构造面向多种MIMD结构的并行算法库*

王 晨 陈德来 倪 南 张德富

(南京大学计算机科学与技术系 南京 210093)

摘要 本文描述了NPAB—1并行算法库的结构和功能,并通过两个例子说明了库中并行算法的设计风格.

关键词 并行算法库,MIMD,可扩充性,调用接口,查询.

局部存储的MIMD系统因能花费较小的代价获取较高的Mflops而越来越引起人们的兴趣,但使用这种系统并不是件容易的事.这是由于自动并行编译技术还没有成熟到实用阶段,至少暂时还不可能象写串行程序一样写并行程序;程序员因而不得不在怎样将几乎无穷无尽的并行算法与千差万别的拓扑结构相匹配上花费大量精力.解决这一问题的最简单方法是构造一个并行算法库或一组可重用的子程序集.这类算法库在向量处理机上有BLAS,LAPACK[1],在单TRANSPUTER上有NAG串行FORTRAN库,NAG串行图形库,FOLIB,Liverpool底层库[2]等.面向MIMD模型的并行算法库目前尚不多见,其中有Liverpool大学正在研制的Mark 1 Library.

我们在TRANSPUTER阵列(TSB98—9)上设计的NPAB—1并行算法库具有以下特点:(1)库中不仅存放了并行算法,而且还存有与算法所对应的系统拓扑结构以及它们之间的映射关系.(2)提供了丰富友好的用户接口,用户可从宿主机或通过网络调用算法库.(3)对并行处理系统拓扑结构设计颇有辅助作用.(4)开放性好,在库中可不断扩充并行算法及其拓扑结构.(5)算法在MIMD模型上具有可移植性.(6)算法本身的可扩充性.

1 NPAB—1的结构模型

NPAB—1并行算法库由3部分组成:算法集管理层、调用接口层、算法集,其关系如图1所示.

算法集中包含了提供给TRANSPUTER运行的常用子程序,如矩阵乘法,多项式求值,DFT,FFT,解线性方程组,解常微分方程以及排序等.每一组算法对应着多种拓扑结构.

调用接口层提供了在宿主机上由C语言调用算法集中子程序的方法,包含一个可拆卸的调用接口库invoke.lib,应用程序员只要连上这个库便可使用算法集中算法.调用方法对

---

* 作者**王晨**,1971年生,硕士生,主要研究领域为并行处理.**陈德来**,1965年生,博士生,主要研究领域为并行处理.**倪南**,1969年生,硕士生,主要研究领域为并行处理.**张德富**,1937年生,教授,主要研究领域为并行处理与分布式计算.

本文通讯联系人:王晨,南京210093,南京大学计算机科学与技术系

本文1994—06—27收到,1994—12—03定稿

其它语言是可扩充的,并且可以管理通过网络来的访问.

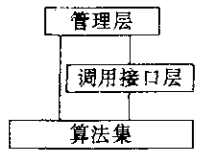

图1

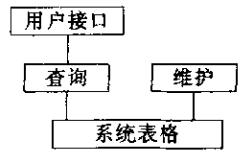

图2

与串行算法库不同,管理层的作用比较特殊.因为并行算法跟结构紧密相关,用户在使用库时还需对其中固有拓扑结构、问题规模有所了解,管理层从多方面为用户提供了这些查询;另外它还支持库的可扩充性,具有对系统表格的维护功能.管理层的功能如图 2 所示.

用户接口接收来自用户的查询请求,然后对用户请求进行分析.用户查询可分为 3 种:

(1)已知欲求解的问题和现有并行处理计算机的拓扑结构,要查询哪一类算法在该机上解此问题是高效的.

(2)已知并行处理计算机的拓扑结构,欲查询该结构适于解哪类问题.

(3)依照欲求解的问题和并行算法查找相应的并行处理计算机的拓扑结构.

查询的作用是将查询请求转化为算法名、调用格式返回用户.

