第32卷 第19期 计 算 机 工 程 2006年10月
Vol.32 № 19 Computer Engineering October 2006
·博士论文· 文章编号：1000—3428(2006)19—0003—03 文献标识码：A 中图分类号：TP302

SMP集群系统上可扩展并行特征问题求解器研究

赵永华[1,2,3]，迟学斌[1]，姜金荣[1]
(1. 中国科学院计算机网络信息中心超级计算中心，北京 100080；2. 中国科学院软件所，北京 100080；
3. 德州学院计算机科学系，德州 253000)

摘 要：基于对称三对角特征问题的分而治之方法，提出了一个适合SMP集群环境的多级混合并行算法。SMP节点内的并行求解采用了粗粒度和细粒度两种OpenMP并行。为了改善纯MPI算法中的负载不平衡，混合并行算法使用了动态任务分配方法。在深腾6800上的试验表明，混合并行算法具有好的扩展性和加速比。

关键词：SMP集群；MPI+OpenMP；混合并行；并行求解器

Study on Scalable Parallel Eigenproblem Solver on SMP Cluster

ZHAO Yonghua[1,2,3], CHI Xuebin[1], JIANG Jinrong[1]
(1. Supercomputing Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing 100080; 2. Software Institute of Chinese Academy of Sciences, Beijing 100080; 3. Department of Computer Science, Dezhou University, Dezhou 253000)

【Abstract】 Based on divide-and-conquer method of eigenvalues of symmetric tridiagonal matrices, a multilevel hybrid parallel algorithm that is suitable to cluster of SMP's environments is presented. Parallel solution within the SMP node takes both coarse-grain and fine-grain OpenMP parallel. In order to improve load balance of pure MPI algorithm, a dynamic work sharing is used in hybrid parallel algorithm. An experiment analysis on the DeepComp 6800 shows the hybrid algorithm performs good scalability and speedup.

【Key words】 SMP cluster; MPI+OpenMP; Hybrid parallel; Parallel solver

对称三对角特征问题是对称矩阵特征问题的核心。解决该问题最有效的并行方法是分而治之(D&C)方法。许多研究人员针对共享存储多处理机、超立方MPC并行机、工作站集群等提出了不同的算法和实现方法[1,2]。

当前，大部分高性能计算机是SMP(Symmetric Multiprocessor)集群系统。它们是由拥有多个处理器的SMP节点和连接各节点的快速网络构成的多级体系结构。由于具有共享存储和分布存储两级存储结构，节点内使用共享内存并行、节点间通过消息传递的混合并行(MPI+OpenMP或MP+Pthreads)模型潜在地能更有效地利用SMP集群的特性。而这种混合并行模型也已应用到coastal wave分析、大气研究、分子动态分析等许多领域[3~5]。

基于秩-2修正+Laguerre迭代[2]的D&C算法，本文提出了SMP集群系统下的多级混合并行算法。该算法使用了粗粒度和细粒度两种并行模型，这降低了算法中的通信开销。另外，通过节点内的动态工作分配改善了纯MPI并行算法的负载平衡。

1 秩-2修正的分而治之算法结构

设三对角对称矩阵 T 为

$$\boldsymbol{T}=\begin{pmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & b_2 & \ddots & \ddots & \\ & & \ddots & a_{n-1} & b_{n-1} \\ & & & b_{n-1} & a_n \end{pmatrix}$$

为了求得 $f(\lambda)\equiv\det[T-\lambda I]$ 的部分或全部 0(即特征值)，引入 T 的分裂矩阵 $\hat{T}=\begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix}$，则 T 可表示为 $\hat{T}$ 的秩-2修正形式

$$\boldsymbol{T}=\hat{T}+\begin{pmatrix} & \vdots & \vdots & \\ \cdots & 0 & b_k & \cdots \\ \cdots & b_k & 0 & \cdots \\ & \vdots & \vdots & \end{pmatrix}$$

定理[1] 设 $\lambda_1<\lambda_2<\cdots<\lambda_n$ 和 $\hat{\lambda}_1<\hat{\lambda}_2<\ldots<\hat{\lambda}_n$ 分别是 $\hat{T}$ 和T的特征值，那么

