计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(9):246−250 [doi: 10.15888/j.cnki.csa.007078]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于分治法求解对称三对角矩阵特征问题的混合并行实现①

朱京乔[1,2], 赵永华[1]

[1](中国科学院 计算机网络信息中心, 北京 100190)
[2](中国科学院大学, 北京 100049)
通讯作者: 赵永华, E-mail: yhzhao@sccas.cn

摘　要: 基于对称三对角矩阵特征求解的分而治之方法, 提出了一种改进的使用 MPI/Cilk 模型求解的混合并行实现, 结合节点间数据并行和节点内多任务并行, 实现了对分治算法中分治阶段和合并阶段的多任务划分和动态调度. 节点内利用 Cilk 任务并行模型解决了线程级并行的数据依赖和饥饿等待等问题, 提高了并行性; 节点间通过改进合并过程中的通信流程, 使组内进程间只进行互补的数据交换, 降低了通信开销. 数值实验体现了该混合并行算法在计算效率和扩展性方面的优势.

关键词: 并行计算; 对称特征问题; 分治算法; Cilk

引用格式: 朱京乔,赵永华.基于分治法求解对称三对角矩阵特征问题的混合并行实现.计算机系统应用,2019,28(9):246–250. http://www.c-s-a.org.cn/1003-3254/7078.html

Hybrid Parallel Algorithm Using MPI/Cilk for Symmetric Tridiagonal Eigenproblems

ZHU Jing-Qiao[1,2], ZHAO Yong-Hua[1]

[1](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Divide and conquer algorithm is widely used for tridiagonal matrix eigenproblems while computing efficiency and storage limitation are always bottlenecks for large scale problems. In this study, the proposed eigenproblem algorithm based on hybrid parallel paradigm with MPI/Cilk optimizes the divide and conquer algorithm both at data and task levels. The introduced task-based parallelization mechanism inside computing nodes solves the problem in data dependence and thread starvation by directed acyclic graph model. By coarse-grained partition of tasks the overhead of data communication among MPI nodes is also optimized, which helps to improve load balance. The numerical test is carried out and the result is compared with the pure MPI and MPI/openMP parallel algorithm, which shows the performance and efficiency of the algorithm.

Key words: parallel computing; symmetric eigenproblem; DC method; Cilk

对称矩阵的特征求解问题在科学计算领域普遍存在, 包括计算物理、计算化学、结构力学、纳米材料等前沿学科. 通常的处理方法是先通过正交变换把原矩阵化为三对角矩阵, 如使用 Householder 或 Givens 方法, 再求解三对角阵的特征值和特征向量, 然后映射回原矩阵的特征值和特征向量. 三对角阵的特征求解

① 基金项目: 国家重点研发计划 (2017YFB0202202, 2016YFB0201302); 中国科学院“十三五”信息化建设专项 (XXH13506-405)
Foundation item: National Key Research and Development Program of China (2017YFB0202202, 2016YFB0201302); CAS Special Fund for Informatization Construction in 13th Five-Year Plan (XXH13506-405)
收稿时间: 2019-03-05; 修改时间: 2019-04-02; 采用时间: 2019-04-10; csa 在线出版时间: 2019-09-05

算法有很多种, 常见的有 QR 法、二分法、MRRR 方法和分而治之方法等等. 由于 QR 法和二分法求解特征向量时往往需要显式的正交过程, 时间复杂度为 $O(n^3)$, 常用于中小规模矩阵的特征求解. 而分治法和 MRRR 法能避免这一过程, 分治法的平均时间复杂度仅为 $O(n^{2.3})$, 其分而治之思想对于大规模矩阵特征问题的并行求解求解有着天然的优势.