系统表格中保存了算法集的所有信息,表格的每一项是每个算法所对应的属性,如算法名、参数表、系统拓扑结构、算法规模、算法的性能评价等.查询建立在系统表格之上.表格的数据结构如表 1 所示.

表 1

算法名	拓扑结构	规模	性能	参数表	相关算法

其中,算法名由 P_开头的串唯一确定;拓扑结构保存该拓扑结构在 topu.inf 文件中的偏移值,topu.inf 包含了处理机间的相联矩阵;规模为该算法所能承担的最大数据量;性能保存了存放该算法加速比、效率参数曲线或表格的文件名;参数表存放表示调用格式的字符串;而相关算法指向同一类型算法中下一算法名的相对于系统表格的偏移值,最后一个同类算法的相关算法项填入表格中同类第一个算法的地址,这样同一类算法在系统表格中表现为一个单环,增加了查询的速度.

维护模块根据算法集中的变化调整系统表格,支持系统的可扩充性.

2　算法示例

NPAB—1 并行算法库中含有数值和非数值两类算法,算法用并行 C 实现.下面以两个算法为例说明并行算法库的使用.

(1)矩阵相乘的 Strassen 算法

Strassen 算法的基本思想是利用分块矩阵进行运算,减少子矩阵的乘法次数.设

$$A=\begin{bmatrix} A11 & A12 \\ A21 & A22 \end{bmatrix} \qquad B=\begin{bmatrix} B11 & B12 \\ B21 & B22 \end{bmatrix} \qquad C=A\times B$$

其中 A、B 为 $n*n$ 阶矩阵,Aij,$Bij(i=1,2;j=1,2)$为$(n/2*n/2)$阶矩阵(若 n 为奇数,则将 A、B 扩充为 $n+1$ 行,$n+1$ 列 A'、B',扩充的行列用 0 填充).按传统算法,矩阵乘法有

8次子矩阵乘法和4次子矩阵加法.而Strassen算法的计算方案为

$M1=(A12-A22)(B21+B22)$

$M2=(A11+A22)(B11+B22)$

$M3=(A11-A21)(B11+B12)$

$M4=(A11+A12)B22$

$M5=A11(B12-B22)$

$M6=A22(B21-B11)$

$M7=(A21+A22)B11$

$C11=M1+M2-M4+M6$

$C12=M4+M5$

$C21=M6+M7$

$C22=M2-M3+M5-M7$

共需7次子矩阵乘法和18次子矩阵加法.减少了一次子矩阵乘法,增加了14次子矩阵加法.因为子矩阵块是$(n/2)*(n/2)$的,乘法的运算量为$O(n^3)$,加法的运算量是$O(n^2)$,易知当$n>14$时,Strassen算法的计算量$T(n)$小于一般算法.

$$T(n)=7T(n/2)^3+18(n/2)^2$$

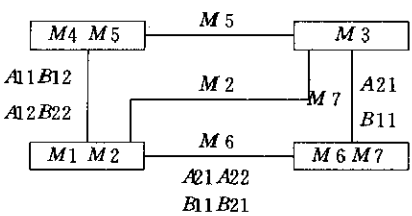

图3

设$n=2^k$,上式为

$$T(n)=7^{\log_2 n}*T(1)+\Sigma 7^{i-1}*18(n/2^i)^2$$
$$=7^{\log_2 n}*T(1)+(18/7)n^2\Sigma(7/4)^i$$
$$\leqslant 7^{\log_2 n}*T(1)+6*7^{\log_2 n}$$

$$7^{\log_2 n}=n^{\log_2 7}\approx n^{2.81}$$

$$T(n)=O(n^{2.81})$$

该算法在图3所示拓扑结构的并行机上运行.

