·先进计算与数据处理· 文章编号:1000-3428(2022)01-0149-06 文献标志码:A 中图分类号:TP311.1

GRAPES动力框架中大规模稀疏线性系统并行求解及优化

张 琨[1],贾金芳[1],严文昕[1],黄建强[1,2],王晓英[1]
(1.青海大学 计算机技术与应用系,西宁 810016;2.清华大学 计算机科学与技术系,北京 100084)

摘 要: 赫姆霍兹方程求解是GRAPES数值天气预报系统动力框架中的核心部分,可转换为大规模稀疏线性系统的求解问题,但受限于硬件资源和数据规模,其求解效率成为限制系统计算性能提升的瓶颈。分别通过MPI、MPI+OpenMP、CUDA三种并行方式实现求解大规模稀疏线性方程组的广义共轭余差法,并利用不完全分解LU预处理子(ILU)优化系数矩阵的条件数,加快迭代法收敛。在CPU并行方案中,MPI负责进程间粗粒度并行和通信,OpenMP结合共享内存实现进程内部的细粒度并行,而在GPU并行方案中,CUDA模型采用数据传输、访存合并及共享存储器方面的优化措施。实验结果表明,通过预处理优化减少迭代次数对计算性能提升明显,MPI+OpenMP混合并行优化较MPI并行优化性能提高约35%,CUDA并行优化较MPI+OpenMP混合并行优化性能提高约50%,优化性能最佳。

关键词: 稀疏线性系统;广义共轭余差法;信息传递接口;OpenMP编程;统一计算架构

开放科学(资源服务)标志码(OSID):

中文引用格式: 张琨,贾金芳,严文昕,等.GRAPES动力框架中大规模稀疏线性系统并行求解及优化[J].计算机工程,2022,48(1):149-154,162.

英文引用格式: ZHANG K,JIA J F,YAN W X,et al.Parallel solution and optimization of large-scale sparse linear system in GRAPES dynamic framework[J].Computer Engineering,2022,48(1):149-154,162.

Parallel Solution and Optimization of Large-Scale Sparse Linear System in GRAPES Dynamic Framework

ZHANG Kun[1],JIA Jinfang[1],YAN Wenxin[1],HUANG Jianqiang[1,2],WANG Xiaoying[1]
(1. Department of Computer Technology and Applications,Qinghai University,Xining 810016,China;
2. Department of Computer Science and Technology,Tsinghua University,Beijing 100084,China)

【Abstract】 The Helmholtz equation is the core of dynamic framework of Global and Regional Assimilation Prediction System(GRAPES) for numerical weather forecast.This equation can essentially be transformed into the solution of a large-scale sparse linear system,but the solution efficiency is limited by hardware resources and scaling data size,and becomes a bottleneck of the system computing performance.This paper explores three parallel methods(MPI,MPI+OpenMP and CUDA) of implementing the Generalized Conjugate Residual(GCR) method for solving large-scale sparse linear equations.At the same time,the ILU preconditioner is used to optimize the number of conditions of the coefficient matrix,which speeds up the convergence of the iterative method.In the CPU parallel scheme,MPI is responsible for coarse-grained parallelism and communication between processes,and OpenMP introduces shared memory to achieve fine-grained parallelism within the process.In the GPU parallel scheme,the CUDA model uses the optimization approaches of data transmission,coalesced access and shared memory.Experimental results show that the performance of MPI+OpenMP hybrid parallel optimization is about 35% higher than that of MPI parallel optimization,and the performance of CUDA parallel optimization is about 50% higher than that of MPI+OpenMP hybrid parallel optimization,which gets the best performance.

【Key words】 sparse linear system;Generalized Conjugate Residual(GCR) method;Message Passing Interface(MPI);OpenMP programming;Compute Unified Device Architecture(CUDA)

DOI: 10.19678/j.issn.1000-3428.0060080

基金项目: 国家自然科学基金(61762074,62062059);青海省科技计划(2019-ZJ-7034);教育部"春晖计划"科研基金(QDCH2018001)。

作者简介: 张 琨(1997—),男,硕士研究生,主研方向为高性能计算;贾金芳(通信作者),讲师、硕士;严文昕,硕士研究生;黄建强,副教授、博士研究生;王晓英,教授、博士。

收稿日期: 2020-11-23 **修回日期:** 2021-01-17 **E-mail:** 543860105@qq.com

0 概述

全球区域同化预报系统(Global and Regional Assimilation Prediction System,GRAPES)是我国自主研发的新一代数值天气预报系统,能够改善天气预报模式的性能并有效提高天气预报精度。GRAPES动力框架的计算核心是赫姆霍兹方程的求解,该方程是原始大气方程组经过一系列离散化处理之后形成的大规模稀疏线性方程组。目前对线性方程组的求解方法主要有直接法和迭代法[1-2]两种。由于计算机硬件资源的限制,使用直接法并不能完成有效求解,而迭代法具有硬件存储空间要求较小、原始矩阵在计算过程中保持不变等优点。因此,目前求解大规模稀疏线性方程组主要选择迭代法。