分而治之求解对称三对角矩阵特征问题的算法最早由 Cuppen 提出[1], 后 M. Gu 和 S. C. Eisenstat 等人作出改进, 提高了该算法求解的特征向量的正交性[2], 使得该算法能够在实际求解中应用. 该算法采用分而治之思想, 将原矩阵划分为若干子矩阵, 先求出子矩阵的特征分解, 再通过修正计算, 逐级合并子矩阵的结果, 回代得到原矩阵的特征分解. 分治法具有很强的灵活性, 适合大规模三对角矩阵特征问题求解的并行化求解.

分治算法的并行化工作很早就开始展开[3–5], 如基于分布式存储的 MPI 进程级并行, 基于共享内存的 openMP 线程级并行, 还有基于 SMP 的 MPI/openMP 混合并行等模型. 并行化的实现主要基于数据并行的考量, 通过划分数据到多个核或节点上计算, 在负载均衡的条件下能取得较好的并行效果. 但对于分治和递归类的场景, 当数据依赖呈现复杂的网状结构时, 难以实现理想的负载均衡效果.

基于任务的并行编程模型按照功能将一个应用划分成多个能并行执行的模块, 可以是细粒度的, 也可以是粗粒度的. 常见的任务并行模型有 TBB,Cilk 和 SMPSS 等, 使用这些模型时无需关心任务的具体分配和调度细节.

Cilk 作为一个基于任务的并行编程模型, 为共享内存系统提供了一个高效的任务窃取调度器, 适合用来为分治和递归类算法实现高效的多核任务并行. 对于并行编程, Cilk 为串行程序的并行化提供了向量化、锁、视图等机制[6], 通过关键字来声明操作, 改造后的代码具有语义化、可读性强的特点.

1 分治算法求解的基本原理

1.1 矩阵划分

对任意 n 阶实对称三对角矩阵 T, 基于秩 1 修正从第 k 行作如下划分:

$$T=\begin{pmatrix} T_1 & \beta e_k e_1^{\mathrm{T}} \\ \beta e_1 e_k^{\mathrm{T}} & T_2 \end{pmatrix} =\begin{pmatrix} T'_1 & 0 \\ 0 & T'_2 \end{pmatrix}+\beta\begin{pmatrix} e_k \\ e_1 \end{pmatrix}\begin{pmatrix} e_k^{\mathrm{T}} & e_1^{\mathrm{T}} \end{pmatrix} \tag{1}$$

其中, T'_1, T'_2 是三对角子矩阵块, e_k 代表第 k 个元素为 1 的单位向量, 元素 β 是 T 的第 k 个副对角元素. 为使修正矩阵元素一致, 对 T_1 的末位元素和 T_2 的首元素都减去了相同的元素.

分治过程一般采用折半划分, 对于划分后的矩阵 T', 若规模未达到指定阈值, 则继续划分.

1.2 分治法求解过程

若划分后的子矩阵规模达到设定阈值, 调用 QR 或者其他算法得到子矩阵 T'_1, T'_2 的特征值分解:

$$T'_1=Q_1D_1Q_1^{\mathrm{T}}, T'_2=Q_2D_2Q_2^{\mathrm{T}} \tag{2}$$

令修正向量

$$z=diag(Q_1,Q_2)^{\mathrm{T}}\begin{pmatrix} e_k \\ e_1 \end{pmatrix}, \tag{3}$$

则矩阵 T 的特征分解如下:

$$T=\begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix}\left[\begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix}+\beta zz^{\mathrm{T}}\right]\begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix}^{\mathrm{T}} =Q\left(D+\beta zz^{\mathrm{T}}\right)Q^{\mathrm{T}}. \tag{4}$$

令 $D+\beta zz^{\mathrm{T}}=W$, 因为 $diag(Q_1, Q_2)$ 为正交矩阵, 所以 T 和 W 相似有相同的特征值, 求 T 的特征分解 $T=Q\Lambda Q^{\mathrm{T}}$, 可以通过先求出 W 的特征分解: $W=P\Lambda P^{\mathrm{T}}$, 再令 $Q=diag(Q_1, Q_2)P$, 即可求得 T 的特征分解.