设传输一个$n/2$阶的矩阵所需时间为$comm(n/2)$,进行$n/2$阶加法和乘法所需的时间分别为$add(n/2)$,$mul(n/2)$.当$comm(n/2)<<add(n/2)$,$comm(n/2)<<mul(n/2)$时,并行计算时间和通信时间之和为

$$10comm(n/2)+7add(n/2)+2mul(n/2)$$
$$=10comm(n/2)+(n^3/4+7n^2/4)u+n^3v/4$$

则加速比S为

$$S=\frac{n^3(u+v)}{10comm(n/2)+(n^3+7n^2)u/4+n^3v/4}\approx 4$$

这个实现具有扩充性,可以将计算$Mi(i=1,\ldots,7)$中矩阵乘法按Strassen算法继续分解,而拓扑结构图中每一结点细化为4个处理机组成的子集.

用M_i^n表示第n次细化后某个方格上的Mi值$(i=1,\ldots,7)$,如$M_1^0=M_1$;用A_i^n、B_i^n表示第n次细化后待相乘的矩阵,

$$M_i^n=A_i^{n+1}*B_i^{n+1};$$

易知$A_1^1=A_{12}^0-A_{22}^0=A_{12}-A_{22}$,$B_1^1=B_{21}^0+B_{22}^0=B_{21}+B_{22}$

而$M_i^n=A_i^{n+1}*B_i^{n+1}$,$M_i^{n-1}=A_i^n*B_i^n$

$A_1^{n+1}=((A_i^n)_{12}-(A_i^n)_{22})$,$B_1^{n+1}=((B_i^n)_{21}+(B_i^n)_{22})$

$A_2^{n+1}=((A_i^n)_{11}+(A_i^n)_{22}),B_2^{n+1}=((B_i^n)_{11}+(B_i^n)_{22})$

$A_3^{n+1}=((A_i^n)_{11}-(A_i^n)_{21}),B_3^{n+1}=((B_i^n)_{11}+(B_i^n)_{12})$

$A_4^{n+1}=((A_i^n)_{11}+(A_i^n)_{12}),B_4^{n+1}=(B_i^n)_{22}$

$A_5^{n+1}=(A_i^n)_{11},B_5^{n+1}=((B_i^n)_{12}-(B_i^n)_{22})$

$A_6^{n+1}=(A_i^n)_{22},B_6^{n+1}=((B_i^n)_{21}-(B_i^n)_{11})$

$A_7^{n+1}=((A_i^n)_{21}+(A_i^n)_{22}),B_7^{n+1}=(B_i^n)_{22}$

这是各结点上数据分布的递推式，只需将初始矩阵增广成 2^k 阶，便可在各方格上求出数据分布.

应用程序员在需要做矩阵乘法运算时，通过我们的查询模块可以了解到诸如流水线、Strassen 等算法的性能、对应的拓扑结构，并根据他对自己问题规模的知识选择一种实现. 对于形如 $C=A\times B$ 的矩阵乘法，4 个处理机的 Strassen 算法调用格式如下：

int P_Strassen_4(double *A,double *B,double *C,long dimension);

Remark: return −1 indicate error.

(2)树机排序

Distributed−memory 并行模型上的排序一般都因为通信开销太大而不能取得令人满意的加速比. 相对来说将处理机联成树形，网络直径较小($2\log_2 n$)，通信开销也较低. 我们将 7 个处理机连成如图 4 所示的二叉树.

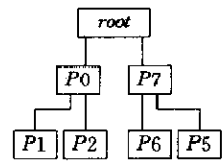

图4

算法描述：

①划分：根处理机(*root*)将待排序的数据分为两等分，分别向左、右子女 P_{left}、P_{right} 发送(这里是 P_0, P_7)，若 P_{left}、$P_{right}<n-1$ 层，则在 P_{left}、P_{right} 上重复 *root* 的动作；若 P_{left}、$P_{right}=n-1$ 层，则将收到的数分成三等分，分别放在本处理机和左右子女上.

②排序：在树的 $n-1$ 层和 n 层用 *Quicksort* 并行排序.