GRAPES数值天气预报系统采用的迭代法为广义共轭余差(Generalized Conjugate Residual,GCR)法。文献[3]利用基准测试程序对GRAPES并行应用进行分析,发现在迭代法计算过程中全局通信对整个系统的性能影响较大,指出在进行性能优化时需要考虑系统通信所带来的性能下降问题。文献[4]针对赫姆霍兹方程求解过程中通信占比高的现象,引入短向量计算替代长向量计算,以减少通信开销。文献[5]在使用重启动和截断两种方式进行性能优化的同时,还使用一种新的稀疏近似逆预条件子来加快算法收敛速度。

研究发现,增加重启动和截断处理的广义共轭余差法能够减少数据存储量,同时改善计算的局部性。文献[6]为改善GRAPES模式动力框架中存在的极点问题,利用阴阳网格设计新的动力框架,与原GRAPES模式框架相比,新的动力框架表现出较好的数值稳定性及计算性能。文献[7]利用MPI和OpenMP混合并行的方式提升GRAPES模式的并行度,以适应主流硬件架构既有分布式内存又有共享内存的情况。文献[8]结合国产高性能平台实现了基于MPI+共享变量编程模型的众核线程级并行的多级并行方案,提升了方程求解的运行效率。文献[9]基于PETSc科学计算库和Hypre预条件库实现了广义最小残差(Generalized Minimal Residual,GMRES)法,与GRAPES模式现有的GCR算法相比,GMRES算法在更高分辨率的条件下具有较好的性能。文献[10]则利用CPU设备结合线程级并行、矢量化和数据重用技术提升了算法计算性能。当前GRAPES模式动力框架中广义共轭余差法的优化工作主要集中在通用CPU及国产高性能机器领域,部分学者讨论了算法本身的改进方案,但大多是与CPU设备进行兼容,而结合GPU的性能优化相对较少[11]。

本文针对GRAPES数值天气预报系统中的赫姆霍兹方程求解问题,分别通过MPI、MPI+OpenMP、CUDA三种并行方式实现求解大规模稀疏线性方程组的广义共轭余差法,并对测试结果进行对比分析,评估优化性能。

1 GRAPES模式中大规模稀疏线性方程组求解

GRAPES数值天气预报系统主要包括动力框架和物理过程参数化方案两部分[12-13],其中动力框架为核心部分,将动力框架方程组进行简化和离散后可得到GRAPES模式的基本预报方程[14-16]。对该方程进行处理后,动力框架的主要计算内容就变成对一个包含大型稀疏矩阵的赫姆霍兹方程的求解[17-19]。如图1所示,赫姆霍兹方程的求解过程计算量庞大,占据整个过程求解计算量的30%以上。随着模式分辨率的提高,方程的计算量还会增加。因此,对赫姆霍兹方程进行高效求解成为动力框架性能提升的关键[20]。

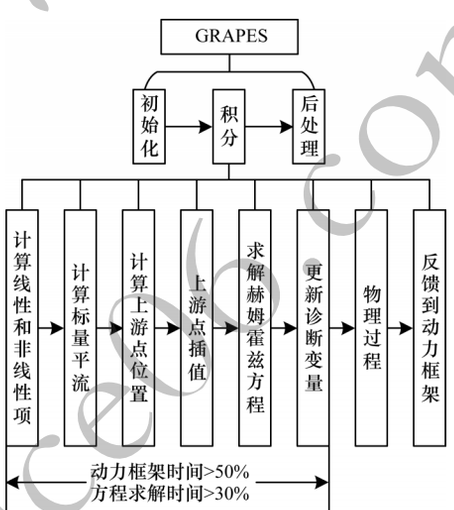


图1 GRAPES模式核心计算

Fig.1 GRAPES core computing

如图2所示,GRAPES模式动力框架部分的赫姆霍兹方程中计算每一个格点的方程需要选取空间中与其相关的19个格点作为系数[21]。

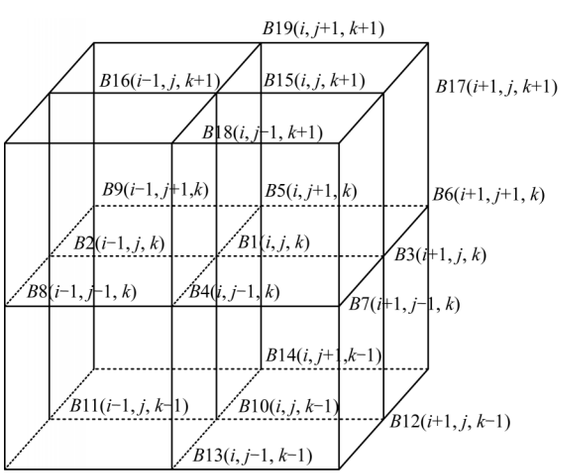


图2 赫姆霍兹方程系数空间分布

Fig.2 Spatial distribution of the coefficients of Helmholtz equation

空间格点的计算公式如下:

$$\begin{aligned}\left(\xi_{\Pi 0}\right)_{i,j,k}=&B_1(\Pi)_{i,j,k}+B_2(\Pi)_{i-1,j,k}+B_3(\Pi)_{i+1,j,k}+\\&B_4(\Pi)_{i,j-1,k}+B_5(\Pi)_{i,j+1,k}+B_6(\Pi)_{i+1,j+1,k}+\\&B_7(\Pi)_{i+1,j-1,k}+B_8(\Pi)_{i-1,j-1,k}+B_9(\Pi)_{i-1,j+1,k}+\\&B_{10}(\Pi)_{i,j,k-1}+B_{11}(\Pi)_{j-1,j,k-1}+B_{12}(\Pi)_{i+1,j,k-1}+\\&B_{13}(\Pi)_{i,j-1,k-1}+B_{14}(\Pi)_{i,j+1,k-1}+B_{15}(\Pi)_{i,j,k+1}+\\&B_{16}(\Pi)_{i-1,j,k+1}+B_{17}(\Pi)_{i+1,j,k+1}+B_{18}(\Pi)_{i,j-1,k+1}+\\&B_{19}(\Pi)_{i,j+1,k+1}\end{aligned}\tag{1}$$

将每一个空间格点的计算进行组合后可以得到一个大规模的稀疏线性方程组,该方程组可简化为:

$$\boldsymbol{Ax}=\boldsymbol{b}\tag{2}$$

其中:$\boldsymbol{A}$是一个大规模稀疏矩阵;$\boldsymbol{x}$是规模为空间总格点数的解向量;$\boldsymbol{b}$是与$\boldsymbol{x}$相对应的方程右端向量。因此,GRAPES模式动力框架中赫姆霍兹方程的求解就是对一个大规模稀疏线性方程组的求解。在当前系统中,方程求解所采用的算法为广义共轭余差法,在满足精度要求的前提下,广义共轭余差法算法收敛较快,整体算法易于实现,综合表现较好[22]。广义共轭余差法的具体内容如算法1所示。

算法1　广义共轭余差法

输入　矩阵$\boldsymbol{A}$,初始解$\boldsymbol{x}_0$,右端向量$\boldsymbol{b}$

输出　近似解$\boldsymbol{x}$

1.$r_0=b-Ax_0$,$p_0=r_0$

2.for i=1:maximum number of iterations do

3.$\alpha_{i-1}=(r_{i-1},Ap_{i-1})/(Ap_{i-1},Ap_{i-1})$

4.$x_i=x_{i-1}+\alpha_i p_{i-1}$

5.$r_i=r_{i-1}-\alpha_i Ap_{i-1}$

6.if converged then exit;

7.for j=int[(i-1)/k]k,…,i-1 do

8.$\alpha_{ij}=-(Ar,Ap_j)/(Ap_j,Ap_j)$

9.end for

10.$p_i=r_i+\sum_{j=\mathrm{int}[(i-1)/k]k}^{i-1}\alpha_{ij}p_j$

11.$Ap_i=Ar_i+\sum_{j=\mathrm{int}[(i-1)/k]k}^{i-1}\alpha_{ij}Ap_j$

12.end for

2　广义共轭余差法的并行实现

2.1　问题模型

本文主要考虑问题模型$\boldsymbol{Ax}=\boldsymbol{b}$,等式中的$\boldsymbol{A}$为稀疏矩阵,$\boldsymbol{x}$为需要求解的向量,$\boldsymbol{b}$为右端向量。系数矩阵具有明显的稀疏性,在矩阵存储格式的选择上,统一使用CSR(Compressed Sparse Row)格式[23-24]。方程求解算法选择广义共轭余差法,算法主要计算内容包括稀疏矩阵向量乘、向量内积、向量数乘。

2.2　系数矩阵预处理优化

迭代法具有依赖系数矩阵条件数的特点,可能存在收敛速度慢甚至不收敛的情况,因此,通常使用预处理技术对算法进行优化。目前的预处理技术主要通过构建预条件子将原方程$\boldsymbol{Ax}=\boldsymbol{b}$转化为$\boldsymbol{M}^{(-1)}\boldsymbol{Ax}=\boldsymbol{M}^{(-1)}\boldsymbol{b}$等形式求解。常用的预条件子构造方法有稀疏近似逆预条件、不完全分解预条件等。本实验所采用的预条件技术为不完全分解LU预条件子(ILU)。预处理过程将原始矩阵$\boldsymbol{A}$分为上三角矩阵$\boldsymbol{U}$、下三角矩阵$\boldsymbol{L}$及残差矩阵$\boldsymbol{R}$,分解之后的矩阵可以满足不同条件$\boldsymbol{P}$,例如与原系数矩阵的稀疏结构保持一致。构造ILU预条件子的过程如算法2所示。

