

有限元并行计算的 MPI 程序设计

雒战平, 刘之行

(西安交通大学理学院, 710049, 西安)

摘要: 以 Poisson 方程边值问题的求解为背景, 实现了有限元并行计算的 MPI 程序设计. 通过生成一种特殊结构的刚度矩阵, 并在此基础上, 设计了一套有效的并行计算策略, 使计算的并行性得到很好的开拓, 实现了包括刚度矩阵的生成、刚度矩阵的三角分解以及解三角方程组的并行执行. 程序在国家高性能计算中心(西安)的曙光 3000 上进行了数值试验, 结果表明, 随着开辟进程数目的增多, 加速比变得比较理想, 当进程数目为 30 时, 表明该进程数目在最优进程值附近. 在 60 台处理器(进程)上计算 18 万个节点的大规模问题时, 共耗时 176.964 15 s.

关键词: 并行计算; 分布式存储; 数据结构; 有限元方法; MPI 编程

中图分类号: O242.21 **文献标识码:** A **文章编号:** 0253-987X(2004)08-0873-04

MPI Programming for Parallel Computing of Finite Element Method

Luo Zhanping, Liu Zhixing

(School of Sciences, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: Taken solving Poisson boundary problem as a sample, a MPI programming for parallel computing of finite element method is completed. On the basis of generating a kind of stiffness matrix with special structure, a task assignment is proposed to implement parallelly constructing the stiffness matrix, computing the LDL^T factors and solving the triangular systems. Some numerical experiments carried out on the high performance computer SHUGUANG 3000 indicate that higher efficiency can be obtained as the number of processor increases, the optimized number approaches to 30. It takes less than 3 minutes when the problem is with 180 000 nodes.

Key words: parallel computing; distributed storage; data structure; finite element method; MPI programming

并行计算对于大规模科学与工程计算具有十分重要的意义. 目前, 并行计算机的基本存储方式主要有共享存储与分布式存储两种^[1]. 在共享存储的并行机上, 编程相对简单, 却限制了并行计算的扩展性, 而分布式存储的并行机避免了共享内存带来的这一瓶颈, 具有很好的扩展性, 能极大提高并行计算的性能. MPI 消息传递编程模型为分布式存储的并行计算提供了强有力的编程环境. 消息传递(即各个并行执行的进程之间通过消息传递来交换信息、协调步伐、控制执行)需要程序员显式地将负载和数据

分配给进程, 程序员必须解决所有的交互问题, 包括数据映射、通信、同步和聚集. 负载分配通常用拥有者计算法则来完成, 即由拥有数据块的进程来完成相应计算. 在 MPI 的编程模型中, 计算是由一个或多个进程通过调用库函数进行消息收发来实现通信的.

本文对 Poisson 方程第一边值问题的有限元解给出了完整的并行计算模型, 并设计了相应的 MPI 程序. 程序的基本框架对于一大类问题(刚度矩阵对称正定)的有限元法并行求解都是适用的.

收稿日期: 2004-01-12. 作者简介: 雒战平(1975~), 男, 硕士生; 刘之行(联系人), 男, 教授. 基金项目: 国家自然科学基金资助项目(10371095).

1 有限元并行计算方法

本文研究二维 Poisson 方程第一边值问题的有限元法^[2]并行求解,问题模型如下

$$\left. \begin{aligned} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} &= f(x, y) \quad (x, y) \text{ in } G \\ u &= (x, y) \quad (x, y) \text{ on } \partial G \end{aligned} \right\} \quad (1)$$

将区域 G 一维剖分后,产生的刚度矩阵 A 具有如图 1 所示的特殊结构^[3]. A 被分割为 r 个区,共 $(2r-1)^2$ 个子块,除图中标出的子块以外,其他子块(含标记为 X 的子块)均为块零矩阵. A 稀疏且对称正定.对 A 进行三角分解,即 $A = LDL^T$,其中 L 为单位下三角矩阵, D 为对角阵,分解后得到的 L 与矩阵 A 下三角部分的结构基本相同,不同之处仅在于出现了填充子块 X .由三角分解的基本公式可知: r 区中的 L_{ii} 、 $r+1$ 区中的 L_{ij} 分别可以独立并行地计算,只有 r 区中 L_{ij} 的计算,由于数据相关而必须等待.