$\hat{\lambda}_1\in(\lambda_1,\lambda_2)$，$\hat{\lambda}_n\in(\lambda_{n-1},\lambda_n)$，$\hat{\lambda}_i\in(\lambda_{i-1},\lambda_{i+1})$，$2\le i\le n$-1；

$\lambda_1\in(\hat{\lambda}_1-|b_k|,\hat{\lambda}_1)$，$\lambda_n\in(\hat{\lambda}_n,\hat{\lambda}_n+|b_k|)$，$\lambda_i\in(\hat{\lambda}_{i-1},\hat{\lambda}_{i+1})$，$2\le i\le n$-1

引理 对每个 i =1, …, n，在开区间 $(\hat{\lambda}_i,\lambda_i)$ (或 $(\lambda_i,\hat{\lambda}_i)$) 内不存在 λ_j 或 $\hat{\lambda}_j$ (j =1, …, n)。即假若 L_i 是具有端点 $\hat{\lambda}_i$ 和 λ_i 的开区间，那么 $\{L_i\}_{i=1}^{n}$ 是不相交区间的聚集。

按照上面的引理，假如已得到T_0和T_1的全部特征值，则需对这些特征值进行合并和升序排序，得到有序序列 $\hat{\lambda}_1,\hat{\lambda}_2,\ldots,\hat{\lambda}_n$。然后以 $\hat{\lambda}_i$ (i=1,…,n)为起点，可通过Laguerre[1]迭代求得T 的特征值 λ_i。而Laguerre是一个3阶收敛的迭代方法，具有比二分法或Newton法更快的收敛速度。

基金项目：国家“863”计划超级计算网格节点建设基金资助项目(2002AA104540)；中国科学院知识创新工程信息化建设专项“超级计算环境建设与应用”(INF05-SCE)

作者简介：赵永华(1966－)，男，副教授、博士生，主研方向：高性能计算方法与软件，集群计算和并行算法；迟学斌，研究员、博导；姜金荣，博士生

收稿日期：2005-10-11 **E-mail：**yhzhao@sccas.cn

为了得到T_0和T_1的特征值，分裂过程可递归进行下去，直到得到可使用QR等串行算法快速求解的小规模问题。图 1 给出了分而治之算法的任务树。

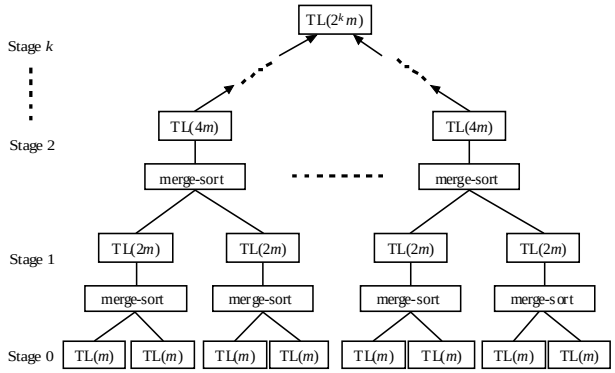

图 1 D&C 算法任务树

2 MPI 并行算法

尽管分而治之算法逻辑上可看作树，但树中的节点不能表示并行实现中的处理器，否则将严重降低算法的并行性。设计并行算法的一个重要目标是使每个进程的计算时间最大化。根据这一原则，下面给出了一个有效的 MPI 并行算法。

不失一般性，设三对角矩阵T的阶$n=2^m$，处理器数量$p=2^k$，$s=m-k$。MPI并行算法描述如下：

置 $i=1$，然后执行下列操作。

第 1 步：将n阶三对角矩阵T分解成p个阶为 2^s的子矩阵$T_0, T_1,\ldots,T_{p-1}$，并分布到处理器$P_0, P_1,\ldots,P_{p-1}$。每个处理器独立地计算子矩阵特征值。

第 2 步：逻辑上将处理器按$\left[P_0,\ldots,P_{2^i-1}\right]$，$\left[P_{2^i},\ldots,P_{2^{2i}-1}\right]$，…，$\left[P_{2^{k-1}},\ldots,P_{2^k-1}\right]$分成 2^{k-i}组，每组包含 2^i个处理器。(1) 各组处理器将最近计算的特征值传递给本组最左边的处理器(接收器)。(2)该接收器对接收到的 2^{i+s} 个特征值合并、排序。(3) 接收器将排序后的特征值分发回本组内的所有进程。