算法2　ILU分解

输入　矩阵$\boldsymbol{A}$,分解条件$\boldsymbol{P}$

输出　预处理子$\boldsymbol{M}$

1.for i=2,3,…,n do

2.for k=1,2,…,i-1 and (i,k)∉P do

3.$A_{ik}=A_{ik}/A_{kk}$

4.for j=k+1,…,n and (i,j) ∉P do

5.$A_{ij}=A_{ij}-A_{ik}*A_{kj}$

6.end for

7.end for

8.end for

2.3　MPI并行及MPI+OpenMP混合并行

通信开销是MPI程序中不可忽视的部分,尤其是随着进程规模的增大,在计算时间减少的同时也会增加程序的通信开销[25-26]。因此,在设计MPI并行算法时,需要平衡计算任务划分与通信函数使用之间的关系。在广义共轭余差法中,每一个迭代步骤之间都存在数据依赖,并行任务主要选择在每一个迭代步骤内部进行。MPI并行方式将计算任务根据分配的进程数量进行划分,每个进程处理不同的子任务,最后利用进程间通信进行数据同步。

本文实验所测试的MPI+OpenMP混合并行方式,在MPI并行方式的基础上,利用多线程实现了更细粒度的任务划分。OpenMP是共享内存并行编程模式,与MPI并行模式相比可减少数据通信的时间。在矩阵向量乘部分的并行处理中,MPI+OpenMP混合并行任务划分如图3所示。矩阵数据按行子域划分为不同的row_part,每个MPI进程只计算所属row_part与向量的乘积,各进程并行执行计算任务。同时在MPI进程内部利用OpenMP模式将row_part划分为更细粒度的subrow_part,结合线程级并行完成子任务计算。

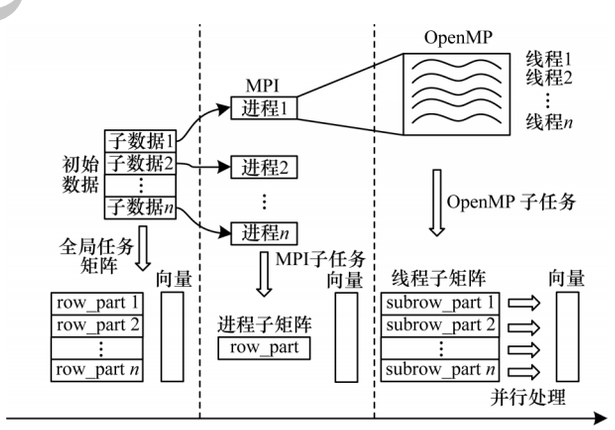


图3　MPI+OpenMP混合并行结构

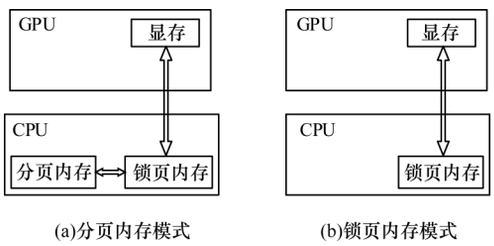
Fig.3　MPI + OpenMP hybrid parallel structure

2.4　GPU并行优化

2.4.1　数据传输优化

利用CUDA并行求解方程时主要开销在于GPU与CPU之间的数据传输,为减少数据传输时间开销,避免

PCI-E总线上的数据传输成为程序性能提升的瓶颈,将矩阵$\boldsymbol{A}$以及各个中间向量等都保存在GPU上。在计算过程中利用显存进行通信,避免和主机端进行频繁的数据传输。无法避免的数据传输过程只出现在需要进行收敛判断及主机端计算的情况下,同时在数据传输时采用页锁定内存(pinned memory)进一步提高数据传输速度。如图4所示,与可分页内存(pageable memory)相比,页锁定内存能够减少一次数据传输操作,提高数据传输效率。



(a)分页内存模式 (b)锁页内存模式

图4 内存模式对比

Fig.4 Comparison of memory modes

2.4.2 访存优化

GPU对全局存储器的访问速度也会对程序性能造成一定影响,利用CUDA模型可以实现线程块间及线程块内两级并行,从而完成对全局存储器的合并访问。初始的数据存储格式由于将需要连续访问的数据离散存储,不满足线程合并访问的条件,因此需要对数据存储格式进行转换。将三维格式的矩阵转化为一维格式存储之后,每一个网格点在计算过程中同一线程束内的线程所访问的内存空间是连续的,满足合并访存的要求。如图5所示,当同一线程束内的线程访存连续时,能够将多次访存操作减少为一次访存操作,从而减少计算过程中的访存开销。