$A_{1,1}$							
	$A_{2,2}$						
		$A_{3,3}$					
	I 区		\dots			II 区	
				$A_{r,r}$			
$A_{r+1,1}$	$A_{r+1,2}$				$A_{r+1,r}$		
	$A_{r+2,2}$	$A_{r+2,3}$			X	$A_{r+2,r}$	
	II 区	\dots	\dots		X	\dots	
			$A_{2r-1,r-1}$	$A_{2r-1,r}$	III 区	X	$A_{2r-1,2r-1}$

图 1 刚度矩阵的结构

矩阵 A 三角分解后,考虑方程组 $(LDL^T)x = f$ 的求解.令 $y = L^T x$,由式 $LDy = f$ 执行前代过程解出 y ,再由式 $L^T x = y$ 执行回代过程解出 x .其中 $f = (f_1, f_2, \dots, f_{2r-1})^T$, $y = (y_1, y_2, \dots, y_{2r-1})^T$, $x = (x_1, x_2, \dots, x_{2r-1})^T$,这里 f_i, y_i, x_i 均为与子块对应的子向量.上述 y_i 和 x_i 绝大部分能独立并行计算,仅有少部分不能并行计算.

2 并程序序设计

2.1 处理器任务的分配与负载的平衡

我们称不可分割的计算单元为一个任务^[4],那

么 A 中非零块矩阵 A_{ij} 的生成以及分解 A 得到对应位置的 L_{ij} 都可视为一个任务.并行计算是最大限度地开拓任务并行执行的机会,尽量减少通讯次数,同时要考虑到数据的分布式存储^[5]与处理器间负载的相对平衡.

图 2 中以子区域数目 $r=4$ 、处理器数目 $i=4$ 为例,给出了对应的任务分配方式.

p_1							
	p_2						
		p_3					
			p_4				
p_1	p_2				p_1		
	p_2	p_3			p_2	p_2	
		p_3	p_4		p_3	p_3	

图 2 任务的分配

一般地说,如果有 r 个处理器参与计算,则将 G 分割为 r 块.此时,第 $i(2 \leq i \leq r-1)$ 个处理器 p_i 将进行如下计算:在生成刚度矩阵阶段,生成刚度矩阵的子块 $A_{ii}, A_{r+i,i}, A_{r+i-1,i}, A_{r+i,r+i}$ 以及右端向量 f_i, f_{r+i} ;在三角分解阶段,计算生成 $L_{ii}, D_{ii}, L_{r+i,i}, L_{r+i-1,i}, L_{r+i,r+i-1}, L_{r+i,r+i}, D_{r+i,r+i}$;在解方程组后期工作中,计算 y_i, y_{i+r} 和 x_i, x_{i+r} .相应地,以上这些数据就存储于 p_i 的内存中,计算过程中需要其他数据时可以通过消息传递从其他处理器中获得. p_1, p_r 的计算任务量有所不同,此处不再详述.显然,上述任务分配方式具有相当好的负载平衡.