对于 W 的特征分解, 需要根据划分后矩阵的特征值和原矩阵特征值之间的间隔性[7], 通过求解对应的 secular 方程 (3):

$$1+\beta\sum_i \frac{x_i^2}{d_i-\lambda}=0, d_i=D_{ii} \tag{5}$$

得到相应的的特征值 λ. 文献[6]讨论了方程 (3) 根的分布特性和解法, 在特征区间内使用牛顿法或拉格朗日法迭代逼近区间内的特征值, 线性时间内即可收敛. 解出所有的特征值后, 再通过式 $(D-\lambda I)^{-1}x$[1]求得每个特征值对应的特征向量, 回代到式 (1) 中得到 T 的特征分解.

整个求解过程在逻辑上可看作树型结构, 在叶子结点上进行特征求解, 在非叶子结点上进行特征合并.

1.3 注意事项

文献[5]中指出合并过程中如果 D 的主对角线上出现了相同的元素或者 z 向量中出现 0 元素时, 在迭代前进行 deflation 操作可以避免特征区间内迭代不收敛的情况. 通过 Givens 正交旋转变换, 部分特征值和特征向量就能原地得到. 如果求得的特征值前后过于接近, 按原方法求得的特征向量会丢失正交性, 文献[8,9]讨论了特征值间隔过小时的特征向量的求法, 避免了显式正交过程. 文献[2]根据特征问题的反问题给出了特征向量的改进求法, 保证了计算结果的正交性.

2 分治算法的并行化

2.1 节点内并行实现

单节点内求解三对角矩阵块的特征分解主要包含两个过程, 分别是求出最小划分矩阵的特征分解以及逐步合并特征值和特征向量. 由于每个子矩阵的特征计算以及同级子矩阵之间的合并过程是相互独立的, 下面结合 Cilk 模型给出递归实现的分治算法在单节点多核环境下的并行化算法.

算法 1. 分治算法基于 Cilk 的节点内并行

1) 判断输入矩阵的规模, 如果矩阵规模小于等于设定 阈值, 执行 2); 否则执行 3);
2) 调用特征求解算法进行特征分解, 返回结果;
3) 划分矩阵 T 得子矩阵 T_1, T_2, 通过 Cilk 执行任务划分, 并行地对 T_1, T_2 递归调用步骤 1), 求出子矩阵的特征值和特征向量后, 进行排序和迭代逼近等操作合并子矩阵的特征值和特征向量, 得到 T 的特征分解, 返回结果.

Cilk 在程序递归执行过程中不断划分新的任务, 产生 Cilk 线程去处理, 并把其分配到空闲的核上, 和父线程并行执行. 一个 Cilk 任务是在同步、生成新线程或返回结果之前执行的最长序列化指令集或代码段[10]. 求解过程中, 递归调用 T_1 之前的过程可看作任务 A, 递归调用 T_2 的过程可看作任务 B, 同步及合并过程可看作任务 C. 现假设程序只进行了四次嵌套调用, 则全局任务划分后生成的任务流如图 1 所示.

2.2 多节点混合并行实现

实现分治算法的多节点并行, 需要先把原始矩阵按逻辑树结构划分成小的子矩阵分发到叶子结点上进行初步计算, 然后通过 MPI 通信汇总每棵子树的计算结果, 把整理后的结果重新发送到子结点进行合并操作, 重复此过程, 直到返回到树的根结点. 为了减少进程间通信, 充分利用 Cilk 任务并行机制, 需要对原矩阵进行较粗粒度的划分, 以榨取单节点处理器的计算性能.