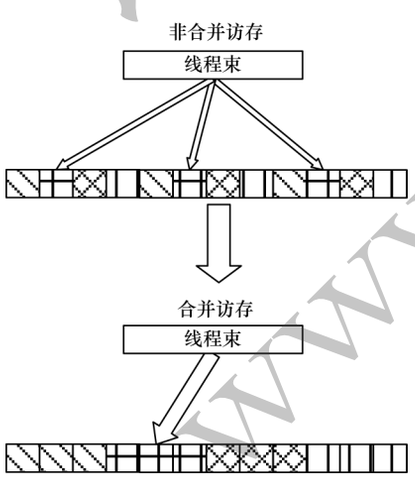


图5 合并访存

Fig.5 Coalesced access

2.4.3 存储器优化

在GPU计算过程中,存在一部分数据只需要线程块内部的线程访问。对于这一部分数据,利用共享存储器进行保存。共享存储器的访问延迟比全局存储器低,在计算过程中线程块内部的线程只需要访问共享存储器中,而不需要去访问全局存储器,从而进一步提高访存的速度。

在向量内积的计算过程中,并行方式如图6所示。内积计算过程可以表示成$\sum_{i=1}^{n}a_ib_i$形式,向量$\boldsymbol{a}$和向量$\boldsymbol{b}$分别对应图中Vector a和Vector b。当核函数启动后,将向量子任务subVector分配到不同的线程块中。由于向量内积涉及到对同一内存地址的修改操作,因此先利用线程块访问各自私有的共享存储器进行局部规约操作,这一部分计算内容可以并行执行。当所有subVector都完成局部归约后,再进行最后的规约操作,从而提高整体的计算速度。

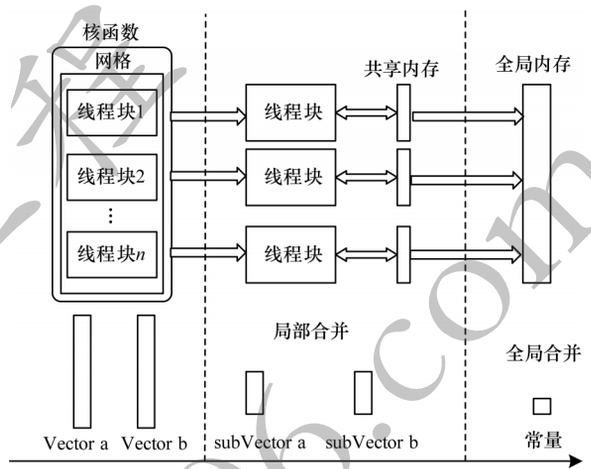


图6 CUDA内积并行结构

Fig.6 CUDA dot parallel structure

3 实验与分析

3.1 实验环境及测试数据

实验中共使用5个CPU节点,节点处理器为Intel® Xeon® CPU E5-2692 v2 @ 2.20 GHz。每个节点两块CPU(共24核心),节点内存为64 GB;实验使用GPU为Tesla T4,显存容量为16 GB;CUDA版本为v10.1。实验所使用的测试数据包含系数矩阵$\boldsymbol{A}$、初始解$\boldsymbol{x}_0$、右端向量$\boldsymbol{b}$,其中系数矩阵$\boldsymbol{A}$规模为360×180×38,共2 462 400个网格点。

3.2 结果分析

实验测试了数据在不同计算方式下的计算残差。结果表明,使用3种并行方式实现的广义共轭余差法与串行方式能够得到一致的计算结果,验证了不同并行方式求解稀疏线性方程组的正确性和有效性。

3.2.1 预处理优化结果

对原始GCR算法进行预处理优化后,方程的收敛速度获得大幅提升。如图7所示,在误差精度要求为1E−10的条件下,预处理算法收敛所需的迭代次数从134次下降至21次。ILU预处理的引入能够改善方程的求解性能,同时也会增加新的计算内容。但是在并行算法中,计算量已经不是限制性能的最大因素,例如在MPI程序中,随着并行度的增加,进程之间的通信成为限制程序性能的最大因素。在利用MPI实现的并行GCR算法中,每一次迭代过程都

