块的数据；对于向量 $\boldsymbol{z}$，则传输矩阵第 5 到 7 号行块非零元对应的数据.

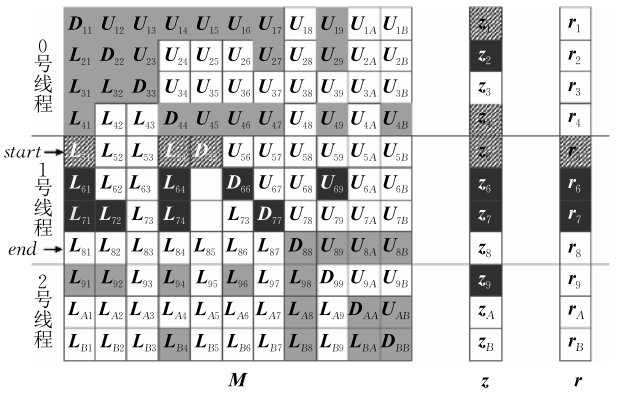

图 5　多行块通信打包版本示意图

3.2　计算与通信重叠

进一步地，为了缓解通信瓶颈，本文采用双缓冲结构进行计算与通信重叠. 同样地，在这里只讨论迭代更新解的过程. 双缓冲结构的原理如图 6 所示，首先，在异构芯片存储器中开辟两块相同大小的存储空间作为双缓冲，然后将数据传输到其中某一个缓冲，接下来使用这个缓冲上的数据进行计算，与此同时，启动通信操作将下一部分数据传输到另一个缓冲中，然后每次两个缓冲交替进行计算、通信操作，最终达到将计算与通信重叠的目的. 图 6 所示的情况是在一次传 1 个行块的数据基础上进行计算和通信的重叠，其中，通信和计算操作按照颜色由浅至深的顺序执行，在竖直方向上的对齐则表示相互重叠. 计算与通信重叠也可以一次传多行结合起来，但由于需要在异构芯片存储器中开辟双缓冲，每次传输的行块数量也受到了一定的限制.

图 6　计算与通信重叠示意图

3.3　丢弃非关键元素的低通信复杂性数值优化

以上设计的两种优化方法都是基于并行实现

的,除此之外,基于数值算法的优化方法也能有效地缓解通信瓶颈.如前文中提到,预处理算子可以是非精确的,可利用这一性质,在不过多牺牲预处理效果的前提下,有效降低通信开销,达到提高算法并行效率的目的.由于块 Gauss-Seidel/Jacobi 算法本质上是迭代法,所有行块的数据在每步迭代中都要传输至异构芯片存储器,为了减少数据传输量,将迭代的思想也融入到通信中,迭代地对不同行块数据传输.除此之外,通过观察可以发现,非结构网格产生的矩阵大多非零元素都位于对角线附近,为了减少传输到异构芯片存储器上的数据,考虑将算法 1 的第 14 行中带有 *out* 项忽略,只传输位于当前线程的向量 $\boldsymbol{r}$ 数据和其对应的矩阵非零元块.基于以上两点想法,提出了低通信复杂性块 Gauss-Seidel/Jacobi 算法,其框架如算法 2 所示.

算法 2. 丢弃非关键元素的低通信复杂性块 Gauss-Seidel/Jacobi 算法.

输入:非零元对角块 $\boldsymbol{D}_{ii}$、向量 $\boldsymbol{r}_i$、上三角矩阵 $\boldsymbol{L}_{ij}^{in}$、下三角矩阵 $\boldsymbol{U}_{ij}^{in}$

输出:第 k 次迭代得到的解 $\boldsymbol{z}_i^{k+1}$

//步骤 1. 使用 n 个线程并行求解对角块矩阵的逆

1. For $i=start,\cdots,end$
2. 拷贝 $\boldsymbol{D}_{ii}$ 至异构芯片存储器
3. 计算 $\boldsymbol{D}_{ii}^{-1}$
4. 拷贝 $\boldsymbol{D}_{ii}^{-1}$ 至主存
5. End for

//步骤 2. 使用 n 个线程并行计算初始值

6. For $i=start,\cdots,end$
7. 拷贝 $\boldsymbol{D}_{ii}$ 和 $\boldsymbol{r}_i$ 至异构芯片存储器
8. $\boldsymbol{z}_i^0=\boldsymbol{D}_{ii}^{-1}\boldsymbol{r}_i$
9. 拷贝 $\boldsymbol{z}_i^0$ 至主存
10. End for