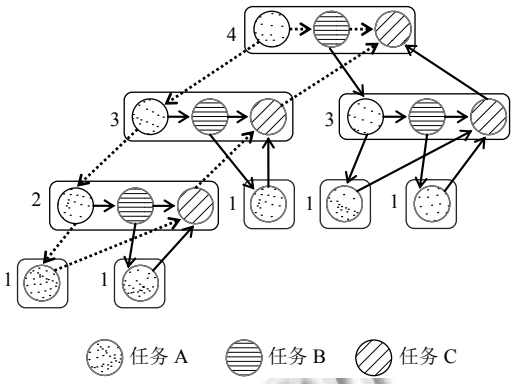

图 1 分治算法的任务并行流程

对于 n 阶对称三对角矩阵 T 在 N 个进程上的特征求解, 假设 $N=2^k$, $n=2^j$, 算法初始时把原矩阵划分为 N 个子矩阵, 每个子矩阵阶数为 2^{j-k}, 下面给出分治算法在多节点上的混合并行实现.

算法 2. 分治算法基于 MPI/Cilk 的节点间并行

首先, 对于划分到 N 个进程上的每个初始子矩阵, 调用算法 1 求出各部分特征值和特征向量;
接着, 进行 p 轮循环 (p 对应逻辑树的高度). 每轮循环先把进程分组并确定主控进程. 每个 MPI 进程根据自己的进程类别按序选择执行下列步骤:
1) 组内进程通过互补通信交换求得的局部特征向量矩阵, 接着向主控进程发送本进程求得的局部特征值和拼接的修正向量;
2) 主控进程收集所有组内进程求得的局部特征值进行排序, 并保留排序后的位置索引数组, 并根据此数组将修正向量排序, 将排好序的特征值和修正向量划分到组内各进程;
3) 组内进程接收主控进程发送过来的排好序的部分特征值和修正向量, 求解 secular 方程和特征向量;
4) 各组内进程按列执行分块矩阵乘法, 更新当前轮循环的部分特征向量矩阵, 接着执行下一轮循环.

2.3 算法分析

图 1 最上方左面对应递归的最外层, 每个出度大于 1 的结点都会派生出新的 Cilk 线程, 执行新的任务. 程序执行过程产生的所有的 Cilk 线程数为图 1 中节点数, 关键路径长度为图 1 中虚线部分路径长度. 程序的并行性为 Cilk 线程数与关键路径长度之比, 执行时间大于等于所有任务平均到每个核上运行的时间和关键路径上任务运行花费的总时间.

算法 2 中在对特征值进行排序后, 同时记录排序后各位置元素的原来位置的索引, 避免对特征向量重复排序, 方便下一轮特征向量的计算. 在第 p 轮循环中, 从 0 号进程开始每 2^{p+1} 个进程分为一组, 每隔 2^{p+1} 个进程被选为选为主控线程. 更新局部特征向量矩阵时,

为了减少数据通信开销, 避免主控进程进行统一收发, 使组内进程间只进行必要的局部特征向量的互补交换, 然后按列执行分块矩阵乘法, 分别计算更新的特征向量矩阵的部分, 通信量为 $O(n^2/N)$.

算法 1 和算法 2 中的合并过程涉及大量计算密集型操作, 占据程序执行的主要时间, 可以使用 Cilk 提供的数据并行机制加速. Cilk 通过数据划分形成一个个可供调度的线程级任务并行执行, 通过基于贪心策略的任务窃取方式执行调度: 当一个核完成自己的任务而其它核还有很多任务未完成, 此时会将忙的核上的任务重新分配给空闲的核.

由于在迭代计算不同特征值的近似值时, 所执行的迭代次数可能不同, 均匀分配任务可能导致线程间的负载不平衡. Cilk 通过工作密取机制从其他线程获取一部分工作量, 能够避免单核上出现任务过载和饥饿等待的问题, 同时能将密取的次数控制在最低水平, 减少密取带来的性能开销. 若单节点内划分的数据粒度适当, 就能利用 Cilk 机制充分发挥多核处理器的计算和调度能力, 得到较好的负载均衡.