第 3 步：(1)每个处理器得到本组处理器需要处理的 2^{i+s}阶子矩阵。(2)以第 2 步接收到的 2^s个特征值为初始点，使用Laguerre迭代近似计算 2^{i+s}阶子矩阵的 2^s个特征值。(3) 置$i=i+1$，若$i=k$结束，否则转第 2 步。

上面算法第 2 步完成 MPI 进程间的消息传递，这一过程在各进程组间并行进行。但随着 i 的增加，每组中进程数成倍增加，这使通信代价逐次变大。这样，当使用的处理器数很大时，将引起较大的通信开销。第 3 步各进程并行求解相同数量的近似特征值，但由于求解不同近似特征值时迭代的次数可能不同，这使该算法潜在地存在着负载不平衡问题。而上述问题是 MPI 并行算法所固有的，将在混合并行算法中得到改善。

3 在 SMP 集群上的混合并行算法

给定一个有效的消息传递算法，可使用增量过程设计一个有效的混合并行算法。

3.1 混合多级并行算法结构

由于 SMP 集群本身具有集群系统的分布存储和 SMP 系统的共享存储两级特性，与此对应将混合并行算法的设计分解为机群级算法和节点级算法。在机群级算法设计时，SMP 集群系统被作为一个由节点构成的分布存储系统，算法采用分布消息传递并行。但和纯 MPI 算法中每个处理器运行一个进程不同，在机群级算法中每个进程映射为一个节点。节点级算法也称为 SMP 算法，将集群级算法中分配给每个节点的计算任务映射成一个有效的节点内多线程并行算法。本文的节点级算法基于 OpenMP 的多线程并行。

最简单的 OpenMP 并行是使用 OpenMP 工作共享结构的循环-嵌套并行，每个线程独立地执行它的循环部分。这种并行又称为细粒度 OpenMP 并行。

另一种 OpenMP 并行称为粗粒度 OpenMP 并行。在粗粒度 OpenMP 并行中，OpenMP 采用 SPMD 并行编程模式，每个线程本身类似一个 MPI 进程。线程功能的说明依赖库函数 omp_get_num_threads( )和 omp_get_thread_num( )，它们分别返回并行域内线程数量和执行线程的序号 *id*(编号为 0 的线程为主线程)。其后根据它们的值对线程功能编码。不同的线程可执行相同或不同的操作，这样实现了数据和任务的并行。

当将上面两种 OpenMP 并行同 MPI 结合在一起时，可得到 MPI+细粒度 OpenMP 和 MPI+粗粒度 OpenMP 两种混合并行方式。

3.2 MPI+OpenMP 混合并行算法

考虑n阶三对角对称矩阵T在np个r-路SMP节点上的特征问题，不妨设$n=2^m$，$np=2^d$，$r=2^k$，所对应的处理器数量$s=2^p=2^{d+k}$。

首先将上面的MPI并行算法直接转换为一个集群级算法。这时矩阵按节点个数分解成np个阶为 2^{m-d}的子矩阵并分布在各节点上。相应地，循环变量i对应的循环次数变为d。这里将集群算法分为：Cphase1，Cphase2 ，Cphase3，分别对应MPI算法中的第 1 步、第 2 步和第 3 步。图 2 给出了集群级算法结构。

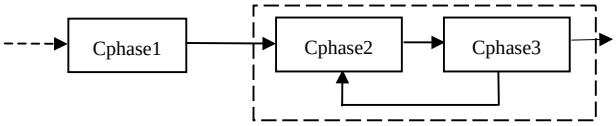

图 2 集群级算法结构

为了得到混合并行算法，须将集群算法中的 Cphase1 和 Cphase2 映射为节点内的 OpenMP 多线程并行。

(1)映射 Cphase1 为 OpenMP 多线程并行

Cphase1 完成节点内 2^{m-d}阶子矩阵的特征值求解。这类似于MPI并行，需构造一个OpenMP粗粒度并行算法。将Cphase1 在节点内的线程并行对应的各操作步表示为Nphase1、Nphase2 和Nphase3。对Nphase1 而言，由于节点内所有线程共享内存，因此矩阵在线程间分解时，需计算每个线程的数组范围，并得到数组区域到线程的映射。而Nphase2 是通过数据复制完成特征值的合并-排序和在线程间的散播，从而避免了通信开销。在Nphase3 中每个线程通过Laguerre迭代完成近似特征值的计算。

