一种改进的并行 Orthodir(m)算法

左定喜 吴 帆 李肯立
(湖南大学信息科学与工程学院 长沙 410082)

摘 要 通过将 Orthodir(m)算法的两个向量内积改成几个连续内积,改变算法数据相关性,提出了改进的 Orthodir(m)算法(IOrthodir(m)算法)。改进的算法具有与原算法相同的收敛性。理论分析表明,当处理器数目达到一定数量时,IOrthodir(m)算法计算速度快于原算法,扩展性方面也要优于 Orthodir(m)算法。实验证实,IOrthodir(m)算法优于 Orthodir(m)算法。

关键词 Orthodir(m),并行,非对称,稀疏线性方程组,Krylov
中图法分类号 TP391.9 **文献标识码** A

Improved Parallel Orthodir(m) Algorithm

ZUO Ding-xi WU Fan LI Ken-li
(Information Science and Engineering College, Hunan University, Changsha 410082, China)

Abstract By changing two inner product of vectors into several continuous inner products, this paper changed the data dependency and puted forward an improved Orthodir(m) algorithm(IOrthodir(m)). Two algorithms share the same convergence. Theory analysis shows that when the processors reach a certain number, IOrthodir(m) acts better than Orthodir(m). Meanwhile IOrthodir(m) is better than Orthodir(m) in extensibility. Experimental results also show that IOrthodir(m) is superior to Orthodir(m).

Keywords Orthodir(m), Parallel computing, Asymmetric, Sparse linear system, Krylov

1 引言

实际生产生活中的科学工程计算问题都转化为求解方程组 $Ax=b$。高效、快速地求解该问题既有理论意义又有实际价值。Krylov 子空间算法是目前最高效的求解稀疏线性方程组的算法。该方法存储量小、计算量小且易于并行等,适合于求解稀疏线性方程组。关于 Krylov 子空间的并行算法,前人做了很多工作。Dazevedo[4]对 CG 算法进行了修改,得到了改进的 CG 算法。Merurant[5]和 Saad[6]在减少 CG 算法的同步开销方面做了很多工作。刘[7]提出的改进适合并行计算的 CR 算法,同步开销减少一半,提高了效率。张[2]提出了一种适合于分布式并行计算的平方共轭残差法,其将两个全局同步化点降低为一个,提高了并行性。

Orthodir 方法是一种 Krylov 子空间算法,是由 Young 和 Jea[8]提出的,类似于 GCR 算法。Orthodir(m)算法是 Orthodir 算法的截断算法。对于非对称矩阵,Orthodir 算法不具有可行性,因为迭代计算会占用很大的存储空间来保存之前计算得到的数据以用于下一次计算,这通常是不可接受的。截断算法 Orthodir(m)只需要保存很少的几个向量用于下一次迭代计算(m 取值一般不大)。Orthodir(m)算法具有 Orthodir 算法的大部分性质。当矩阵对称或者反对称时,两方法等价[9]。Orthodir(m)方法的计算主要包括向量间内积和矩阵向量之间的乘积。矩阵向量乘积的通信成为并行计算的瓶颈。本文提出了一种改进算法(IOrthodir(m)算法),IOrthodir(m)算法将分离的内积数学变换为可并行执行的连续内积,减少了通信次数。IOrthodir(m)算法的数值稳定性与原算法相同,同步开销减少为原来的一半。

2 算法描述

对于给定的线性方程组 $Ax=b$,其中矩阵 $A\in R^{N*N}$ 为非对称非奇异矩阵,$b\in R^N$,$x\in R^N$。算法 1 为 Orthodir(m)算法。分析算法 1 可知,在并行求解时,步(2)和步(5)的内积计算需要进行全局同步,同时具有数据相关性,上一个计算不出来,下一个计算便无法进行,从而增加了时间开销,影响算法性能。

算法 1 Orthodir(m)算法

(1) $r_0=b-Ax_0$, $p_0=r_0$

对 k=0,1,2…直到收敛计算:

(2) $\alpha_k=(r_k,Ap_k)/(Ap_k,Ap_k)$

(3) $x_{k+1}=x_k+\alpha_k p_k$

(4) $r_{k+1}=r_k-\alpha_k p_k$

到稿日期:2012-10-19 返修日期:2012-12-07 本文受国家自然科学基金重点项目(61133005),国家自然科学基金项目(61070057,61103047),国家科技支撑计划项目(2012BAH09B02),教育部科技创新工程重大项目培育资金项目(708066),教育部博士点基金(20100161110019),湖南省杰出青年基金(12JJ1011)资助。

左定喜(1985—),男,硕士生,主要研究方向为并行计算,E-mail:zuodingxi@126.com;吴 帆(1973—),男,博士生,主要研究方向为并行计算、生物计算;李肯立(1971—),男,教授,博士生导师,主要研究方向为并行处理、网格计算、DNA 计算以及实时与混成嵌入式系统。

(5)$\beta_{jk}=(A^2p_k,Ap_j)/(Ap_j,Ap_j)$,j=k−m+1,…,k

(6)$p_{k+1}=Ap_k+\sum_{j=k-m+1}^{k}\beta_{jk}p_j$

(7)$Ap_{k+1}=A^2p_k+\sum_{j=k-m+1}^{k}\beta_{jk}Ap_j$

3 算法分析

以下讨论中,算法分析基于分布式存储并行机模型,具有 M 台机器。两台机器完成一次通信所用时间为 T(包括准备发送的时间 T_z 和传输单位字的时间 T_s)。

假设矩阵按行划分,每个浮点运算时间为 T_f。矩阵 A 的阶数 N 能被 M 整除,各处理机之间不存在负载不平衡问题。在每一个迭代步中并行矩阵向量乘的计算时间为 $(2p-1)(N/M)T_f$,其中 p 为稀疏矩阵每行非零元素的个数。在最坏情况下,矩阵向量乘需要用到全部向量,分布式并行处理机之间需要一次全收集通信,通信时间为 $\log M(T_z+N/M*T_s)$。对一般的稀疏矩阵,如带状矩阵,只需相邻处理机间通信,通信时间为 $2(T_z+p*T_s)$。通信时间与处理机台数 M 和矩阵阶数 N 无关。这种带状的稀疏矩阵具有代表性,这种情况下的矩阵向量乘的通信时间如式(1)所示:

$$T_{matrix\_vector}=2(T_z+p*T_s)+(2p-1)(N/M)T_f \quad (1)$$

关于 k 个向量内积的并行计算时间,每台处理机并行求解局部向量内积,计算时间为 $2(kN/M)T_f$。对局部内积求和并且使每台处理机都拥有这 k 个和的通信过程为全归约过程,计算时间为 $2\log M(T_z+kT_s)$。k 个向量内积的并行计算时间如式(2)所示:

$$T_{vector\_vector}(k)=2(kN/M)T_f+2\log M(T_z+kT_s) \quad (2)$$

向量校正不需要通信,k 个向量相加的计算时间如式(3)所示:

$$T_{jz}(k)=2(k-1)(N/M)*T_f \quad (3)$$

每步迭代的计算通信基本相同,取一个迭代步对算法 1 进行分析。算法 1 每次迭代需要 1 次矩阵向量乘、4 次向量校正、$2m+2$ 次向量内积计算、m 个参数 β_{jk} 和 1 个参数 α_k 校正的浮点计算,每迭代步的并行计算时间如式(4)所示:

$$\begin{aligned}T_{Orthodir}&=T_{matrix\_vector}+2*T_{jz}(2)+2*T_{jz}(m)+(2m+1)T_f+2*T_{vector\_vector}(m)+2*T_{vector\_vector}(2)\\&=(2+8\log M)T_z+(2p+4m\log M+8\log M)T_s+(4m+2p+11)N/M*T_f+(2m+1)T_f\end{aligned} \quad (4)$$