2.2 分布式存储的数据结构

由于 A 对称正定,所以存储时只存储下半部分(包括对角块).由 2.1 节知, A 分布式存储于参与计算的处理器 p_i 中.每个处理器 p_i 都存储有 A 中 r 区的一部分块矩阵,同区的块矩阵有相同的数据结构. r 区、 $r+1$ 区、 $r+2$ 区、 $r+3$ 区、 $r+4$ 区、 $r+5$ 区、 $r+6$ 区、 $r+7$ 区、 $r+8$ 区、 $r+9$ 区、 $r+10$ 区、 $r+11$ 区、 $r+12$ 区、 $r+13$ 区、 $r+14$ 区、 $r+15$ 区、 $r+16$ 区、 $r+17$ 区、 $r+18$ 区、 $r+19$ 区、 $r+20$ 区、 $r+21$ 区、 $r+22$ 区、 $r+23$ 区、 $r+24$ 区、 $r+25$ 区、 $r+26$ 区、 $r+27$ 区、 $r+28$ 区、 $r+29$ 区、 $r+30$ 区、 $r+31$ 区、 $r+32$ 区、 $r+33$ 区、 $r+34$ 区、 $r+35$ 区、 $r+36$ 区、 $r+37$ 区、 $r+38$ 区、 $r+39$ 区、 $r+40$ 区、 $r+41$ 区、 $r+42$ 区、 $r+43$ 区、 $r+44$ 区、 $r+45$ 区、 $r+46$ 区、 $r+47$ 区、 $r+48$ 区、 $r+49$ 区、 $r+50$ 区、 $r+51$ 区、 $r+52$ 区、 $r+53$ 区、 $r+54$ 区、 $r+55$ 区、 $r+56$ 区、 $r+57$ 区、 $r+58$ 区、 $r+59$ 区、 $r+60$ 区、 $r+61$ 区、 $r+62$ 区、 $r+63$ 区、 $r+64$ 区、 $r+65$ 区、 $r+66$ 区、 $r+67$ 区、 $r+68$ 区、 $r+69$ 区、 $r+70$ 区、 $r+71$ 区、 $r+72$ 区、 $r+73$ 区、 $r+74$ 区、 $r+75$ 区、 $r+76$ 区、 $r+77$ 区、 $r+78$ 区、 $r+79$ 区、 $r+80$ 区、 $r+81$ 区、 $r+82$ 区、 $r+83$ 区、 $r+84$ 区、 $r+85$ 区、 $r+86$ 区、 $r+87$ 区、 $r+88$ 区、 $r+89$ 区、 $r+90$ 区、 $r+91$ 区、 $r+92$ 区、 $r+93$ 区、 $r+94$ 区、 $r+95$ 区、 $r+96$ 区、 $r+97$ 区、 $r+98$ 区、 $r+99$ 区、 $r+100$ 区、 $r+101$ 区、 $r+102$ 区、 $r+103$ 区、 $r+104$ 区、 $r+105$ 区、 $r+106$ 区、 $r+107$ 区、 $r+108$ 区、 $r+109$ 区、 $r+110$ 区、 $r+111$ 区、 $r+112$ 区、 $r+113$ 区、 $r+114$ 区、 $r+115$ 区、 $r+116$ 区、 $r+117$ 区、 $r+118$ 区、 $r+119$ 区、 $r+120$ 区、 $r+121$ 区、 $r+122$ 区、 $r+123$ 区、 $r+124$ 区、 $r+125$ 区、 $r+126$ 区、 $r+127$ 区、 $r+128$ 区、 $r+129$ 区、 $r+130$ 区、 $r+131$ 区、 $r+132$ 区、 $r+133$ 区、 $r+134$ 区、 $r+135$ 区、 $r+136$ 区、 $r+137$ 区、 $r+138$ 区、 $r+139$ 区、 $r+140$ 区、 $r+141$ 区、 $r+142$ 区、 $r+143$ 区、 $r+144$ 区、 $r+145$ 区、 $r+146$ 区、 $r+147$ 区、 $r+148$ 区、 $r+149$ 区、 $r+150$ 区、 $r+151$ 区、 $r+152$ 区、 $r+153$ 区、 $r+154$ 区、 $r+155$ 区、 $r+156$ 区、 $r+157$ 区、 $r+158$ 区、 $r+159$ 区、 $r+160$ 区、 $r+161$ 区、 $r+162$ 区、 $r+163$ 区、 $r+164$ 区、 $r+165$ 区、 $r+166$ 区、 $r+167$ 区、 $r+168$ 区、 $r+169$ 区、 $r+170$ 区、 $r+171$ 区、 $r+172$ 区、 $r+173$ 区、 $r+174$ 区、 $r+175$ 区、 $r+176$ 区、 $r+177$ 区、 $r+178$ 区、 $r+179$ 区、 $r+180$ 区、 $r+181$ 区、 $r+182$ 区、 $r+183$ 区、 $r+184$ 区、 $r+185$ 区、 $r+186$ 区、 $r+187$ 区、 $r+188$ 区、 $r+189$ 区、 $r+190$ 区、 $r+191$ 区、 $r+192$ 区、 $r+193$ 区、 $r+194$ 区、 $r+195$ 区、 $r+196$ 区、 $r+197$ 区、 $r+198$ 区、 $r+199$ 区、 $r+200$ 区、 $r+201$ 区、 $r+202$ 区、 $r+203$ 区、 $r+204$ 区、 $r+205$ 区、 $r+206$ 区、 $r+207$ 区、 $r+208$ 区、 $r+209$ 区、 $r+210$ 区、 $r+211$ 区、 $r+212$ 区、 $r+213$ 区、 $r+214$ 区、 $r+215$ 区、 $r+216$ 区、 $r+217$ 区、 $r+218$ 区、 $r+219$ 区、 $r+220$ 区、 $r+221$ 区、 $r+222$ 区、 $r+223$ 区、 $r+224$ 区、 $r+225$ 区、 $r+226$ 区、 $r+227$ 区、 $r+228$ 区、 $r+229$ 区、 $r+230$ 区、 $r+231$ 区、 $r+232$ 区、 $r+233$ 区、 $r+234$ 区、 $r+235$ 区、 $r+236$ 区、 $r+237$ 区、 $r+238$ 区、 $r+239$ 区、 $r+240$ 区、 $r+241$ 区、 $r+242$ 区、 $r+243$ 区、 $r+244$ 区、 $r+245$ 区、 $r+246$ 区、 $r+247$ 区、 $r+248$ 区、 $r+249$ 区、 $r+250$ 区、 $r+251$ 区、 $r+252$ 区、 $r+253$ 区、 $r+254$ 区、 $r+255$ 区、 $r+256$ 区、 $r+257$ 区、 $r+258$ 区、 $r+259$ 区、 $r+260$ 区、 $r+261$ 区、 $r+262$ 区、 $r+263$ 区、 $r+264$ 区、 $r+265$ 区、 $r+266$ 区、 $r+267$ 区、 $r+268$ 区、 $r+269$ 区、 $r+270$ 区、 $r+271$ 区、 $r+272$ 区、 $r+273$ 区、 $r+274$ 区、 $r+275$ 区、 $r+276$ 区、 $r+277$ 区、 $r+278$ 区、 $r+279$ 区、 $r+280$ 区、 $r+281$ 区、 $r+282$ 区、 $r+283$ 区、 $r+284$ 区、 $r+285$ 区、 $r+286$ 区、 $r+287$ 区、 $r+288$ 区、 $r+289$ 区、 $r+290$ 区、 $r+291$ 区、 $r+292$ 区、 $r+293$ 区、 $r+294$ 区、 $r+295$ 区、 $r+296$ 区、 $r+297$ 区、 $r+298$ 区、 $r+299$ 区、 $r+300$ 区、 $r+301$ 区、 $r+302$ 区、 $r+303$ 区、 $r+304$ 区、 $r+305$ 区、 $r+306$ 区、 $r+307$ 区、 $r+308$ 区、 $r+309$ 区、 $r+310$ 区、 $r+311$ 区、 $r+312$ 区、 $r+313$ 区、 $r+314$ 区、 $r+315$ 区、 $r+316$ 区、 $r+317$ 区、 $r+318$ 区、 $r+319$ 区、 $r+320$ 区、 $r+321$ 区、 $r+322$ 区、 $r+323$ 区、 $r+324$ 区、 $r+325$ 区、 $r+326$ 区、 $r+327$ 区、 $r+328$ 区、 $r+329$ 区、 $r+330$ 区、 $r+331$ 区、 $r+332$ 区、 $r+333$ 区、 $r+334$ 区、 $r+335$ 区、 $r+336$ 区、 $r+337$ 区、 $r+338$ 区、 $r+339$ 区、 $r+340$ 区、 $r+341$ 区、 $r+342$ 区、 $r+343$ 区、 $r+344$ 区、 $r+345$ 区、 $r+346$ 区、 $r+347$ 区、 $r+348$ 区、 $r+349$ 区、 $r+350$ 区、 $r+351$ 区、 $r+352$ 区、 $r+353$ 区、 $r+354$ 区、 $r+355$ 区、 $r+356$ 区、 $r+357$ 区、 $r+358$ 区、 $r+359$ 区、 $r+360$ 区、 $r+361$ 区、 $r+362$ 区、 $r+363$ 区、 $r+364$ 区、 $r+365$ 区、 $r+366$ 区、 $r+367$ 区、 $r+368$ 区、 $r+369$ 区、 $r+370$ 区、 $r+371$ 