会进行数据通信,所以,利用预处理技术加快算法收敛速度之后,整体的计算性能可以获得提升。

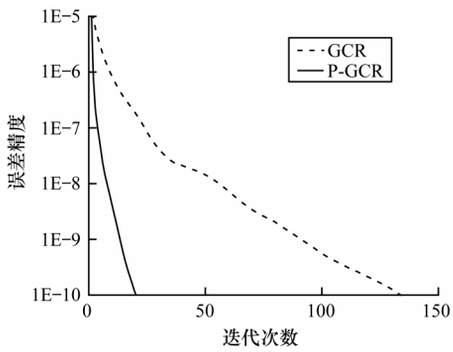


图7　预处理优化对计算性能的影响

Fig.7　Effect of preprocessing optimization on computational performance

3.2.2　MPI并行结果

虽然使用MPI并行方式能够通过提高进程数目减少子任务的计算量,从而有效提升计算部分速度,但是也会增加数据通信的时间开销。MPI并行算法在实现过程中避免了冗余的通信开销,只对部分必要的向量更新操作进行通信。MPI并行算法在不同进程规模下的运行时间如图8所示。可以看出,随着进程规模的增加,运行时间呈先下降后上升的趋势。这是因为MPI算法的进程数量会影响进程之间的通信开销,当进程数为32时,并行计算任务所带来的性能提升不足以抵消增加的通信开销,程序整体性能就会下降。由于MPI并行方式存在可扩展性问题,因此只通过增加进程数量的方式来提高性能并不能高效地利用计算资源。

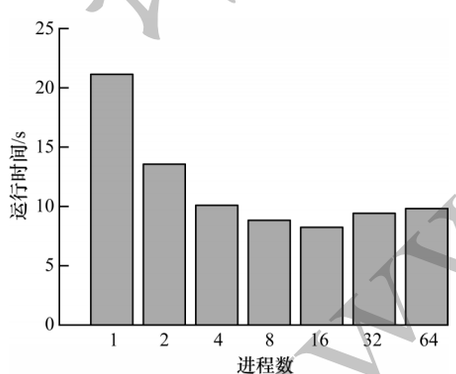


图8　MPI运行时间

Fig.8　Runtime of MPI

3.2.3　MPI+OpenMP混合并行结果

与MPI并行方式相比,MPI+OpenMP混合并行方式充分结合了共享内存的优势,在不改变原始MPI并行方式粗粒度通信的情况下,提高了进程内部的线程级并行度。图9为MPI并行与MPI+OpenMP混合并行在不同核数规模下的计算性能对比。可以看出,当计算所使用的核数过少时,利用OpenMP进行更细粒度的并行划分之后性能并没有明显改善,但随着核数的增加,OpenMP的共享内存优势得以发挥,计算性能相比MPI并行方式有一定提升。当核数为64时,MPI并行方式已经出现性能下降,但是MPI+OpenMP混合并行方式的性能依然较稳定。因此,在MPI并行方式由于可扩展性原因造成计算效率下降的情况下,使用MPI+OpenMP混合并行方式可以获得更好的性能。

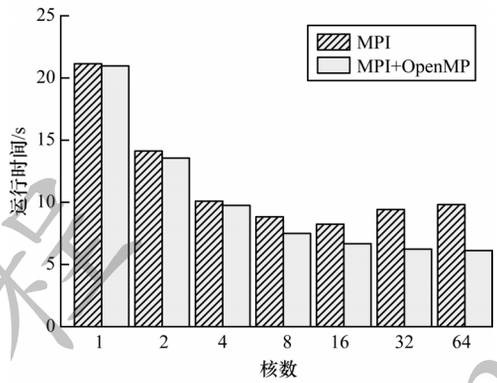


图9　MPI与MPI+OpenMP运行时间对比

Fig.9　Runtime comparison between MPI and MPI+OpenMP

3.2.4　GPU并行结果

基于GPU的CUDA并行方式在优化措施上采用了共享存储器优化、全局存储器访存优化以及主从设备数据传输优化。CUDA并行与MPI+OpenMP混合并行在算法求解过程中主要步骤的性能对比如图10所示。与MPI+OpenMP混合并行相比,CUDA并行在矩阵向量乘部分(ar)有32%的性能提升,向量数乘部分(p&ap,x&r)有88%的性能提升,向量内积部分(beta,alpha,residual)有80%的性能提升。