$$\begin{aligned}(q_{k+1},q_{k+1})&=((A^2p_k+\sum_{j=k-m+1}^{k}\beta_{jk}Ap_j),(A^2p_k+\sum_{j=k-m+1}^{k}\beta_{jk}Ap_j))\\&=(A^2p_k,A^2p_k)+2(A^2p_k,\sum_{j=k-m+1}^{k}\beta_{jk}Ap_j)+\sum_{j=k-m+1}^{k}\beta_{jk}^2(Ap_j,Ap_j)\\&=A(Ap_k,Ap_k)+2A(Ap_k,\sum_{j=k-m+1}^{k}\beta_{jk}p_j)+\sum_{j=k-m+1}^{k}\beta_{jk}^2(Ap_j,Ap_j)\\&=A(q_k,q_k)+\sum_{j=k-m+1}^{k}\beta_{jk}^2(q_j,q_j)\end{aligned} \quad (5)$$

由向量 p_j 的正交性质[8]有:

由式(5)看出内积(q_k,q_k)可以通过迭代的方法求出,从而可以消除算法 1 中两次向量内积计算中的数据相关性。

令 $a_k=(r_k,Ap_k)$,$b_k=(Ap_k,Ap_k)$,$q_k=Ap_k$

改进 Orthodir(m)算法如下:

算法 2 IOrthodir(m)算法

(1)$r_0=b-Ax_0$,$p_0=r_0$

对 k=0,1,2…,直到收敛计算:

(2)$\alpha_k=a_k/b_k$

(3)$x_{k+1}=x_k+\alpha_kp_k$

(4)$r_{k+1}=r_k-\alpha_kp_k$

(5)$\beta_{jk}=(Aq_k,q_j)/b_j$,j=k−m+1,…,k

(6)$a_{k+1}=(r_{k+1},Ap_{k+1})$

(7)$b_{k+1}=Ab_k+\sum_{j=k-m+1}^{k}\beta_{jk}^2b_j$

(8)$p_{k+1}=q_k+\sum_{j=k-m+1}^{k}\beta_{jk}p_j$

(9)$q_{k+1}=Aq_k+\sum_{j=k-m+1}^{k}\beta_{jk}q_j$

对于 IOrthodir(m)算法中的每次迭代,需要 1 次矩阵向量乘、5 次向量校正计算、$m+1$ 次向量内积计算、m 个参数 β_{jk} 和 1 个 α_k 的浮点计算。

IOrthodir(m)算法每迭代步的计算时间如式(6)所示:

$$\begin{aligned}T_{IOrthodir}&=T_{matrix\_vector}+2*T_{jz}(2)+3*T_{jz}(m)+T_{vector\_vector}(m)+T_{vector\_vector}(2)+(3m+1)T_f\\&=(2+4\log M)T_z+(2p+2m\log M+4\log M)T_s+(2p+4m+3)N/M*T_f+(3m+1)T_f\end{aligned} \quad (6)$$

比较式(4)、式(6),当处理器数 M 满足式(7)时,改进算法比原算法的计算时间要少。

$$M\log M>(M*m-8N)T_f/(4T_z),T_z>>T_s>T_f \quad (7)$$

对 $T_{Orthodir}$ 的计算公式,求 $\partial T_{Orthodir}/\partial M=0$ 可以得到 Orthodir(m)算法计算时间最小时的处理器数目。

$$M_{Orthodir(m)}=\frac{(4m+2p+11)NT_f}{4(2T_z+mT_s+2T_s)} \quad (8)$$

由推导过程可以看出,IOrthodir(m)算法和 Orthodir(m)算法在本质上是等价的,两个算法仅仅是次序不同,数值计算结果也应该一样。通过给出的数值实验结果来看,收敛的速度相同。

