国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第13卷第3期 1992年9月 Vol. 14 No. 3

巨型机通用数学库软件与并行算法*

胡庆丰 李晓梅

（电子计算机系）

摘 要 本文介绍了巨型机通用数学库软件的概念和意义，指出了巨型机数学软件开发的技术途径与发展趋势，讨论了并行算法在其中的关键性作用，结合实例分析了巨型机通用数学库软件研制中的并行算法设计与实现。

关键词 巨型机，科学计算程序，并行算法，数学软件

分类号 TP311，TP317

当今计算能力与存储容量最为强大的通用高性能计算机系统——巨型机正在大型科学工程计算和大规模数据处理中发挥重要的作用。用户对于丰富和完善巨型机通用数学库软件提出更高的要求。巨型机用户不仅要求提供功能齐全、正确可靠的数学软件环境，更希望这些软件高速有效，充分发挥巨型机的超级计算能力。

通用数学库软件是执行基本数学计算的一种应用软件产品，是最基础、最通用的应用库软件之一，是科技计算和应用软件开发不可缺少的基本构件。由于巨型机系统体系结构的多样化和复杂性，传统的串行数学软件产品在算法的设计与实现上没有考虑其并行特性，加之目前的自动向量化和自动并行化技术尚难以充分开发程序的并行性，因而比起传统的串行计算环境来，人们更希望巨型机有充实、方便、高效的通用数学库软件，以支撑用户解题过程，发挥巨型机高速计算优势。

1 巨型机通用数学库软件的开发

巨型机通用数学库软件的开发经历了两个阶段。从以 Cray-1 为代表的向量巨型机投入使用至今，人们结合向量计算特点，开发算法中的向量成分，主要采用循环级和程序级的优化，以拟合向量处理体系结构，在原有的标量计算软件产品基础上，开发出了一批向量化数学库软件。我们已在 YH-1 上建立了向量线性代数库、向量特征值特征向量库等；同时，针对具体的机器环境，重点优化最基本的计算内核，如 YH-1 科学计算库，其中的核心子程序包由高度向量化的银河汇编子程序组成，而其它 FORTRAN 子程序则通过调用经优化的汇编代码，采用循环展开、循环优化、使用银河向量 FORTRAN 语言等优化措施以实现其高速运行。

Cray X-MP 等向量多处理机的问世，使得并行计算库软件开发成为必需。国际上纷纷开始研究并行数学软件，如美国 Argonne 实验室等机构和 Purdue 大学等单位正在研制并行计算的线性代数程序库 LAPACK 以及椭圆型偏微分方程解题环境 ELLPACK 等。结合向量多处理机系统的研制，我们也正在开展并行数学库软件的研究工作。

* 国家自然科学基金资助项目
1991年9月4日收稿

与通用数学软件着重于解决易用性、重用性和解题的整体效率这一发展趋势相适应，巨型机数学软件不仅要致力于开发标准软部件和改善用户接口，而且要改进数值软件开发环境和解题环境，更应当考虑到巨型机的不同结构将直接影响算法设计和实现，需透彻了解目标机的内部结构，进行并行算法研究工作。当前巨型机数学库软件研制中的主要技术途径有如下三个方面。

(1) 模块化

定义一些基本算法模块作为库软件的核心，并为不同巨型机结构编写相适配的 FORTRAN 程序版本或汇编代码，使之局部优化，以充分利用具体结构的特点，高度发挥目标机的运行效率，从而使基于其上的其它算法模块也得到较好的运行效果。

线性代数计算是数值计算中最基本的内容之一。当前，基本线代数计算程序包 BLAS (Basic Linear Algebra Subprograms) 已成为向量化与并行化数学软件的核心。为提高并行计算的粒度，BLAS 的功能经多次扩充已形成三级标准：BLAS-1 涉及向量与向量的运算，BLAS-2 涉及矩阵与向量的运算，BLAS-3 涉及矩阵与矩阵的运算。

(2) 研究并行算法

针对向量和并行体系结构设计新的数值算法，或修改原有的算法。这里一方面是对上面的核心模块作重点优化，一方面则是研究如何将数学问题分解为对这些核心模块的调用。

并行算法设计所采取的策略是对求解问题的重新排序 (reordering) 和分而治之 (devide and conquer)，将问题重新排序或分解为若干个不相关或弱相关的子问题以并行处理。这些子问题一般并非完全独立，需要在运行过程中进行机间通讯以交换信息或实现同步。例如求解偏微分方程组的区域分裂法，由于各子域间存在公共边界，从而在计算进程中必须交换公共边界上的信息。