3 实验结果与分析

我们在中科院“元”超算上的 cpuII 计算队列进行了数值实验, 节点上搭载的是 Intel E5-2680 V3 芯片, 每块芯片包含 12 颗主频为 2.5 GHz 的计算核心. MPI 程序使用 Intel mpiicpc 编译并链接到 Cilk Plus 运行时. 实验对象是 30 000 阶主对角元素是 4, 副对角元素是 1 实对称三对角矩阵. 矩阵划分求解的规模阈值设为 200 阶, 实验最终求出全部特征值和特征向量 (通过环境变量设置每个节点使用 2~12 个 Cilk 线程进行实验).

表 1 是 MPI 模型与 MPI/Cilk 混合模型在 4~128 个核、2~12 个 Cilk 线程参与计算下所用时间汇总. 从表中可以看出, 相同计算条件下 MPI/Cilk 混合并行计算时间要少于纯 MPI 方法, 而随着参与计算的 Cilk 线程数的增加, 计算用时也越来越少, 这表明当计算任务密集时, 计算效率随着可供调度的线程的增加而增加. 当参与计算的 Cilk 线程数超过 10 时, 对于当前规模的矩阵计算获得的加速提升效果逐渐变得有限, 这是因为 Cilk 是基于贪心策略执行调度的, 当任务粒度较小时不会频繁的执行任务窃取, 从而减小系统开销. 而在较少节点参与计算时, 获得的加速效果较为明显, 这说明对数据进行粗粒度划分时能取得较好性能.

图 2 是 MPI 模型和 MPI/Cilk 模型在不同参数下的加速比变化趋势曲线 (已测出在单节点单线程环境下的平均串行时间为 880 秒左右). 可以看出, 随着参与计算的核数和 Cilk 线程数增加, 三条曲线的上升趋势都逐渐变得平缓, 出现这种现象的原因主要为计算任务粒度变细和进程间通信开销增加导致的. MPI/Cilk 方法的曲线在一开始上升较快, 随着进程数的增加, 计算任务粒度逐渐减小, 加速效果也随之下降, 但仍比纯 MPI 方法变缓得更慢. 这表明 MPI/Cilk 方法的并行效果要优于纯 MPI 方法, 拥有更好的可扩展性. 随着可供调度的核数增多, 使用更多的 Cilk 线程数参与计算能获得更好的计算性能, Cilk 动态调度的优势也渐渐体现了出来.

表 2 给出了 MPI/Cilk 模型 (使用 8 个 Cilk 线程) 同 ELPA、ScaLAPACK 软件包实现的分治算法求解 30000 阶矩阵特征的计算时间比较. ELPA 和 ScaLAPACK 是求解线性代数问题领域中应用广泛的并行软件包. 从表中数据可以得出, 当参与计算的核数较少时, MPI/Cilk 方法与 ELPA、Scalapack 中分治算法计算用时接近, 随着参与计算的核数增多, MPI/Cilk 方法、ELPA 方法与 ScaLAPACK 方法的计算性能逐渐拉开, 但本文的 MPI/Cilk 方法与 ELPA 方法的扩展性仍然存在着一定差距. 这是由于 ELPA 按照块级循环分布矩阵, 实现了对计算流程的更细粒度的控制.

表 1 MPI 模型和 MPI/Cilk 模型运行时间

核数	时间 (秒)						
	MPI/Cilk(12 threads)	MPI/Cilk(10 threads)	MPI/Cilk(8 threads)	MPI/Cilk(6 threads)	MPI/Cilk(4 threads)	MPI/Cilk(2 threads)	PureMPI
4	-	-	-	-	229.14	234.72	241.47
8	-	-	123.72	130.28	139.26	143.86	155.62
16	64.21	66.38	68.67	72.89	77.27	82.26	92.47
32	43.15	45.79	49.90	54.04	58.23	63.39	69.54
64	37.47	39.03	41.76	43.38	46.68	52.44	59.71
128	32.17	34.85	35.42	37.66	41.51	45.35	52.12