(2)映射 Cphase3 为一个多线程的 OpenMP 并行

Cphase3 由节点内进程通过Laguerre迭代计算 2^{m-d}个近似特征值。这对应着一个循环结构，可使用OpenMP工作共享制导由多个线程通过细粒度并行完成。在缺省调度(static)方式下，OpenMP制导把循环均匀地分配给节点内所有线程(即节点内的处理器)。但由于Laguerre迭代在计算不同特征值的近似值时，达到指定精度所执行的迭代次数可能不同，这样上述的均匀分配将导致线程间的负载不平衡。为了改善负载平衡问题，同时减少由同步引起的开销，我们将循环在线程间的分配采用了schedule(dynamic, *)动态调度方式。类似地，使用Nphase表示Cphase3 的OpenMP并行。

在 Cphase2 中，由每个进程中的主线程完成特征值的接收、发送以及特征值的合并和排序。这一过程发生在 OpenMP 并行域外。

图 3 给出了完整的混合并行算法结构。

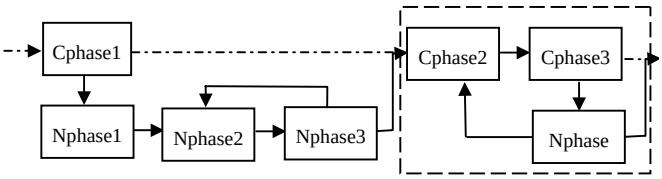

图 3 混合并行算法结构

3.3 通信开销分析

在三对角矩阵特征问题的D&C并行算法中，外循环i的循环次数等于进程数P的对数$\log_2 P$。对于含有r-路SMP节点的集群系统，混合并行算法的进程数等于节点个数np。而MPI并行算法的进程数等于处理器数，即$np*r$。这样混合并行算法的i迭代次数仅为MPI算法迭代次数的$\frac{1}{\log_2^r}$。这极大地减少了由Cphase2 导致的进程间通信次数，降低了通信开销。

4 性能测试

我们通过深腾 6800 系统对混合并行算法和 MPI 算法进行了性能测试和比较。

4.1 深腾 6800 系统

深腾 6800 系统整体为 5 万亿次面向网格的超级计算机系统，包括 256 个四路节点。每个结点配置 4 颗 1.3GHz Itanium2 CPU，系统峰值速度为 5.3T Flop/s，内存总容量为 2.6TB。高速连接网络为 Qsnet，点到点通信带宽大于 300MB/s，延迟时间小于 7μs。深腾 6800 软件环境包括 Redhat Linux Advanced Server 2.1 操作系统，支持 OpenMP 制导的 Intel Fortran Compiler7.0，C/C++编译器，以及 Intel MPICH1.2.4 并行环境。

4.2 MPI 版本和 MPI+OpenMP 版本性能比较

这里仅给出对固定问题规模使用不同处理器在两种版本上的测试结果。测试矩阵为 Toeplitz 矩阵[1,4,1]，矩阵规模是 40 000*40 000。表 1 为纯 MPI 和混合 MPI+OpenMP 并行求解的运行时间。对于混合方式，分别测试了 2 线程和 4 线程的运行情况。而对于 4 个线程，分别给出了动态和静态两种调用方式下的运行时间。表 1 表明，在 4 种情况下，混合模式的并行性能都好于纯 MPI 并行。而在混合模式下，使用 4 线程的并行性能好于 2 线程。在 4 线程情况下，采用动态调度方法的并行性能又好于静态调度方法。并且它们之间的运行时间的差距随着处理器的增加变大。测试结果表明动态调度的多级混合并行算法具有更好的性能。