区、 $r+372$ 区、 $r+373$ 区、 $r+374$ 区、 $r+375$ 区、 $r+376$ 区、 $r+377$ 区、 $r+378$ 区、 $r+379$ 区、 $r+380$ 区、 $r+381$ 区、 $r+382$ 区、 $r+383$ 区、 $r+384$ 区、 $r+385$ 区、 $r+386$ 区、 $r+387$ 区、 $r+388$ 区、 $r+389$ 区、 $r+390$ 区、 $r+391$ 区、 $r+392$ 区、 $r+393$ 区、 $r+394$ 区、 $r+395$ 区、 $r+396$ 区、 $r+397$ 区、 $r+398$ 区、 $r+399$ 区、 $r+400$ 区、 $r+401$ 区、 $r+402$ 区、 $r+403$ 区、 $r+404$ 区、 $r+405$ 区、 $r+406$ 区、 $r+407$ 区、 $r+408$ 区、 $r+409$ 区、 $r+410$ 区、 $r+411$ 区、 $r+412$ 区、 $r+413$ 区、 $r+414$ 区、 $r+415$ 区、 $r+416$ 区、 $r+417$ 区、 $r+418$ 区、 $r+419$ 区、 $r+420$ 区、 $r+421$ 区、 $r+422$ 区、 $r+423$ 区、 $r+424$ 区、 $r+425$ 区、 $r+426$ 区、 $r+427$ 区、 $r+428$ 区、 $r+429$ 区、 $r+430$ 区、 $r+431$ 区、 $r+432$ 区、 $r+433$ 区、 $r+434$ 区、 $r+435$ 区、 $r+436$ 区、 $r+437$ 区、 $r+438$ 区、 $r+439$ 区、 $r+440$ 区、 $r+441$ 区、 $r+442$ 区、 $r+443$ 区、 $r+444$ 区、 $r+445$ 区、 $r+446$ 区、 $r+447$ 区、 $r+448$ 区、 $r+449$ 区、 $r+450$ 区、 $r+451$ 区、 $r+452$ 区、 $r+453$ 区、 $r+454$ 区、 $r+455$ 区、 $r+456$ 区、 $r+457$ 区、 $r+458$ 区、 $r+459$ 区、 $r+460$ 区、 $r+461$ 区、 $r+462$ 区、 $r+463$ 区、 $r+464$ 区、 $r+465$ 区、 $r+466$ 区、 $r+467$ 区、 $r+468$ 区、 $r+469$ 区、 $r+470$ 区、 $r+471$ 区、 $r+472$ 区、 $r+473$ 区、 $r+474$ 区、 $r+475$ 区、 $r+476$ 区、 $r+477$ 区、 $r+478$ 区、 $r+479$ 区、 $r+480$ 区、 $r+481$ 区、 $r+482$ 区、 $r+483$ 区、 $r+484$ 区、 $r+485$ 区、 $r+486$ 区、 $r+487$ 区、 $r+488$ 区、 $r+489$ 区、 $r+490$ 区、 $r+491$ 区、 $r+492$ 区、 $r+493$ 区、 $r+494$ 区、 $r+495$ 区、 $r+496$ 区、 $r+497$ 区、 $r+498$ 区、 $r+499$ 区、 $r+500$ 区、 $r+501$ 区、 $r+502$ 区、 $r+503$ 区、 $r+504$ 区、 $r+505$ 区、 $r+506$ 区、 $r+507$ 区、 $r+508$ 区、 $r+509$ 区、 $r+510$ 区、 $r+511$ 区、 $r+512$ 区、 $r+513$ 区、 $r+514$ 区、 $r+515$ 区、 $r+516$ 区、 $r+517$ 区、 $r+518$ 区、 $r+519$ 区、 $r+520$ 区、 $r+521$ 区、 $r+522$ 区、 $r+523$ 区、 $r+524$ 区、 $r+525$ 区、 $r+526$ 区、 $r+527$ 区、 $r+528$ 区、 $r+529$ 区、 $r+530$ 区、 $r+531$ 区、 $r+532$ 区、 $r+533$ 区、 $r+534$ 区、 $r+535$ 区、 $r+536$ 区、 $r+537$ 区、 $r+538$ 区、 $r+539$ 区、 $r+540$ 区、 $r+541$ 区、 $r+542$ 区、 $r+543$ 区、 $r+544$ 区、 $r+545$ 区、 $r+546$ 区、 $r+547$ 区、 $r+548$ 区、 $r+549$ 区、 $r+550$ 区、 $r+551$ 区、 $r+552$ 区、 $r+553$ 区、 $r+554$ 区、 $r+555$ 区、 $r+556$ 区、 $r+557$ 