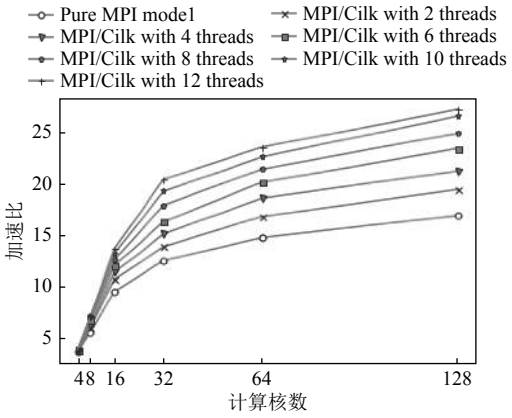

图 2 MPI 模型和 MPI/Cilk 模型的加速比曲线

表 2 MPI/Cilk 方法与 ELPA、ScaLAPACK 求解器时间对比

核数	时间 (秒)		
	ELPA DC solver	MPI/Cilk(8 threads)	Scalapack DC solver
4	214.17	-	269.96
8	119.66	123.72	171.34
16	63.47	68.67	94.51
32	42.93	49.90	62.29
64	30.94	41.76	51.83
128	22.12	35.42	44.36

4 结论与展望

本文针对对称三对角矩阵特征求解的分治算法, 提出了一种改进的基于 MPI/Cilk 的混合并行实现. 在节点间, 进行粗粒度计算任务的划分, 更新局部特征矩阵时通过弱化主控进程的中心作用, 使组内进程之间只进行必要的互补数据交换, 避免了统一收发流程, 减少了进程间的通信开销; 在节点内, 利用 Cilk 的任务并行机制提高了 CPU 的利用率和特征求解过程中的负载均衡性. 实验结果体现了该算法的性能. 本文实现的分治算法与最新的 ELPA 软件包仍然存在一定的性能差距, 还有可以提升的空间. 此外, Cilk 线程启动数和数据划分粒度对计算性能的影响以及同 openMP 等其它任务并行模型的效果对比等等是值得进行下一步研究的方向.

参考文献

1 Cuppen JJM. A divide and conquer method for the symmetric tridiagonal eigenproblem. Numerische Mathematik, 1980, 36(2): 177–195. [doi: 10.1007/BF01396757]

2 Gu M, Eisenstat SC. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. SIAM Journal on Matrix Analysis and Applications, 1994, 15(4): 1266–1276. [doi: 10.1137/S089547989223924X]

3 Tisseur F, Dongarra J. A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures. SIAM Journal on Scientific Computing, 1998, 20(6): 2223–2236.

4 Zhao YH, Chen J, Chi XB. Solving the symmetric tridiagonal eigenproblem using MPI/OpenMP hybrid parallelization. Cao J, Nejdl W, Xu M. Lecture Notes in Computer Science. Berlin: Springer, 2005. 3756. 164–173.

5 Ammar GS, Reichel L, Sorensen DC. An implementation of a divide and conquer algorithm for the unitary eigen problem. ACM Transactions on Mathematical Software, 1992, 18(3): 292–307. [doi: 10.1145/131766.131770]

6 Intel Cilk Plus. Intel Cilk reference manual and documentation, https://www.cilkplus.org/cilk-plus-tutoriall.

7 Melman A. Numerical solution of a secular equation. Numerische Mathematik, 1995, 69(4): 483–493. [doi: 10.1007/s002110050104]

8 Dhillon IS, Parlett BN. Orthogonal eigenvectors and relative gaps. SIAM Journal on Matrix Analysis and Applications, 2003, 25(3): 858–899. [doi: 10.1137/S0895479800370111]

9 Dhillon IS, Parlett BN. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. Linear Algebra and its Applications, 2004, 387: 1–28. [doi: 10.1016/j.laa.2003.12.028]

10 Supercomputing Technologies Group, MIT Laboratory for Computer Science. Cilk-5.4.6 reference manual. Cambridge, Massachusetts, USA: Massachusetts Institute of Technology. http://supertech.csail.mit.edu/cilk/manual-5.4.6.pdf.