//步骤 3. 使用 n 个线程并行迭代更新值

11. For $k=0,\cdots,k$
12. For $i=start,\cdots,end$
13. 迭代地拷贝 N 个行块 $\boldsymbol{D}_{ii}^{-1},\boldsymbol{r}_i,\boldsymbol{L}_{ij}^{in},\boldsymbol{D}_{ii},\boldsymbol{U}_{ij}^{in}$ 至异构芯片存储器
14. $$\boldsymbol{z}_i^{k+1}=\boldsymbol{z}_i^k+\boldsymbol{D}_{ii}^{-1}\left(\boldsymbol{r}_i-\boldsymbol{D}_{ii}\boldsymbol{z}_j^k-\sum_{j=start}^{i-1}\boldsymbol{L}_{ij}^{in}\boldsymbol{z}_j^{k+1}-\sum_{j=i+1}^{end}\boldsymbol{U}_{ij}^{in}\boldsymbol{z}_j^k\right)$$
15. 拷贝 $\boldsymbol{z}_i^{k+1}$ 至主存
16. End for
17. End for

上述算法相较与算法 1 的主要改进在步骤 3,算法 13 行"对 N 个行块迭代拷贝"表示在每次迭代过程中,迭代地选取当前 N 个行块中的其中 1 个进行计算更新,其余行块则跳过不计算.在具体实现中表现为,在每次调用通信操作前,进行 $(i+k)\%Nreuse$ 是否为零的判断,其中 *Nreuse* 表示待选取的行块数目 N.此外,第 14 行将基本算法中带有 *out* 的项忽略,有效地减少了通信传输量.

4 数值实验

神威太湖之光超级计算机是目前 Top500 排名第 1 位的超级计算机,也是世界上首台峰值性能超过 100 PFlops 的超级计算机.而为太湖之光提供强大计算性能的是国产申威 SW26010 异构众核芯片,其基本结构如图 7 所示.其中,整个芯片由 4 个核心组(CGs)构成.核心组之间通过片上网络(NoC)互连,每个核心组由 1 个管理核心(MPE)、64 个 8×8 网格分布的计算核心(CPEs)和 1 个存储控制器(MC)组成.管理核心主要负责管理、任务调度和数据通信;计算核心主要负责计算.其中,值得一提的是,为缓解 SW26010 芯片存在的通信瓶颈,芯片设计人员特别为芯片上的每个计算核心(CPE)加上了 64 KB 的编程可控便笺式存储器 SPM(Scratch Pad Memory).SPM 可作为编程可控的高速缓冲 LDM(Local Data Memory),用户通过使用 DMA(Direct Memory Access)操作命令,可以方便地实现 LDM 与主存之间的数据传输.本文面向神威太湖之光超级计算机的异构众核架构,实现了块 Gauss-Seidel/Jacobi 预处理算法及其低通信复杂性优化版本.如图 1 中描述,MPE 的任务类似于 CPU,主要负责初始化和调度操作;CPE 的任务类似于加速器,主要负责并行计算.每个 MPE 与 64 个 CPE 对应,共同负责计算 1 个子区域问题,实现节点内并行计算.

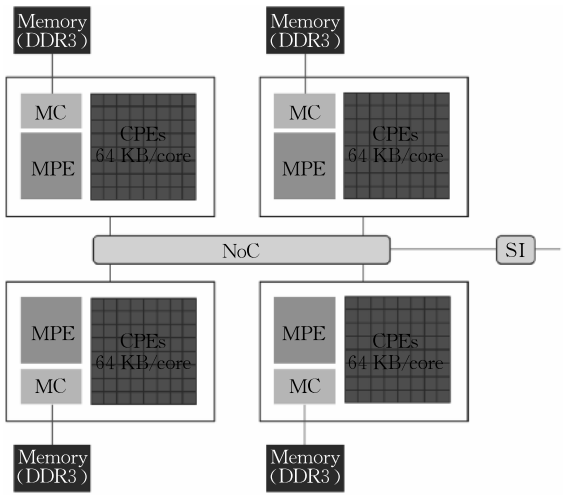

图 7 SW26010 处理器的基本结构

其中，块 Gauss-Seidel/Jacobi 预处理算法的实现基于高性能可移植可拓展工具箱 PETSc(Portable Extensible Toolkit for Scientific computation)[14]. 为了更加方便地对每个线程进行控制和调度，充发掘 SW26010 异构众核处理器线程并行性，在实现时本次实验调用了申威 26010 加速线程库(Athread 库). Athread库是针对主从加速编程模型所设计的轻量级线程库，能够充分发挥多从核的加速性能. 在算法收敛条件方面，设定 Krylov 算法的迭代终止条件为相对误差小于 10^{-4}；对于块 Gauss-Seidel/Jacobi 算法，设置其最大迭代步数为 3.

4.1　应用算例

为了测试上文提出的并行 Gauss-Seidel 预条件算法实际性能，实验选取高速列车外流场模拟[15-16]和汽车外流场模拟[17]这两个真实算例.