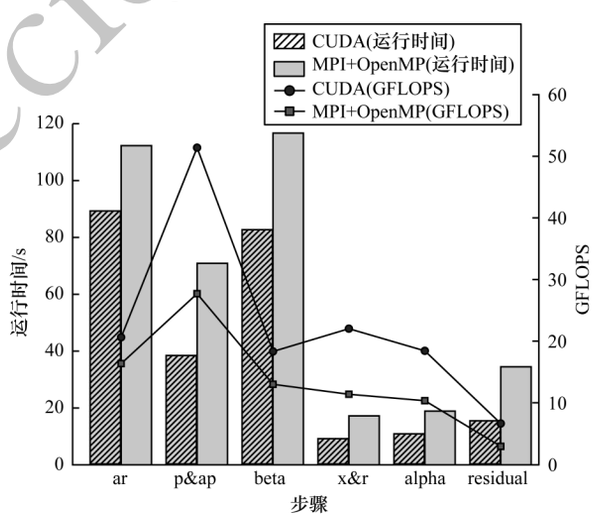


图10　CUDA与MPI+OpenMP主要步骤计算性能对比

Fig.10　Performance comparison of main steps of CUDA and MPI+OpenMP

3.2.5　优化性能对比

对不同优化方式的优化性能进行对比,结果如图11所示。可以看出,通过预处理优化减少算法迭代次数对整体计算性能提升明显,加速比达到4.5。

在预处理的基础上，使用MPI并行方式的性能获得提升，加速比达到11.7。结合OpenMP并行方式进行细粒度任务划分之后，混合并行计算性能相对于MPI并行方式有一定提升，加速比达到15.8。同时数据也显示CUDA并行方式相对于MPI+OpenMP混合并行方式能够获得约50%的性能提升，加速比达到23.4。基于GPU的并行方式在计算时间上的开销远小于基于CPU的并行方式，这得益于GPU数量众多的计算核心。在数据通信方面，GPU并行方式线程级的通信相比于CPU并行方式也具有优势，不需要进行各个进程之间的数据通信，所以基于GPU的CUDA并行方式能获得较好的计算性能。

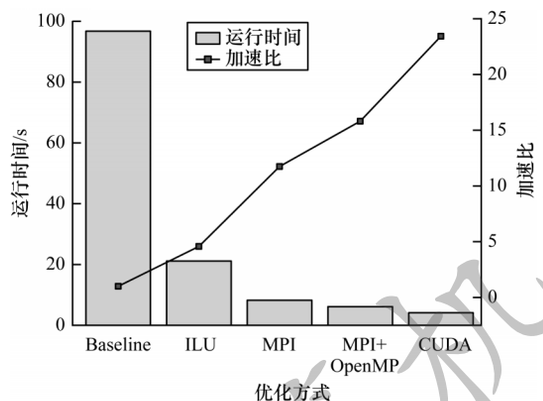


图11 不同并行方式的优化性能对比

Fig.11 Optimization performance comparison of different parallel methods

4 结束语

大规模线性方程组的求解作为科学计算中的核心问题，长期以来都是研究者关注的重点，如何充分利用目前的高性能计算机资源对求解问题的计算性能进行优化是重要的研究方向。本文从GRAPES数值天气预报系统动力框架中的赫姆霍兹方程求解问题出发，使用不同并行方式对广义共轭余差法进行并行及优化，并对计算性能进行对比分析。实验结果表明，在MPI并行、MPI+OpenMP混合并行及CUDA并行3种优化方式中，基于GPU的CUDA并行方式能够获得更好的计算性能。后续将建立多机混合并行的优化模型，测试MPI+CUDA的多节点优化性能，同时还将分析多重网格预处理、稀疏近似逆预处理等其他预处理方式对迭代法收敛速度的影响。

参考文献

[1] AHAMED A K C, MAGOULES F. Iterative methods for sparse linear systems on graphics processing unit[C]// Proceedings of the 14th International Conference on High Performance Computing and Communication and the 9th International Conference on Embedded Software and Systems. Washington D. C., USA: IEEE Press, 2012: 836-842.

[2] VARGA R S, GILLIS J. Matrix iterative analysis[M]. Berlin, Germany: Springer, 1963.

[3] 翟琰，翟季冬，薛巍，等. 大规模并行程序通信性能分析[J]. 华中科技大学学报(自然科学版)，2011，39(S1)：71-75.
ZHAI Y, ZHAI J D, XUE W, et al. Massive parallel program communication performance analysis[J]. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2011, 39(S1): 71-75. (in Chinese)

[4] 杨磊. 通信避免的广义共轭余差算法[D]. 北京：中国气象科学研究院，2019.
YANG L. Communication avoiding generalized conjugate residual method [D]. Beijing: Chinese Academy of Meteorological Sciences, 2019. (in Chinese)

[5] 张理论. 面向气象预报数值模式的高效并行计算研究[D]. 长沙：国防科学技术大学，2002.
ZHANG L L. Research of high-performance parallel computing for numerical models in meteorologic prediction [D]. Changsha: National University of Defense Technology, 2002. (in Chinese)

[6] LI X, PENG X, LI X. An improved dynamic core for a non-hydrostatic model system on the Yin-Yang grid [J]. Advances in Atmospheric Sciences, 2015, 32(5): 648-658.