并行计算机体系结构对数值算法有不容忽视的影响，算法的设计与实现同目标机结构密切相关，必须改变我们习惯的顺序处理观念，探索新的设计思想和实现技术。

(3) 建立新的适合并行处理的语言

作为通用数学库软件的主要程序设计语言，标准 FORTRAN 是典型的串行处理语言。当前的解决办法是采用扩充了的 FORTRAN 语言，建立包含向量/并行识别与处理的 FORTRAN 编译器，为并行软件研制提供必要的工具支撑。如银河向量 FORTRAN 已包含了向量处理成分。期望有包含并行处理成分的 FORTRAN 或新的高级语言问世，以便更好地表达并行算法和避免程序对机器特性的依赖。这样将使为追求程序高效而不得不使用复杂繁琐的汇编语言编码的情形得到改观。

必须指出，并行计算数学软件开发环境和解题环境的研究，将使巨型机通用数学库软件开发产生新的飞跃。

2 并行通用数学库软件开发范例——LAPACK 计划

LAPACK (Linear Algebra Package) 是由 Argonne 实验室、Courant 数学研究所、NAG 公司等共同计划，并正在开发中的 FORTRAN 77 线性代数库。其目的是提供一个一致的求解最普通线性代数问题的子程序集，并且能在当前一系列高性能计算机上有效地运行。

该库以成功的 EISPACK 和 LINPACK 两库为基础，功能包括线性方程组求解、超定方程的最小二乘解、特征值问题、相应的矩阵分解和条件数计算等。这些功能针对实和复的稠密和带状矩阵，但不含一般稀疏阵。这一新软件库将两个子集的算法集成为一个统一的系统化库，并特别致力于新的方法和算法的设计。该库设计的目标机由适当数量 (比如说 1 至 100) 的高性能向量机组成的多处理机系统，这类机器几乎包括所有当前可用于一般科学计算的巨型计算机。

为了达到设计目标，该库的研制不仅继承了原有算法库的高可靠性、绝对可移植性、构件的非递归性和划一的接口设计、优良的程序和文档设计风格等标准数学软部件设计技术，还特别采取了以下技术措施。

(1) 算法实现的优化

巨型机大都具有分级存储结构，以平衡存储访问和向量化浮点操作间的速度差异，编程时必须注意数据的重用，避免程序以存储速度而非浮点速度运行。然而，LINPACK 和 EISPACK 代码的书写风格在大多数情况下均忽略了数据传输的开销。LAPACK 程序则经过仔细的编码，使尽可能多的数据被重复使用，以减少数据传输的开销。

(2) 研究新算法

块算法（Block Algorithm）是 LAPACK 计划中的基本并行化方法。它通过矩阵划分，将算法表示为子矩阵的操作。该方法的性能依赖于块的大小，因此必须研究适合于具体目标机的分块策略，然后开发一个机制，使之能自动地确定最佳的块尺度。

(3) BLAS 的使用

利用线性代数计算固有的矩阵向量特性以及块算法技术，LAPACK 程序设计中力求使尽可能多的计算通过调用基本线代数计算程序包 BLAS1、2 和 3 来实现。BLAS 的使用使得 LAPACK 代码通过这些计算内核的有效实现而“调谐”至给定的结构之上。面向机器的优化局限于这些核心模块，而用户界面则保持一致。

3 并行算法的设计与实现

前已指出，巨型机数学软件分向量计算和并行计算两个层次。目前的大多数巨型机系统，其处理机数目不多，向量运算是提高性能的主要因素，因而向量算法仍是并行算法研究中的主要内容。通过采用扇入法、递归倍增法和循环约化法等技术，尽量开发数学问题中的向量运算成分，使其向量长度充分拟合向量处理结构。在具体实现中，特别注意多功能部件的并发、向量链接、向量寄存器的使用与数据重用等。对于多处理机环境，应进一步开发数学问题的并行性，使子任务的划分与粒度拟合系统的通讯与同步机制，主要的技术手段有块算法、分裂技术和混乱迭代等。同样地，还应特别注意任务的分配、数据的调度以及局部存储器的使用等。

下面就两个简单实例来考察巨型机数学库软件并行算法的设计与实现。

(1) 矩阵乘法的设计与实现

矩阵乘在数值计算问题中频繁出现，是巨型机基本线代数计算子程序包 BLAS 的主要模块之一。设有 $n\times n$ 阶矩阵 A、B，求其乘积矩阵 C. 记 C 的第 i 行为 c_{i*}，第 j 列为 c_{*j}，第 i 行第 j 列的元素为 c_{ij}，对 A、B 亦引入相应记法。依矩阵乘定义，有如下内积表示式

$$C = AB = (a_{i*} \cdot b_{*j})$$

即有伪代码形为

$$\begin{aligned} &\textbf{for } i = 1 \textbf{ to } n \\ &\quad \textbf{for } j = 1 \textbf{ to } n \\ &\qquad c_{ij} = a_{i*} \cdot b_{*j} \end{aligned} \tag{1}$$