表 2 是对应的加速比。表中数值表明混合并行算法加速比明显好于 MPI 并行算法加速比。而对于混合并行算法，采用动态调用比静态调用具有更好的加速比。这说明使用动态调用的混合并行算法具有更好的可扩展性。

表 1 纯 MPI 和 MPI+OpenMP 并行执行时间

节点数 (4-路节点)	运行时间 (s)			
	纯 MPI	MPI+OpenMP (2 线程)	MPI+OpenMP 静态调度 (4 线程)	MPI+OpenMP 动态调度 (4 线程)
1	17.78	17.00	16.27	15.05
2	10.52	9.59	8.955	8.14
4	5.70	4.89	4.50	3.98
8	2.99	2.46	2.08	1.80
16	1.55	1.29	1.13	0.94
32	0.83	0.64	0.53	0.46

表 2 纯 MPI 和 MPI+OpenMP 并行加速比

节点数 (4-路节点)	加速比			
	纯 MPI	MPI+OpenMP (2 线程)	MPI+OpenMP 静态调度 (4 线程)	MPI+OpenMP 动态调度 (4 线程)
1	1	1	1	1
2	1.69	1.72	1.72	1.97
4	3.12	3.15	3.34	3.52
8	5.95	5.99	6.79	7.37
16	11.45	11.55	13.94	14.70
32	21.53	21.89	28.97	31.58

5 总结

本文建立了一个求解对称三对角特征问题的多粒度混合并行算法，在 SMP 集群系统上得到了理想的并行性能。通过节点内使用共享内存并行模型，减少了通信开销。使用动态调度方式改善了负载平衡。实验表明混合并行算法比纯 MPI 算法具有更好的可扩展性。

参考文献

1 Li T Y, Zeng Z. The Laguerre Iteration in Solving the Symmetric Tridiagonal Eigenproblem[J]. SIAM J. Science and Statistical Comput., 1994, 15(5): 1145-1173.

2 Treffz C, Huanf C C. A Scalable Eigenvalue Solver for Symmetric Tridiagonal Matrices[J]. Parallel Computing, 1995, 21(8): 1213-1240.

3 Henty D S. Performance of Hybrid Message Passing and Shared Memory Parallels for Discrete Element Modeling[C]. Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, Dallas, Texas, United States, 2000.

4 Bova S W, Breshears C, Cuicchi C, et al. Dual-level Parallel Analysis of Harbor Wave Response Using MPI and OpenMP[J]. Int. J. High Perform Comput. Appl., 2000, 14 (3): 49-64.

5 Rabense R, Wellein G. Communication and Optimization Aspects of Parallel Programming Models on Hybrid Architecture[J]. International Journal of High Performance Computing Applications, 2003, 17(1): 49-62.

（上接第 2 页）

有所加重，但加重的部分完全可以通过预处理的办法一次性消除，总之，ESCS 方案所取得的优势是很明显的。

参考文献

1 Pointcheval D, Stem J. Security Arguments for Digital Signatures and Blind Signatures[J]. Journal of Cryptology, 2000,13(3): 361-396.

2 Adams C, Llyod S. Understanding Public Key Infrastructure[M]. New Riders Publishing, 1999.

3 Lee B, Kim K. Self-certified Signature[C]. Proc. of Progress in Cryptology: INDOCRYPT'02, Lncs 2551. Springer-Verlag, 2002: 199-214.

4 Girault M. Self-certified Public Keys[C]. Proc. of Advances in Cryptology: Eurocrypt'91, LNCS 547. Springer-Verlag, 1991: 490.

5 Lee B, Kim K. Self-certificate: PKI Using Self-certified Key[C]. Proc. of Conference on Information Security and Cryptology, 2000: 65-73.

6 Chadwick D. An X.509 Role-based Privilege Management Infrastructure[Z]. Business Briefing: Global Infosecurity, 2002.

7 Schnorr C P. Efficient Signatures Generation by Smart Cord[J]. Journal of Cryptology, 1991, 4(3): 161-174.

8 杨义先, 孙 伟, 钮心忻. 现代密码新理论(第 1 版)(M). 北京: 科学出版社, 2002-08.

9 杨义先, 钮心忻. 网络安全理论与技术(第 1 版)(M). 北京: 人民邮电出版社, 2003-10.