区、 $r+558$ 区、 $r+559$ 区、 $r+560$ 区、 $r+561$ 区、 $r+562$ 区、 $r+563$ 区、 $r+564$ 区、 $r+565$ 区、 $r+566$ 区、 $r+567$ 区、 $r+568$ 区、 $r+569$ 区、 $r+570$ 区、 $r+571$ 区、 $r+572$ 区、 $r+573$ 区、 $r+574$ 区、 $r+575$ 区、 $r+576$ 区、 $r+577$ 区、 $r+578$ 区、 $r+579$ 区、 $r+580$ 区、 $r+581$ 区、 $r+582$ 区、 $r+583$ 区、 $r+584$ 区、 $r+585$ 区、 $r+586$ 区、 $r+587$ 区、 $r+588$ 区、 $r+589$ 区、 $r+590$ 区、 $r+591$ 区、 $r+592$ 区、 $r+593$ 区、 $r+594$ 区、 $r+595$ 区、 $r+596$ 区、 $r+597$ 区、 $r+598$ 区、 $r+599$ 区、 $r+600$ 区、 $r+601$ 区、 $r+602$ 区、 $r+603$ 区、 $r+604$ 区、 $r+605$ 区、 $r+606$ 区、 $r+607$ 区、 $r+608$ 区、 $r+609$ 区、 $r+610$ 区、 $r+611$ 区、 $r+612$ 区、 $r+613$ 区、 $r+614$ 区、 $r+615$ 区、 $r+616$ 区、 $r+617$ 区、 $r+618$ 区、 $r+619$ 区、 $r+620$ 区、 $r+621$ 区、 $r+622$ 区、 $r+623$ 区、 $r+624$ 区、 $r+625$ 区、 $r+626$ 区、 $r+627$ 区、 $r+628$ 区、 $r+629$ 区、 $r+630$ 区、 $r+631$ 区、 $r+632$ 区、 $r+633$ 区、 $r+634$ 区、 $r+635$ 区、 $r+636$ 区、 $r+637$ 区、 $r+638$ 区、 $r+639$ 区、 $r+640$ 区、 $r+641$ 区、 $r+642$ 区、 $r+643$ 区、 $r+644$ 区、 $r+645$ 区、 $r+646$ 区、 $r+647$ 区、 $r+648$ 区、 $r+649$ 区、 $r+650$ 区、 $r+651$ 区、 $r+652$ 区、 $r+653$ 区、 $r+654$ 区、 $r+655$ 区、 $r+656$ 区、 $r+657$ 区、 $r+658$ 区、 $r+659$ 区、 $r+660$ 区、 $r+661$ 区、 $r+662$ 区、 $r+663$ 区、 $r+664$ 区、 $r+665$ 区、 $r+666$ 区、 $r+667$ 区、 $r+668$ 区、 $r+669$ 区、 $r+670$ 区、 $r+671$ 区、 $r+672$ 区、 $r+673$ 区、 $r+674$ 区、 $r+675$ 区、 $r+676$ 区、 $r+677$ 区、 $r+678$ 区、 $r+679$ 区、 $r+680$ 区、 $r+681$ 区、 $r+682$ 区、 $r+683$ 区、 $r+684$ 区、 $r+685$ 区、 $r+686$ 区、 $r+687$ 区、 $r+688$ 区、 $r+689$ 区、 $r+690$ 区、 $r+691$ 区、 $r+692$ 区、 $r+693$ 区、 $r+694$ 区、 $r+695$ 区、 $r+696$ 区、 $r+697$ 区、 $r+698$ 区、 $r+699$ 区、 $r+700$ 区、 $r+701$ 区、 $r+702$ 区、 $r+703$ 区、 $r+704$ 区、 $r+705$ 区、 $r+706$ 区、 $r+707$ 区、 $r+708$ 区、 $r+709$ 区、 $r+710$ 区、 $r+711$ 区、 $r+712$ 区、 $r+713$ 区、 $r+714$ 区、 $r+715$ 区、 $r+716$ 区、 $r+717$ 区、 $r+718$ 区、 $r+719$ 区、 $r+720$ 区、 $r+721$ 区、 $r+722$ 区、 $r+723$ 区、 $r+724$ 区、 $r+725$ 区、 $r+726$ 区、 $r+727$ 区、 $r+728$ 区、 $r+729$ 区、 $r+730$ 区、 $r+731$ 区、 $r+732$ 区、 $r+733$ 区、 $r+734$ 区、 $r+735$ 区、 $r+736$ 区、 $r+737$ 区、 $r+738$ 区、 $r+739$ 区、 $r+740$ 区、 $r+741$ 区、 $r+742$ 区、 $r+743$