对于无相应硬指令的向量计算机如 YH-1，以上向量内积计算需从内存分别按直接顺序和间隔方式向寄存器读出行与列两个向量，作对应单元的向量乘，然后用扇入法完成乘积的求和运算。此时，显然数据调度量大，向量操作次数较多且含有部分标量操作，向量化效率不高。

若将式(1) 改写成向量外积的表示形式

$$C = AB = \sum_{k=1}^{n} (a_{*k} \cdot b_{k*})$$

于是有

$$\begin{aligned} &\textbf{for } k = 1 \textbf{ to } n \\ &\quad \textbf{for } j = 1 \textbf{ to } n \end{aligned} \tag{2}$$

$$c_{*j} = c_{*j} + b_{kj} \cdot a_{*k}$$

或将求和运算放入矩阵内，写为线性组合形式的中积算法

$$C = AB = \left(\sum_{k=1}^{n} b_{k1} a_{*k}, \cdots, \sum_{k=1}^{n} b_{kn} a_{*k} \right)$$

即

for $j = 1$ **to** n

for $k = 1$ **to** n (3)

$$c_{*j} = c_{*j} + b_{kj} \cdot a_{*k}$$

易见算法(2)、(3)的计算内核是向量长度为 n 的列向量乘加链接运算，显然均较内积法(1)向量化程度高，向量操作次数少，与 FORTRAN 数组按列存储相拟合，特别是能充分发挥向量链接特性。

式(2)的内层循环需从内存读出一次 a_{*k}，且有 n 次 c_{*j}的读出、写入和 n 个链接运算，而式(3)则有 n 次 a_{*k}的读出和 n 个链接运算，以及 c_{*j}的读出写入各一次，对于只有一个通道、或读写不可并发的计算机系统，显然中积算法所需数据调度时间要少些。在 YH-1 巨型机上，若用一个向量寄存器装 c_{*j}，用两个向量寄存器交替存放 a_{*k}与 $a_{*,k+1}$，则中积算法(3)的内循环可如下实现：

load $c_{*j} = 0$

load a_{*1}

form $c_{*j} = c_{*j} + b_{1j} a_{*1}$ **by chaining, load** a_{*2}

form $c_{*j} = c_{*j} + b_{2j} a_{*2}$ **by chaining load** a_{*3}

…

从而最大限度地达到多功能部件的并发、链接特性的发挥和寄存器数据的重用。显然以上方案易于用汇编代码实现。若用 FORTRAN 语言编码，则应采用循环展开技术，将相应于(3)的内层循环展开为

for $k = 1$ **by step** 2 **to** n

$$c_{*j} = c_{*j} + b_{kj} a_{*k} + b_{k+1,j} a_{*,k+1}$$

以保证 $a_{*,k+1}$的读出与前一个链接运算并发，且使得 c_{*j}的中间结果的读出与写入次数减半。视实际情况，以上循环展开的深度还可增大以取得最佳效果。

(2) 三对角方程组求解

设有如下三对角方程组

$$\begin{bmatrix} a_1 & c_1 & & & \\ b_2 & a_2 & c_2 & & \\ \ddots & \ddots & \ddots & & \\ & b_{n-1} & a_{n-1} & c_{n-1} & \\ & & b_n & a_n & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix} \tag{4}$$

这里系数矩阵为对角占优的或至少是正定的。经典的串行解法——追赶法为纯粹的递归计算，虽然可采用递归倍增技术开发其并行成分，但效果并不理想。循环约化法或循环奇偶约化法是专门针对此类问题而设计的并行解法，可较好地组织向量计算或大规模并行计算。但对于通常的巨型机来说，该算法未考虑到标量部件的效率，特别是未考虑到存储系统的分级结构，如超高速缓存数据的重用等，从而其加速比也不令人满意。

若将原三对角系统作奇偶置换，生成如图(1)所示的新系统，其中 l 为一任意的偏移参量，写为块形式

$$\begin{bmatrix} A_{11} & A_{12} & 0 & 0 \\ A_{21} & A_{22} & A_{23} & A_{24} \\ 0 & A_{32} & A_{33} & 0 \\ 0 & A_{42} & 0 & A_{44} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{5}$$

从第 3 个块方程形式地解出 X_3：

$$X_3 = (-A_{33}^{-1})(A_{32}X_2 - F_3) \tag{6}$$

并从其它块方程中消去，则得到约化方程组

$$\begin{bmatrix} A_{11} & A_{12} & 0 \\ A_{21} & A_{22}^* & A_{24} \\ 0 & A_{42} & A_{44} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2^* \\ F_4 \end{bmatrix} \tag{7}$$

再次为三对角形的，其中两个修正项 A_{22}^*和 F_2^* 分别为

$$\begin{cases} A_{22} = A_{22}^* - A_{23}A_{33}^{-1}A_{32} \\ F_2^* = F_2 - A_{23}A_{33}^{-1}F_3 \end{cases} \tag{8}$$