其中，高速列车外流场模拟算例(以下称算例 1)基于和谐号 CRH380B 的三维高速列车模型，模拟当两节车厢的列车行驶速度为 10 km/h，侧风速度为 72 km/h(沿 x 轴负方向)时列车周围流场的具体情况. 汽车外流场模拟算例(以下称算例 2)基于奥迪 RS5 车型的全尺寸真实模型，其车长 $L=4.637$ m，车宽 $W=2.026$ m，车高(最大高度)$H=1.411$ m，轴距 $B=1.595$ m. 为了尽量避免计算区域壁面的影响，将计算区域设定为 $7L\times7W\times4H$ 大小，汽车车身前部与入流边界的距离设定为 $2L$. 算例模拟汽车行驶速度为 10 km/h，侧风速度为 72 km/h(沿 x 轴负方向)时汽车周围流场的情况.

以上两个算例均采用非结构化网格离散，以获得局部复杂流场的精确结构. 为了方便处理，假定列车和汽车的周围流场均为 25°的标准空气(流体黏性系数 $\mu=1.831\times10^{-5}$ kg/ms，密度 $\rho=1.185$ kg/m^3)，并将初始条件设定为零，时间步长为 $\Delta t=0.01$. 空间离散方面，其使用加稳定化项的 P1-P1 有限元方法；而在时间离散方面，则采用全隐的后向 Euler 有限差分时间离散格式. 由于高速列车和汽车的运行速度一般低于 0.3 Mach，可将其外流场视作不可压缩 Newton 流体，并采用不可压缩 Navier-Stokes 方程对其进行描述. 针对控制方程离散后产生的非线性方程组，该算例使用全耦合的 Newton-Krylov-Schwarz 算法进行求解. 其中，基于区域分解思想的 Schwarz 预处理算法是 NKS 算法中的重要部分，其将线性方程组对应的全矩阵划分为子区域后求解，这里选择限制型加性 Schwarz 算法[3]划分子区域，子区域求解器则选取本文构造的块 Gauss-Seidel/Jacobi 算法.

4.2　预处理子的数值收敛性比较

首先对串行和并行预条件算法的数值收敛性进行比较分析. 通常数值收敛性能用 Krylov 算法的迭代收敛步数衡量，迭代收敛步数越少，表示预处理效果越好；反之，预处理效果则越差. 图 8、9 分别为算例 1、2 在神威太湖之光超级计算机上测试时，使用不同预处理算法后 Krylov 子空间算法的迭代收敛步数. 其中，serial-ILU、serial-GS、copy N lines、compu.-commu. 分别表示使用串行 ILU、串行 Gauss-Seidel、多行块通信打包版本块 Gauss-Seidel/Jacobi、计算与通信重叠版本块 Gauss-Seidel/Jacobi 作为子区域求解器. 同样地，reduce(N)表示使用低通信复杂性块 Gauss-Seidel/Jacobi 算法作为子区域求解器，这里的 N 表示迭代地选取当前 N 个行块中的其中 1 个进行计算更新. 由于低通信复杂性块 Gauss-Seidel/Jacobi 预条件的精度损失较多，使得算例 2 在 1000 步 Krylov 迭代步中仍无法收敛，因此下文中的算例 2 均不针对低通信复杂性块 Gauss-