块对称正定,只存储下半部分,次对角块满存储.

2.3 刚度矩阵的并行生成

刚度矩阵的并行生成比较重要,在此我们重点讨论.实现 A 的分布式存储,处理器 p_i 只需生成对应子区域 G_i 的若干子块矩阵即可.如图 3,以矩形区域 G 为例,它被分割为 3 个子区域 G_1, G_2, G_3 ,刚度矩阵 A 对应被分为 $5 \times 5 = 25$ 个子块,对应有如图 1 的块结构.我们以子区域 G_2 为例,给出其生成刚度矩阵对应子块矩阵的详细过程.

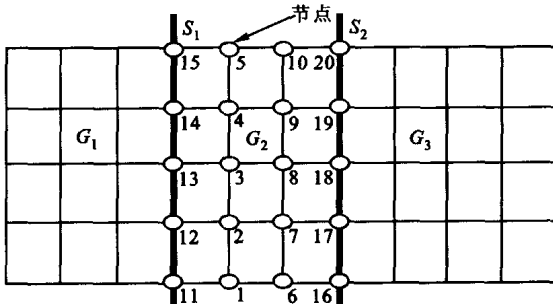


图 3 子块刚度矩阵的合成

子区域 G_2 的节点整体编码见图 3,合成时采用四节点矩形元素.内部节点数目 $n = 10$,左分割线上节点数目 $n_L = 5$,右分割线上节点数目 $n_R = 5$, $A_{2,2}$ 为 $n = 10$ 阶方阵, $A_{4,2}$ 为 5 行 10 列 ($n_L = 5, n = 10$) 矩阵, $A_{5,2}$ 为 5 行 10 列 ($n_R = 5, n = 10$) 矩阵, $A_{4,4}$ 为 5 行 5 列 ($n_L = 5$) 矩阵, $A_{5,5}$ 为 5 行 5 列 ($n_R = 5$) 矩阵.

令 i_1, i_2 为 G_2 中相互影响的两个节点的整体编码,刚度矩阵 A 的生成信息将会存入 $A_{2,2}, A_{4,2}, A_{5,2}, A_{4,4}, A_{5,5}$ 其中的一个子块矩阵内,不妨设有 i_1, i_2 .

(1) 当 $1 \leq i_1, i_2 \leq 10$ 时,其生成的信息存入子块矩阵 $A_{2,2}$ 中.

(2) 当 $11 \leq i_1 \leq 15, 1 \leq i_2 \leq 10$ 时,其生成的信息存入子块矩阵 $A_{4,2}$ 中.

(3) 当 $16 \leq i_1 \leq 20, 1 \leq i_2 \leq 10$ 时,其生成的信息存入子块矩阵 $A_{5,2}$ 中.

(4) 当 $11 \leq i_1 \leq 15, 11 \leq i_2 \leq 15$ 时,其生成的信息存入子块矩阵 $A_{4,4}$ 中.

(5) 当 $16 \leq i_1 \leq 20, 16 \leq i_2 \leq 20$ 时,其生成的信息存入子块矩阵 $A_{5,5}$ 中.

一般地,区域 G 被分割为 r 个子块,处理器 p_i 在完成上述计算后,其内存中存储有子块矩阵和向量实体 $A_{i,i}, f_i, A_{r+i-1,i}, A_{r+i,i}, A_{r+i-1,r+i-1}, f_{r+i-1}, A_{r+i,r+i}, f_{r+i} (A_{r+i-1,r+i-1}, A_{r+i,r+i})$ 表示

子块矩阵 $A_{r+i-1,r+i-1}, A_{r+i,r+i}$ 的一部分, f_{r+i-1}, f_{r+i} 表示子向量 f_{r+i-1}, f_{r+i} 的一部分,下同).我们用 MPI 将 p_i 合成的 $A_{r+i-1,r+i-1}, f_{r+i-1}$ 发送到 p_{i-1} ,叠加到处理器 p_{i-1} 的 $A_{r+i-1,r+i-1}, f_{r+i-1}$ 中去,这样处理器 p_{i-1} 的子块矩阵 $A_{i+r-1,i+r-1}$,子向量 f_{r+i-1} 完整.同时, p_i 从 p_{i+1} 中接收 p_{i+1} 合成的 $A_{i+r,i+r}, f_{i+r}$,叠加到本地内存中的 $A_{i+r,i+r}, f_{r+i}$ 中去,这样本地子块矩阵 $A_{i+r,i+r}$,子向量 f_{r+i} 完整.每个处理器 p_i 完成上述过程之后,就实现了 A 的分布式存储.计算时,刚度矩阵 A 只是概念上存在,并无一个具体的数据实体.上述处理器 p_i 的概念在 MPI 并行计算中,对应于概念进程 p_i .

3 数值试验

式(1)中取 $f(x, y) = 2x, G: 0 \leq x \leq 10, 0 \leq y \leq 1$, $(x, y) = \begin{cases} 0 & x=0 \text{ 或 } y=0 \\ x & y=1 \\ 10y^2 & x=10 \end{cases}$, 此时式(1)问题

的精确解为 $u = xy^2$.为试验方便,子区域块 G_i 间、区域内部都进行等距剖分,合成时采用四节点矩形元素.令 t_i 为进程 p_i 完成本处理器计算所花费的时间,并行计算的时间定义为 $t = \max(t_i)$.加速比 k 定义为 $k = T/t$,其中 T 表示串行计算花费的时间.区域 G_i 误差计算公式为

$$E_i = \left[\sum_{i=1}^{n_i} (U_i - u_i)^2 \right]^{1/2} / \left[\sum_{i=1}^{n_i} U_i^2 \right]^{1/2}$$

式中: U_i 为精确解; u_i 为有限元解; n_i 为进程 p_i 求解点的个数.并行计算的误差定义为 $E = \max(E_i)$.