上面由(4)至(7)的变换称为一个长度为 k（位移量 l）的局部循环约化（Local Cyclic Reduction）步，而由(6)对 X_3 的计算则称为回代步。在(6)、(8)的计算中，A_{33}是一对角线矩阵，其逆易于求出。

原始的块方程组(5)第一行和第四行的所有块都无须修正，即局部约化步可以在原块方程组分离的部分独立地执行，从而可以在向量多处理机系统上，按处理机台数作相应矩阵划分而构造出有效的并行算法。每一个局部约化步和回代步均可通过对各级 BLAS 子程序的调用而组织向量计算。

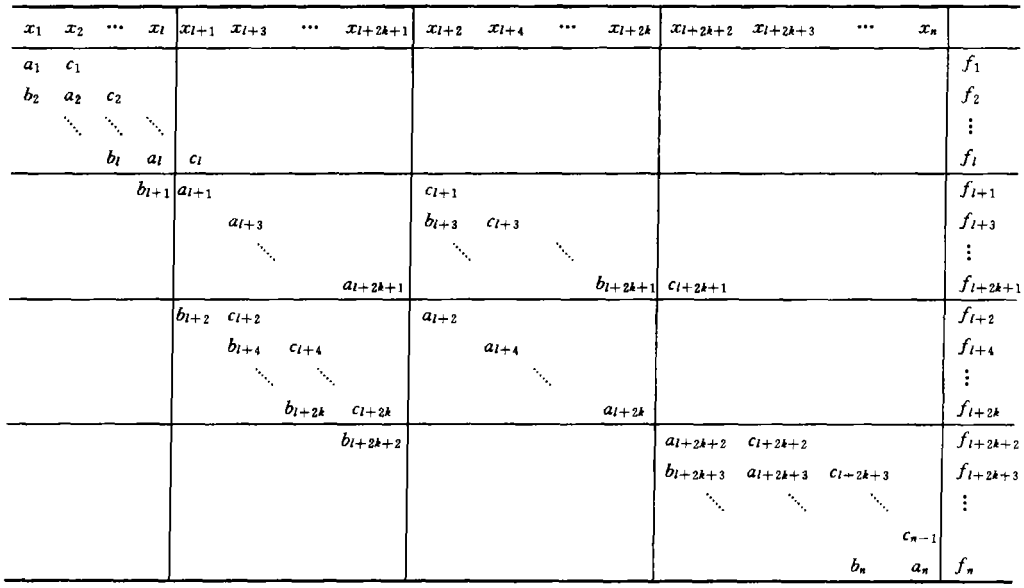

图 1　三对角方程的重新排序

5　结　语

我们在向量巨型机通用数学库软件研制上已有了一些成果和经验，但向量多处理机系统的并行向量数学软件开发工作刚刚起步。本文仅就某些方面进行了粗浅的讨论，期望在今后的研究实践中对其中的关键技术、特别是并行算法的设计思想和实现方法作系统的深化与提高，在此基础上，进一步展开并行计算数学软件开发环境的研究。

参 考 文 献

1 张绮霞．数学软件与解题环境．软件产业，1990，(3)：1～9

2 赛贤福，李晓梅，谢铁柱．同步并行算法．国防科技大学出版社，1986

3 李晓梅．颜宝勇．研制高效向量通用数学程序库方法．软件产业，1990，(4)：9～14

4 何新芳，胡庆丰，李忠．银河机向量线性代数库的并行算法研究及优化．国防科技大学学报，1989，(4)：60～64

5 Herman J J. te Riele. Application of Supercomputers in Mathematics, in Scientific Computing on Supercompufters

6 Ortega J M. Introduction to Parallel and Vector Solution of Linear Systems. Plenum Press, 1988

7 Bischof C H, Dongarra J J. A Linear Algebr Library for High－Performance Computers. in Parallel Supercomputing: Method, Algorithm and Application. edited by G F Carey, John Wiley & Sons, 1989: 45～56

8 Reuter R, Zecca V. A Parallel Method for Solving Tridiagonal Systems of Linear Equation on the IBM 3090 Vector Multiprocessor. in High Performance Computing, edited by J L Delhay and E Gelenbe. Elsevier Science Publishers, B V, 1989

Supercomputer Common－Mathematical－Library－Software and Parallel Algorithms

Hu Qingfeng Li Xiaomei

(Department of Computer Science)

Abstract

The concept and significance of the common－mathematical－library－software for supercomputers are presented first, and then its developing ways and trends are pointed out, the key effects of parallel algorithms in its development are discussed in special, and finally the design and implementation of parallel algorithms are analysed with some examples, methematical software

Key words supercomputers, scientific programs, parallel algorithms, mathematical software