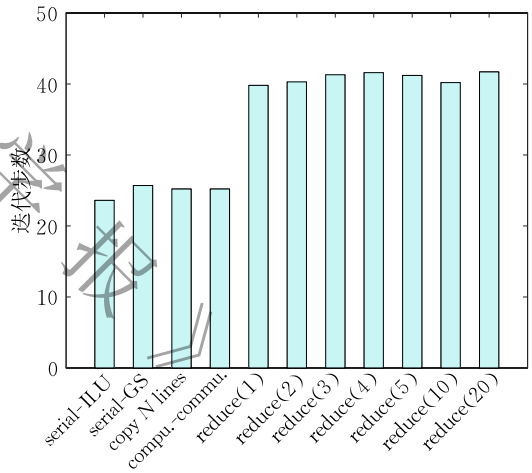

图 8　算例 1 的 Krylov 迭代收敛步数对比

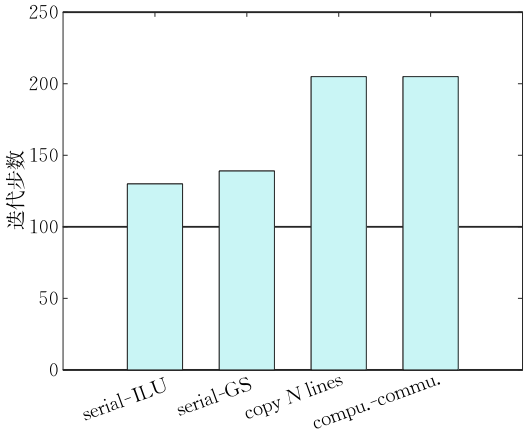

图 9　算例 2 的 Krylov 迭代收敛步数对比

Seidel/Jacobi 算法测试. 测试共使用了 32 个核组(CG),即 32 个进程,其中每个主核(MPE)均与 64 个从核(CPE)协同计算,Krylov 迭代收敛步数均为 10 个时间步的平均值. 从图 8 可以看出,针对算例 1,本文提出的多行块通信打包和计算与通信重叠的并行 Gauss-Seidel 算法与串行 Gauss-Seidel算法的预处理效果基本相同,而低通信复杂性块 Gauss-Seidel/Jacobi 算法的预处理效果则下降较多,仅达到原本串行算法效果的约 60%. 从图 9 可看出,针对算例 2,本文提出的多行块通信打包和计算与通信重叠的并行 Gauss-Seidel 算法较串行 Gauss-Seidel算法的预处理效果下降约 30%. 从这一结果可知,基于并行实现的优化对预处理效果在 Krylov 迭代步数较少时并不产生太大影响,而在步数较多时预处理效果则有一定下降. 此外基于数值算法的优化由于数据丢失过多,预处理效果下降较为严重,甚至会出现难以收敛的情况.

4.3 加速性能分析

在并行算法的加速性能方面,分别对上文的 3 个优化版本进行了实际测试(测试参数设置与上文相同). 图 10、11、12、13 为算例 1 和算例 2 使用基于并行优化的预处理算子的预处理过程耗时和加速比. 其中,纵轴表示预处理过程总耗时,SERIAL 表示只使用 MPE 的串行版本 Gauss-Seidel 预处理算法,横轴表示一次传输打包行块数目. 由于 LDM 的容量限制,每次最多只能传输 5 个行块到 LDM,而计算与通信重叠版本由于使用了双缓冲结构,一次最多只能传输 2 个行块(实测中算例 2 的计算与通信重叠版本最多只能传输 1 个行块). 从图中可以看到,随着传输行块数目的增加,算法加速性能也不断提高,最终,两个优化版本均取得了较好的加速效果,且在一次传输相同数量的行块时,计算与通信重叠版本加速效果优于多行块通信打包版本. 然而在最大加速性能方面,多行块通信打包版本还是略胜一筹,算例 1 的加速比最高达到了 4.16 倍,而算例 2 则最高达到了 2.89 倍. 其中,算例 2 的最高加速比低于算例 1,这是由于算例 2 中基于并行优化的预处理算法效果有一定损失,使得迭代收敛步数增加,加速效果受限. 考虑到 SW26010 的每个核组有 64 个从核,这一加速比并不高. 其原因主要包含两个方面:一方面,从核的计算能力只约为主核的一半,且并行化后主核并未参与计算;另一方面,每个从核的访存操作未对齐,且大量初始化操作并未并行化. 图 14、15 为算例 1 使用低通信复杂性块 Gauss-Seidel/Jacobi 算法时预处理过程的耗时和加速比(算例 2 在 1000 步 Krylov 迭代步内不收敛). 其中,横坐标表示迭代选取行块数目,当其从 15～20 时,加速比略有下降,这是因为行号反映某自由度在单元中的编号,行号相差大的两行矩阵元素可能来自不同单元的离散. 丢弃非关键元素优化中采用的直接替代方式虽然可减少数据传输的开销,却