对本文所研制的程序在国家高性能计算中心(西安)的曙光 3000 上进行了数值试验,结果如表 1、表 2 所示.

表 1 数值结果 1

n	t/s	k	$E/\%$
1	41.575 25		7.57×10^{-5}
5	7.286 36	5.71	$< 1.0 \times 10^{-10}$
10	3.433 43	12.11	$< 1.0 \times 10^{-10}$
20	1.879 58	22.12	$< 1.0 \times 10^{-10}$
30	0.987 72	42.09	$< 1.0 \times 10^{-10}$
50	0.849 53	48.94	$< 1.0 \times 10^{-10}$

表 1 中 G 上分布节点为 5 643 个,横向网格线为 19 条,纵向风格线平均分配给各子区域块.从表

1 可以看出,当我们开辟的进程数目较少时,加速比并不理想,随着开辟进程数目的增多,加速比变得理想。 $n=30$ 表明该进程数目在最优进程值附近。应当指出的是,表 1 中的加速比不是严格意义上的加速比,严格意义上的加速比是指完成同样的计算量时串行耗时与并行耗时之比。在此意义下,加速比超过进程数是不可能的。表 1 中有加速比大于进程数的情形(如 $n=10$ 时),这是由于同时使用了区域分裂算法与并行计算而产生的双重功效(区域分裂算法能减少计算量)。

表 2 数值结果 2

n	t/s	$E/\%$
20	70.320 06	$<1.0 \times 10^{-10}$
30	34.188 94	$<1.0 \times 10^{-10}$
40	29.416 81	$<1.0 \times 10^{-10}$
50	25.359 81	$<1.0 \times 10^{-10}$

表 2 中区域 G 上分布节点为 50 000 个,横向网格线为 19 条,纵向网格线平均分配给各子区域块。此时,计算的规模太大,已经不能由单独的处理器(进程)做串行计算来完成,故表 2 中不能给出加速

比。最后,我们还完成了 60 台处理器(进程)计算 18 万个节点的大规模问题,耗时 176.964 15 s。

4 结束语

本文设计的并行程序有很好的加速效果,适合于大规模科学与工程计算。对于大型问题,往往由于内存的限制而借助于外存设备,但外存设备的存取速度要缓慢得多,这就极大地降低了问题的求解速度。程序中由于数据分布式存储,使得只使用各处理器的内存,就可以求解相当大规模的问题。

参考文献:

- [1] 都志辉. 高性能计算并行编程技术[M]. 北京:清华大学出版社,2001.
- [2] 李开泰,黄艾香,黄庆怀. 有限元方法及其应用[M]. 西安:西安交通大学出版社,1998.
- [3] 刘长学. 超大规模稀疏矩阵计算方法[M]. 上海:上海科学技术出版社,1991.
- [4] Lin Haixiang. A methodology for the parallel direct solution of finite element systems[D]. Delft: Delft University of Technology, 1993.
- [5] Tanenbaum A S. Distributed operating systems[M]. Beijing: Tsinghua University Press, 1997.

(编辑 杜秀杰)

[文摘预登]

基于 JPEG2000 芯片的小波系数存储

梅魁志, 郑南宁, 王 勇, 曹 非, 兰旭光

(西安交通大学人工智能与机器人研究所, 710049, 西安)

针对 JPEG2000 芯片设计中的完全小波系数存储占用大量存储器问题,在小波变换总体结构中对小波系数的 LH、HL 和 HH 子带采用双缓存的物理存储结构。为了解决由此产生的写覆盖,首先建立小波滤波器的时序模型得到输入、输出延时时钟数,根据此时钟数和缓存标志位的状态决定其输入地址发生器的地址,以实现可控的小波滤波器输出。在图像片大小为 256×256 、码块大小为 16×16 时,与完全系数存储结构相比,可节约片上存储器达 576 kb。同时,子带内小波系数的分布模型和缓存内的位平面数统计分析表明,该结构对编码并行性的影响较小,仿真试验结果证明其降低的效率不会超过 2%。