对 $T_{IOrthodir}$ 的计算公式,求 $\partial T_{IOrthodir}/\partial M=0$ 可以得到 IOrthodir(m)算法计算时间最小时的处理器数目。

$$M_{IOrthodir(m)}=\frac{(4m+2p+3)NT_f}{4T_z+2mT_s+4T_s} \quad (9)$$

当 $T_z>>T_s$ 时,$M_{IOrthodir(m)}\approx 2M_{Orthodir(m)}$,说明 IOrthodir($m$)算法的扩展性要比 Orthodir($m$)算法的好。

对比 Orthodir(m)算法,IOrthodir(m)算法的性能提高比率如式(10)所示:

$$\eta=\frac{T_{Orthodir}-T_{IOrthodir}}{T_{Orthodir}}\overset{T_z\geqslant T_s}{\approx}\frac{4\log MT_z+8N/M*T_f-mT_f}{(2+8\log M)T_z+(4m+2p+11)N/M*T_f+(2m+1)T_f} \quad (10)$$

当处理器数目 M 趋近于正无穷时,可以计算得到 $\eta\to 50\%$。因此,对比 Orthodir(m)算法,IOrthodir(m)算法的计算性能可以提高 50%。

4 实验

本文测试环境为 16 台曙光天阔 A440-G 服务器集群,操作系统为 SUSE,互联网络为千兆以太网,并行环境为 MPI 并行。分别对阶数为 2000,3000 的两个非对称稀疏矩阵求解,

(下转第 158 页)

算法的主要特点是网络内节点无需关注当前网络的拓扑结构。在定位过程中，节点之间仅交换定位估计值，就可以有效降低网络通信量。由于实际中 RSSI 值极易受到外部环境干扰，甚至同一节点在不同时刻的 RSSI 值都会发生变化，因此，衰弱环境下移动无线传感器网络的定位算法优化问题是我们下一步的研究方向。

参 考 文 献

[1] 王福豹，史龙，任非原. 无线传感器网络中的自身定位系统和算法[J]. 软件学报，2005，16(5)：857-868

[2] 陈娟，李长庚，宁新鲜. 基于移动信标的无线传感器网络节点定位[J]. 传感技术学报，2009，22(1)：121-125

[3] 嵇玮玮，刘中. DV-Hop 定位算法在随机传感器网络中的应用研究[J]. 电子与信息学报，2008，30(4)：970-974

[4] 曾凡仔，孙正章，罗娟，等. 无线传感器网络的节点定位方法[J]. 通信学报，2008，29(11)：62-66

[5] Reza O-S. Distributed kalman filter with embedded consensus filters[C]// Proceedings of the 44th IEEE Conference on Decision and Control, and European Control Conference. Seville: Eduardo F. Camacho, 2005: 8179-8184

[6] Carlir, Chusoa, Schenato L, et al. Distributed kalman filtering based on consensus strategies[J]. IEEE Journal on Selected Areas in Communications, 2008, 26(4): 622-633

[7] Msechu E J, Roumeliotis S I, Ribeiro A, et al. Decentralized quantized kalman filtering with scalable communication cost[J]. IEEE Transactions on Signal Processing, 2008, 56(8): 3727-3741

[8] Khan U A. Distributed sensor localization in random environments using minimal number of anchor nodes [J]. IEEE Transactions on Signal Processing, 2009, 57(5): 2000-2016

[9] Kar S, Moura J M F, Scaglione A. Gossip algorithms for distributed signal processing[J]. Proceedings of the IEEE, 2010, 98(1): 1847-1864

[10] 王林，张国忠，朱华勇，等. 面向移动传感器网络的自适应一致性融合估计方法[J]. 上海交通大学学报，2011，45(3)：383-392

(上接第 127 页)

比较两个算法的性能，结果如表 1、表 2 所列。收敛判据为相对残差小于 10^{-4}。从两组实验结果来看，由于受并行机速度和网络结构以及求解问题的性质等影响，当处理器数目为 1 时，Orthodir(m)算法优于 IOrthodir(m)算法。随着处理器数目的增大，IOrthodir(m)算法的性能要渐渐优于 Orthodir(m)算法。