③归并收集：叶结点排序完成后向父母结点发回结果，$n-1$ 层收到子女的信后即进行归并(三路归并)，结果发往上层，如此重复，最终在 *root* 上获得结果.

采用此算法的加速比为

$$S=\frac{\mu n\log_2 n}{\mu(n/6)\log_2(n/6)+\nu*(n/2)+2comm(n/2)+2comm(n/6)}\approx 4$$

其中 $\mu n\log_2 n$ 为快速排序的时间复杂度，$comm(k)$表示传送数据量 k 花费的时间.

对于处理机数较多的树结构，上述算法可进一步优化.，

设 D_i 为第 i 层收到的总数据量，$F(D_i)$表示完成第 i 层排序的时间，$f(D_i)$表示在一个结点上将 D_i 个数排序的时间，$k_i=D_{i+1}/D_i$，Mi 表示在第 i 层进行归并的时间，则

$$F(D_i)=f(k_iD_i)+M_i$$

理想的 k_i 应满足使 $f(k_iD_i)=F(D_{i+1})$，充分发挥并行性. 而

$$\begin{aligned}F(D_{i+1})&=f(k_{i+1}*(1-k_i)D_i/2)+M_{i+1}\\&=F(D_{i+2})+M_{i+1}\\&=f(k_{i+2}D_{i+2})+M_{i+2}+M_{i+1}\\&=f(k_{i+2}*(1-k_{i+1})D_{i+1}/2)+M_{i+2}+M_{i+1}\end{aligned}$$

于是 $$f(k_{i+1}D_{i+1})=f(k_{i+2}*(1-k_{i+1})D_{i+1}/2)+M_{i+2}$$

而 $M_{i+2}=O(D_{i+2})$，D_{i+2}很大时 M_{i+2}的影响较小，可忽略. 因此只要知道 k_{i+2}，便可求出 k_{i+1}. 而易知 $k_{n-1}=1/3$，$(Mn=0)$，以后逐次往上迭代便可求得 k_i 的值.

优化后的算法如下：①计算划分因子序列$\{k_i\}$. ②划分，依据 k_i 进行划分、发送直至叶结点. ③并行快速排序. ④收集归并. 这是一种适应树结构的算法.

同样，应用程序员在做排序时，通过查询可了解到库中多机归并排序、树机排序等算法所对应的拓扑结构，加速比和效率随数据量的变化曲线，根据自己的处理机数、能形成的拓扑结构及数据量选择相应的调用形式. 3个处理机和7个处理机的树机排序调用形式如下：

```
int P_Sort_TMerge_3(long * data,long data_len);
int P_Sort_TMerge_7(long * data,long data_len);
Remark:Return -1 indicate error.
```

3 结束语

并行算法库是填平程序员与复杂多变的并行结构之间沟壑的一种方式，它从某种程度上把并行处理机的使用者从大量繁琐的工作中解脱出来，使程序员不必去为一个算法的实现费尽心机. 构造结构适应的并行算法库是使并行处理机实现商用化的重要方面.

参考文献

1 Christian H B, Jack J D. A linear algebra library for high performance computers. In: Carey Grarham F ed. Parallel Supercomputing: Methods, Algorithms and Applications, Chichester: John Wiley & Sons, 1989. 45～56.

2 Delves L. Library software for transputer arrays. In: Perrott R H ed. Software for Parallel Computers, London: Chapman & Hall, 1992. 245～254.

CONSTRUCTING VARIOUS ARCHITECTURE OF MIMD ORIENTED LIBRARY OF PARALLEL ALGORITHMS

Wang Chen Chen Delai Ni Nan Zhang Defu

(*Department of Computer Science and Technology Nanjing University Nanjing* 210093)

Abstract This paper described the architecture and functions of NPAB—1, a parallel algorithm's library. It explained the style of designing these parallel algorithms with two examples as well.

Key words Parallel algorithm library, MIMD, extensible, interface of invoking, query.