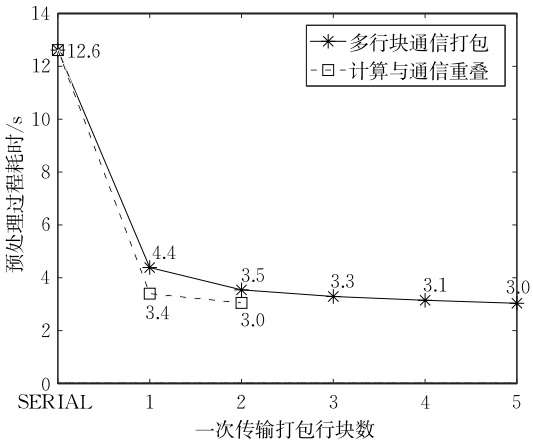

图 10 算例 1 基于并行优化的预处理算法耗时

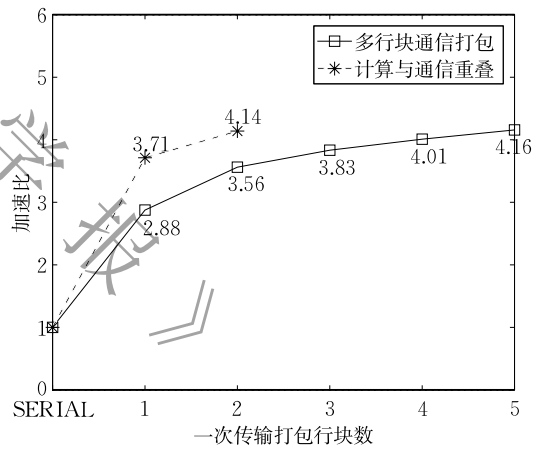

图 11 算例 1 基于并行优化的预处理算法加速比

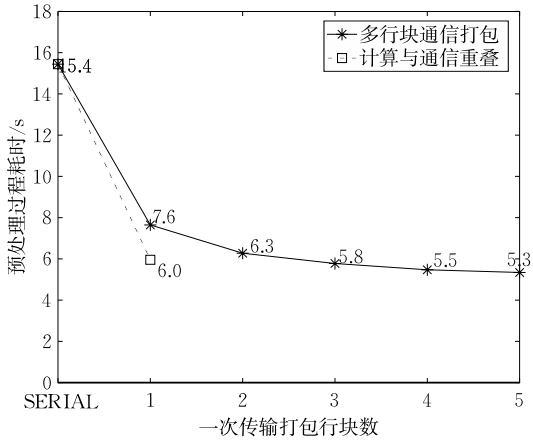

图 12 算例 2 基于并行优化的预处理算法耗时

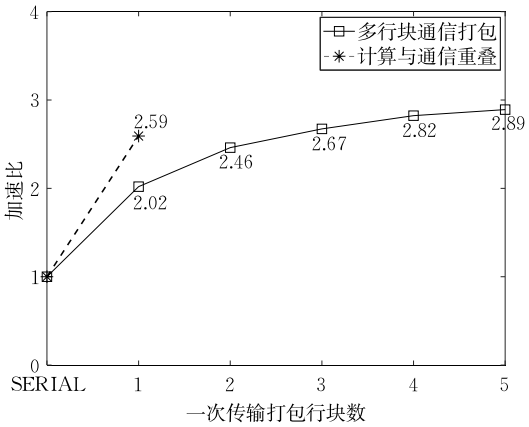

图 13　算例 2 基于并行优化的预处理算法加速比

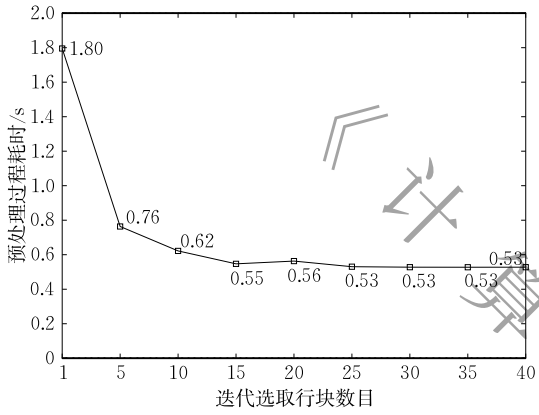

图 14　算例 1 低通信复杂性版本预处理过程耗时

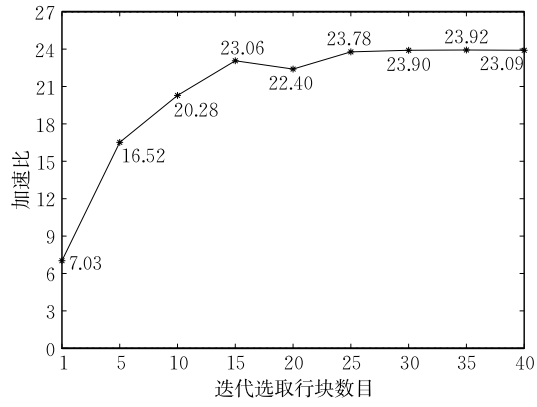

图 15　算例 1 低通信复杂性版本预处理过程加速比

降低了数值精度，导致总的迭代次数增加而计算时间增多. 迭代选取行块数目的取值需要考虑到两者的平衡，以达到最优加速比. 值得一提的是，图 14 显示低通信复杂性版本预处理过程耗时只有基于并行优化的预处理算法耗时的约 10%，虽然其耗时较短，然而会导致较多的 Krylov 迭代步数，最终抵消了性能提升部分，后文将具体讨论 Jacobian 系统的求解时间. 对于该算例，算法的加速性能基本上随着迭代选取行块数目增加而提升，直到其大于 30，算法性能开始趋于平稳，最终达到约 24 倍的加速比.