表 1　2000 阶矩阵(单位：秒)

处理器数目	Orthodir(m)	IOrthodir(m)	η(%)
1	99.6	102.3	−2.7
2	65.5	65.1	0.61
4	45.3	44.7	1.32
8	31.2	29.8	3.35
16	20.7	18.9	8.69

表 2　3000 阶矩阵(单位：秒)

处理器数目	Orthodir(m)	IOrthodir(m)	η(%)
1	151.2	155.1	−3.9
2	110.3	110.2	0.0
4	80.5	77.5	3.72
8	61.1	56.3	7.85
16	49.7	44.1	11.26

结束语　给出的改进 IOrthodir(m)算法具有与原算法相同的收敛速度，没有增加计算量。从理论上证明了该算法具有比原算法更好的扩展性，算法性能更优，最高时性能提高比例能达到 50%。从实验结果数据来看，随着处理器数目的增加，IOrthodir(m)算法的性能优于 Orthodir(m)算法，扩展性方面也更好，更适合于并行计算。理论和实验都证明了 IOrthodir(m)算法是一种更优、扩展性更好的算法。

由于实验室条件有限，下一步打算在更大规模机群上进行实验，验证理论结果。参考文献[1]中的方法，从充分利用局存中数据、数据分配时的负载平衡、通信与计算重叠等方面改进该算法，提高算法效率。

参 考 文 献

[1] 李晓梅，吴建平. Krylov 子空间方法及其并行计算[J]. 计算机科学，2005，32(1)：19-21

[2] 张理涛，黄廷祝，谷同祥，等. 一种适合于分布式并行计算改进的平方共轭残差法[J]. 微电子学与计算机，2008，25(10)：16-20

[3] 刘杰，迟利华，胡庆丰，等. 一种改进的适合并行计算的 TFQMR 算法[J]. 计算机研究与发展，2005，42(7)：1235-1241

[4] Dazevedo E, Eijkhout V, Romaine C. Reducing communication costs in the conjugate gradient algorithm on distributed memory multiprocessors[J]. University of Tennessee, Knoxville: LAPACK Working Note, 1992, 56: 200-207

[5] Merurant G. Multitasking the conjugate gradient on the Cray X—MP/48[J]. Parallel Computing, 1987, 5(2): 267-280

[6] Saad Y. Krylov subspace methods on supercomputers. SIAM Journal on Scientific and Statistical Computing, 1989, 10(8): 1200-1232

[7] 刘杰，刘兴平，迟利华，等. 一种改进的适合并行计算的共轭剩余算法[J]. 计算机学报，2006，29(3)：14-17

[8] Young D M, Jea K C. Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods: Part II[C]// the nonsymmetrizable case, CNA-163, Center for Numerical Analysis, Univ. of Texas. Austin, 1980: 809-910

[9] 张文生. 科学计算中的偏微分方程有限差分法[M]. 北京：高等教育出版社，2006：208-210

[10] Yang L T, Brent R P. Quantitative performance analysis of the improved quasi-minimal residual method on massively distributed memory computers Advances in Engineering Software[J]. 2002, 33(1): 169-177

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

[12] Saad Y. Iterative Methods for Sparse Linear Systems[M]. Philadelphia, USA: Society for Industrial and Applied Mathematics, 2003: 143-200

[13] 莫则尧. 大型科学与工程应用程序并行化关键技术及其应用研究[D]. 北京：北京应用物理与计算数学研究所计算物理实验室，1999

[14] 迟利华. 大型稀疏线性方程组的并行计算[D]. 长沙：国防科技大学，1998

[15] 谷同祥. 大型稀疏线性方程组的并行非定常迭代方法[D]. 北京：北京应用物理与计算数学研究所计算物理实验室，2001