[7] 蒋沁谷，金之雁. GRAPES全球模式MPI与OpenMP混合并行方案[J]. 应用气象学报，2014，25(5)：581-591.
JIANG Q G, JIN Z Y. The hybrid MPI and OpenMP parallel scheme of GRAPES_global model[J]. Journal of Applied Meteorological Science, 2014, 25(5): 581-591. (in Chinese)

[8] 刘钊. 基于国产高性能计算机的GRAPES性能优化研究[D]. 上海：上海交通大学，2014. (in Chinese)
LIU Z. Study of GRAPES numerical weather prediction system optimization on domestic high performance computers[D]. Shanghai: Shanghai Jiaotong University, 2014. (in Chinese)

[9] 伍湘君. GRAPES高分辨率气象数值预报模式并行计算关键技术研究[D]. 长沙：国防科学技术大学，2011.
WU X J. Study on the parallel computing in GRAPES high resolution numerical weather prediction model [D]. Changsha: National University of Defense Technology, 2011. (in Chinese)

[10] HUANG J, XUE W, BIAN H, et al. Helmholtz solving and performance optimization in global/regional assimilation and prediction system[J]. Tsinghua Science and Technology, 2021, 26(3): 335-346.

[11] YAN W X, JIA J F, HUANG J Q, et al. Research of GRAPES numerical weather prediction model [C]// Proceedings of the 4th High Performance Computing and Cluster Technologies Conference and the 3rd International Conference on Big Data and Artificial Intelligence. New York, USA: ACM Press, 2020: 34-41.

[12] ZOU Y, XUE W, LIU S. A case study of large-scale parallel I/O analysis and optimization for numerical weather prediction system [J]. Future Generation Computer Systems, 2014, 100(37): 378-389.

[13] SHEN X, WANG J, LI Z, et al. Research and operational development of numerical weather prediction in China[J]. Journal of Meteorological Research, 2020, 34(4): 675-698.

[14] CHEN D, XUE J, YANG X, et al. New generation of multi-scale NWP system(GRAPES): general scientific design[J]. Chinese Science Bulletin, 2008, 53(22): 3433-3445.

[15] XU G, CHEN D, XUE J, et al. The program structure designing and optimizing tests of GRAPES physics[J]. Chinese Science Bulletin, 2008, 53(22): 3470-3476.

（下转第162页）

（上接第154页）

[16] HUANG B, CHEN D, LI X, et al. Improvement of the semi-Lagrangian advection scheme in the GRAPES model: theoretical analysis and idealized tests[J]. Advances in Atmospheric Sciences, 2014, 31(3): 693-704.

[17] 陈德辉, 沈学顺. 新一代数值预报系统GRAPES研究进展[J]. 应用气象学报, 2006, 17(6): 773-777.
CHEN D H, SHEN X S. Recent progress on GRAPES research and application[J]. Journal of Applied Meteorological Science, 2006, 17(6): 773-777. (in Chinese)

[18] QIAN J H, SEMAZZI F H M, SCROGGS J S. A global nonhydrostatic semi-Lagrangian atmospheric model with orography[J]. Monthly Weather Review, 1998, 126(3): 747-771.

[19] ROBERT A. A semi-Lagrangian and semi-implicit numerical integration scheme for the primitive meteorological equations[J]. Journal of the Meteorological Society of Japan, 1982, 60(1): 319-325.

[20] 宋君强, 伍湘君, 张理论, 等. GRAPES模式中Helmhothz方程两种求解方法的对比研究[J]. 计算机工程与科学, 2011, 33(11): 65-70.
SONG J Q, WU X J, ZHANG L L, et al. A study of the two Helmholtz solvers in the GRAPES model using GCR and GMRES[J]. Computer Engineering and Science, 2011, 33(11): 65-70. (in Chinese)

[21] 薛纪善, 陈德辉. 数值预报系统GRAPES的科学设计与应用[M]. 北京: 科学出版社, 2008.
XUE J S, CHEN D H. Scientific design and application of numerical prediction system GRAPES[M]. Beijing: Science Press, 2008. (in Chinese)

[22] SAAD Y. Iterative methods for sparse linear systems[M]. [S. l.]: Society for Industrial and Applied Mathematics, 2003.

[23] KREUTZER M, HAGER G, WELLEIN G, et al. A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units[J]. SIAM Journal on Scientific Computing, 2014, 36(5): 401-423.

[24] ZARDOSHTI P, KHUNJUSH F, SARBAZI-AZAD H. Adaptive sparse matrix representation for efficient matrix-vector multiplication[J]. The Journal of Supercomputing, 2016, 72(9): 3366-3386.

[25] CHUNDURI S, PARKER S, BALAJI P, et al. Characterization of MPI usage on a production supercomputer[C]//Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis. Washington D. C., USA: IEEE Press, 2018: 386-400.

[26] PANDA D K, SUBRAMONI H, CHU C, et al. The MVAPICH project: transforming research into high-performance MPI library for HPC community[J]. Journal of Computational Science, 2020, 52(1): 1-10.

编辑 金胡考