由于预处理的最终目的是将 Jacobian 系统的求解时间缩短，进一步测试本文提出的几种并行方式求解整个线性系统的耗时以验证算法的有效性. 以算例 1 的结果为例，图 16 为多行块通信打包和计算与通信重叠两个版本预处理算法求解线性 Jacobian 系统的耗时对比. 其中，纵坐标表示求解整个线性 Jacobian 系统时间，横坐标同图 10、11. 从图中可以看到，两个版本的并行方式均对 Jacobian 系统求解有较好的加速效果，相较于串行预处理算法，最高将计算时间减少了 55%. 图 17 为低通信复杂性块 Gauss-Seidel/Jacobi 算法求解线性 Jacobian 系统的总耗时. 不难发现，其对整个线性系统的加速效果稍差于基于并行优化的两个并行版本. 使用低通信复杂性块 Gauss-Seidel/Jacobi 算法的预处理过程耗时虽然较短，但由于其丢弃的数据过多，预处理效果不佳，Krylov 迭代收敛步数相对于基于并行优化的两个版本明显增多，最终导致其对整个线性系统的效果并不理想.

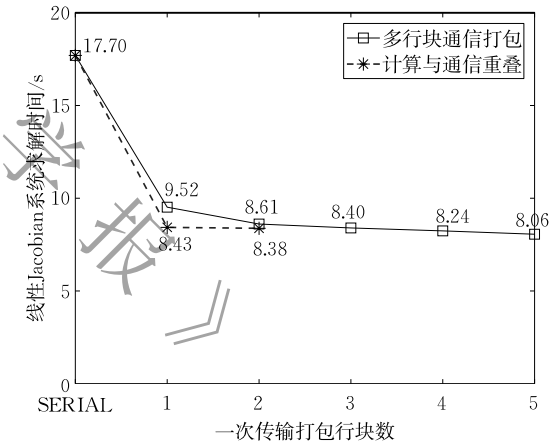

图 16　算法 1 求解 Jacobian 系统总耗时

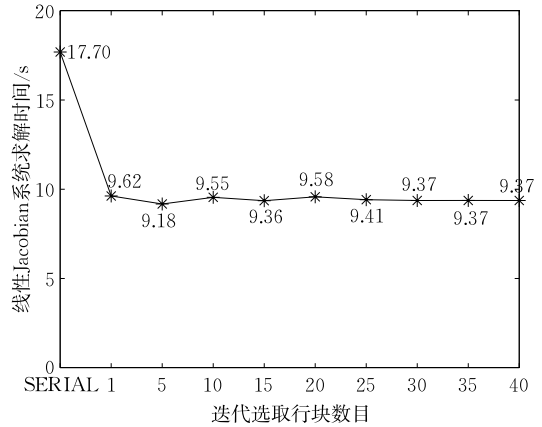

图 17　算法 2 求解 Jacobian 系统总耗时

4.4 可扩展性分析

可扩展性也是衡量并行算法性能的重要指标之一.基于强可扩展性和从核(CPE)可扩展性两个方面测试提出的并行算法.由于3个版本的优化算法均基于基本Gauss-Seidel/Jacobi算法,且基于并行和数值的优化均不影响其可扩展性,这里只测试最基本的Gauss-Seidel/Jacobi算法的强可扩展性.图18为Gauss-Seidel/Jacobi算法的强可扩展性测试图,其中,星号和方块标记的折线分别为算例1和算例2的可扩展曲线.该测试中每个核组(CGs)均使用64个从核(CPEs),以1024核的测试数据为基准,当处理器总核数数达到33280时,算例1、2的并行效率分别为61%和56%,可知该算法具有良好的强可扩展性.图19为改变每个核组(CGs)使用从核(CPEs)数量的可扩展性测试图,其中,将MPI进程数(MPE数)固定为32,实线上的数值为每个点的并行效率,星号和方块标记的折线分别为算例1和算例2的从核可扩展曲线.图中折线的上升则表示预处理时间缩短,可以看到,随着从核数目的增加,算例1的从核可扩展曲线呈上升趋势.而算例2的从核可扩展曲线在从核数为40～56之间时呈下降趋势,实测曲线与理想曲线有一定的差距,其主要由于随着从核数的增加,算法精度下降,迭代收敛步数的增加抵消了部分并行加速效果,而这一现象仅存在于对预处理算子精度敏感的迭代过程中.最终,当每个核组使用从核数目达到64时,并行效率保持在35%和29%.

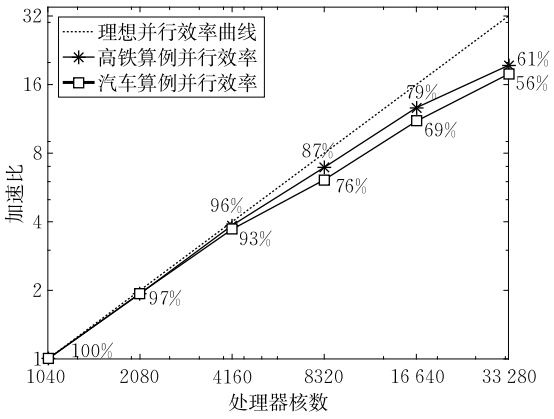

图18 基本块Gauss-Seidel/Jacobi算法的强可扩展性

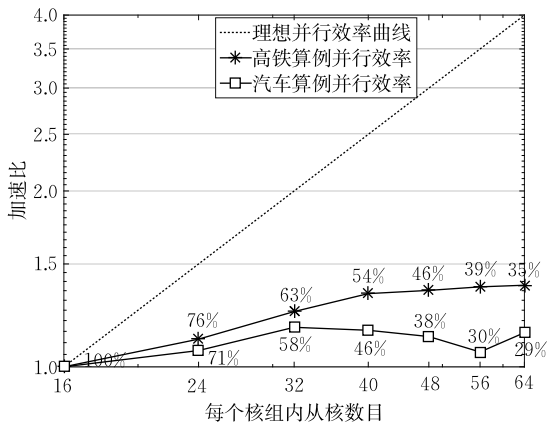

图19 从核可扩展性

4.5 算法通用性讨论

本文提出的基本块Gauss-Seidel/Jacobi算法作为一种新型预条件算法,具有通用性,其可在例如CPU+GPU、CPU+MIC和申威SW26010等异构众核架构上实现.本文对该算法采用的实现方式是面向申威SW26010异构众核芯片,并不一定适合其它异构众核架构.具体地,针对不同的异构众核架构,需要有针对性地选取不同的实现方式.此外,本文设计的几种低通信复杂性优化方法同样是面向SW26010芯片,旨在降低通信复杂性,但面向其他异构众核架构的优化也提供了参考.其中,多行块通信打包和通信与计算重叠优化可适用于需调用DMA通信的异构众核架构,而丢弃非关键元素优化作为一种数值方法则具有通用性.

5 结论

面向非结构网格问题,提出并实现了适用于异构众核架构的块Gauss-Seidel/Jacobi预处理算法.针对异构众核架构存在的通信瓶颈,设计了多行块通信打包、计算与通信重叠和丢弃非关键元素的低通信复杂性数值优化三种性能优化策略.选取神威太湖之光超级计算机作为算法的实现平台,并针对其国产异构众核芯片SW26010,实现了上述三种低通信复杂性的优化策略.使用多个真实算例进行算法测试的结果显示,块Gauss-Seidel/Jacobi算法具有良好的加速效果和可扩展性,对比串行Gauss-Seidel算法,其加速比最高达到4.16倍.使用1040核的测试数据为基准,在核数达到33280时,块Gauss-Seidel预处理算法的并行效率可达61%.

致 谢 感谢国家超级计算无锡中心给予作者实验测试环境.

参 考 文 献

[1] Chen R L, Cai X C. A scalable domain decomposition method and applications in simulation and optimization of fluids. Scientia Sinica Math, 2016, 46: 915-928(in Chinese)

(陈荣亮，蔡小川. 高可扩展区域分解算法及其在流体模拟和优化问题中的应用. 中国科学：数学，2016，46(7)：915)

[2] Cai X C, Keyes D E, Venkatakrishnan V. Newton-Krylov-Schwarz: An implicit solver for CFD. Institute for Computer Applications in Science and Engineering (ICASE), Technical Report: ICASE-95-87, 1995

[3] Cai X C, Sarkis M. A restricted additive Schwarz preconditioner for general sparse linear systems. SIAM Journal on Scientific Computing, 1999, 21(2): 792-797

[4] Fu H, Liao J, Yang J, et al. The Sunway TaihuLight supercomputer: System and applications. Science China Information Sciences, 2016, 59(7): 072001

[5] Grote M J, Huckle T. Parallel preconditioning with sparse approximate inverses. SIAM Journal on Scientific Computing, 1997, 18(3): 838-853

[6] Chow E, Patel A. Fine-grained parallel incomplete LU factorization. SIAM Journal on Scientific Computing, 2015, 37(2): C169-C193

[7] Yang C, Xue W, Fu H, et al. 10M-core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Salt Lake City, USA, 2016: 57-68

[8] Heuveline V, Lukarski D, Weiss J P. Enhanced parallel ILU(p)-based preconditioners for multi-core CPUs and GPUs: The power (q)-pattern method. Preprint Series of the Engineering Mathematics and Computing Lab, 2011 (08): 1-36

[9] Heuveline V, Lukarski D, Weiss J P. Scalable multi-coloring preconditioning for multi-core CPUs and GPUs//Proceedings of the European Conference on Parallel Processing. Berlin, Germany, 2010: 389-397

[10] Cotronis Y, Konstantinidis E, Louka M A, et al. Parallel SOR for solving the convection diffusion equation using GPUs with CUDA//Proceedings of the European Conference on Parallel Processing. Berlin, Germany, 2012: 575-586

[11] Di P, Wu H, Xue J, et al. Parallelizing SOR for GPGPUs using alternate loop tiling. Parallel Computing, 2012, 38(6): 310-328

[12] Saad Y. Iterative Methods for Sparse Linear Systems. Boston, USA: PWS Publishing Company, 1996

[13] Tang X Q, Wang L J, et al. Numerical Methods. Beijing: Science Press, 2015(in Chinese)
(唐旭清，王林君等. 数值计算方法. 北京：科学出版社，2015)

[14] Balay S, Abhyankar S, Adams M, et al. PETSc users manual revision 3.5. Argonne, Argonne National Laboratory, Technical Report: ANL-95/11 Rev. 3.5, 2014

[15] Chen R, Wu Y, Yan Z, et al. A parallel domain decomposition method for 3D unsteady incompressible flows at high Reynolds number. Journal of Scientific Computing, 2014, 58(2): 275-289

[16] Yan Z, Chen R, Cai X C. A scalable numerical method for simulating flows around high-speed train under crosswind conditions. Communications in Computational Physics, 2014, 15(4): 944-958

[17] Yan Z, Chen R, et al. Scalable numerical simulation of flows around car based on thousands of processors//Proceedings of the 9th China CAE Annual Conference. Huangshan, China, 2013: 1009-0134(in Chinese)
(闫争争，陈荣亮等. 基于数千核计算平台的汽车外流场可扩展并行数值模拟. 中国 CAE 工程分析技术年会. 安徽，黄山，2013：1009-0134)

WU Li-Lei, master candidate. His main research interests include domain decomposition method and high performance computing.

CHEN Rong-Liang, Ph. D., associate professor. His main research interests include partial differential equations, shape optimization problem, parallel computing.

LUO Li, Ph. D., assistant professor. His main research interests include computational fluid dynamics and parallel computing.

YAN Zheng-Zheng, Ph. D., assistant professor. His main research interests include computational fluid dynamics and parallel computing.

LIAO Zi-Ju, Ph. D., assistant professor. His main research interests include computational fluid dynamics and parallel computing.

CHI Li-Hua, Ph. D., associate professor. Her main research interests include parallel algorithm and high performance computing.

LIU Jie, Ph. D., professor. His main research interests include parallel algorithm, high performance computing, and machine learning.

Background

Heterogeneous many-core architecture provides a promising and practical approach to exascale computing, but also induces great challenge to the realization of classical numerical solvers. NKS is a fully-implicit solver which has been widely used in

fluid dynamic simulations over the decades. To make good use of the extreme parallelism provided by heterogeneous many-core architecture, NKS algorithms need to be redesigned. The subdomain preconditioners are the most time-consuming and computation-intensive part in NKS. Generally, ILU type and Gauss-Seidel type preconditioners are adopted. ILU type preconditioner has been efficiently parallelized on heterogeneous many-core architecture. However, it is highly non-trivial to extend this method to unstructured mesh problems. Instead, we focus on the Gauss-Seidel type method.

The previous works of parallelizing Gauss-Seidel type method are mostly for CPU+GPU architecture. It is necessary to develop a parallel Gauss-Seidel type method for general heterogeneous architecture. Besides, the missing of geometric information in matrices of unstructured mesh problems made the classical geometry-based parallel strategies fail.

Based on the algebra-based parallel strategy, a block Gauss-Seidel/Jacobi preconditioner for heterogeneous many-core architecture was proposed. The algorithm is used as the subdomain solver in NKS. To alleviate the memory bandwidth bottleneck in most heterogeneous chips, a series of low-communication-complexity optimization strategies were designed, such as multiple lines packed copying, computation-communication overlapping and the low-communication-complexity numerical optimization.

The Sunway TaihuLight supercomputer, ranked the first in the latest Top500 list, is exactly based on heterogeneous many-core architecture. Based on the homegrown SW26010 heterogeneous many-core architecture processor, the block Gauss-Seidel/Jacobi algorithm and its optimized version were modified and implemented. Numerical results show that the proposed block Gauss-Seidel/Jacobi algorithm delivers a 4.16x speedup comparing to the sequential version. For the parallel efficiency, the block Gauss-Seidel/Jacobi algorithm achieves a 61% efficiency as the number of processors increases from 1040 to 33280.

The research was supported in part by the Special Project on High-Performance Computing under the National Key R&D Program (No. 2017YFB0202104 and No. 2016YFB020060), the NSFC under Nos. 91530324, 91430218, 11602282, 61531166003, 11701547, and 11571100, and the Shenzhen Basic Research Nos. JCYJ20170307165328836, JCYJ20160331193229720, and JSGG20170824154458183.