基于网络的数据并行计算中分布数组描述研究*

胡长军[1], 丁文魁[2], 黄其军[2] 向 华[2], 许卓群[2]

[1](清华大学 计算机科学与技术系,北京 100084);
[2](北京大学 计算机科学技术系,北京 100871)
E-mail: huchangjun@tsinghua.edu.cn
http://www.cs.tsinghua.edu.cn

摘要: 如何描述分布数组是基于网络数据并行计算的基本问题.从网络并行计算的一般需求出发,讨论了分布数组描述DAD(distributed array descriptor)的内容和结构,具体给出了p-HPF并行编译系统的DAD结构定义.针对分布数据的稀疏存储和紧凑存储模型,给出了数据在Block分布、Cyclic分布和Block_Cyclic(*k*)分布方式下,全局数组到局部数组转换的计算方法,这些方法已在p-HPF编译器中得到实现并证明了其有效性.最后讨论了分布数据描述的标准化对实现并行计算系统的可移植性和可重用性的意义.

关 键 词: 分布数组描述;网络并行计算;HPF语言;并行编译
中图法分类号: TP311 文献标识码: A

无论用何种方式实现基于网络的数据并行计算,都要解决3个层次的问题.第1,将全局数据以适当方式在各结点上进行分布,并按一定规则将串行程序划分成可在多个结点上并行执行的进程;第2,为保证程序正确性和高效性而进行通信检测和调度,确定通信类型、时机、方式等;第3,将可并行执行的多个进程映射到具体体系结构上去执行.我们可以通过高级程序设计语言,如F77和网络通信最基本的协议如TCP/IP实现网络数据并行计算,这需要程序员处理数据划分、通信等细节,也可以通过高级语言调用一些并行计算通信服务类库如MPI,PVM等实现,程序员只负责数据划分和通信安排,具体实现通过接口交由类库完成,但编程仍很繁琐.进入20世纪90年代以来,为了提高并行程序的可扩展性和可移植性,出现了以HPF(high performance Fortran)[1]为代表的高级数据并行程序设计语言,它在语言级提供数据分布指导语句,通过并行编译系统自动处理3个层次的问题,实现计算模型和具体计算体系结构的分离,使网络并行计算编程简单化.20世纪90年代中后期以来出现了一些典型的HPF并行编译系统,如Fortran D[2],Vienna Fortran[3]以及我们开发的p-HPF[4,5]等.近年来,综合运用各种并行模式(HPF,F77+MPI,...)解决大规模并行应用实际问题的研究引起了广泛重视,也出现了一些应用算法库如ScaLAPACK[6]等.综观HPF编译器、高级语言(C,FORTRAN等)+通信库(MPI,PVM等)以及并行应用算法库(ScaLAPACK,LAPACK等)等几种网络数据并行计算方式的实现,一个共同的策略就是将系统实现分为两层,底层是运行支持系统,高层是编译或处理系统.编译或处理部分用来解决数据分布、数据及计算的相关性分析、通信的检测调度等问题,产生出含有运行支持系统调用语句并可在各结点并行执行的结点程序;在各结点程序并行执行时,运行支持系统根据结点程序中的调用语句,解决结点间各种类型的数据通信、数据分布的动态调整等问题.实际上,通过运行支持策略实现网络数据并行计算已经获得了广泛的认可.一些实用的运行支持系统

* 收稿日期: 2000-03-16; 修改日期: 2000-09-25
基金项目: 国家自然科学基金资助项目(60173004);国家863高科技发展计划资助项目(863-306-ZT01-02-3)
作者简介: 胡长军(1963－),男,河北沧县人,博士,教授,主要研究领域为并行计算,并行编译,并行软件工程,并行应用系统,软件体系结构,数据处理;丁文魁(1946－),男,河南兰考人,副教授,主要研究领域为并行计算,并行编译技术;黄其军(1973－),男,安徽肥东人,博士生,主要研究领域为并行计算,并行编译技术,大规模并行应用系统开发技术;向华(1973－),女,上海人,博士,主要研究领域为并行计算,并行编译技术,分布式系统;许卓群(1936－),男,河南乳山人,教授,博士生导师,主要研究领域为并行计算,并行编译,分布式信息系统,电子商务,可视化系统.

如 Adlib,Chaos 等已获得广泛应用.这种实现策略的一个核心问题是,上层的编译或处理部分和底层运行支持的接口问题,上层设计者处理的是全局数据,而运行支持部分涉及的是分布在各接点上的局部数据.这就需要两者之间有一个协议,实现全局与局部之间信息的相互转换.协议内容主要是对关于数组分布信息的描述,将这些信息收集并以适当的形式进行组织就形成了分布数组描述子 DAD(distribute array descriptor).可见,DAD 不仅是并行编译系统设计的核心数据结构,而且也是实现网络数据并行计算所必须涉及的关键问题.本文首先结合 p_HPF 并行编译系统的设计与实现,研究分布数组描述的内容及组织原则,具体给出了 p-HPF 采用的 DAD 结构,并与 ScaLAPACK 的 DAD 进行了对比.然后针对分布数组稀疏和紧凑存储的模型,重点给出了数组在 Block,Cyclic 及一般分布形式 Block_Cyclic(*k*)情况下,DAD 局部元素的计算方法;最后论述了建立 DAD 结构规范的重要性.

1 DAD 的内容确定和组织

DAD 作为全局数组和局部数据存储之间的接口,其内容和组织方式与组织和网络并行计算的具体需求相关.

1.1 DAD的内容确定

DAD 用来描述数据的分布信息,它的内容首先取决于它所起的作用,我们先看一个用 FORTRAN D 实现数据并行计算的一个例子,如图 1 所示.

```
Processors P(p)
Real x(l:u)
Distribute x(block_cyclic(k))
Do i=l,u
    x(i)=F(i)
Enddo
```

(a) A fragment of HPF program

(a) HPF 源程序片段

```
Real x(1:ceiling((u–l)/p))
lb$=LowerLoopBound(l)
ub$=UpperLoopBound(u)
Do i=lb$,ub$
    if (owner(x(i))
  i$=GlobalLoopIndex(i,p)
    x(i)=F(i$)
    End if
Enddo
```

(b) Node program

(b) 结点程序

Fig.1 The fragment of a HPF program and its SPMD code

图 1 HPF 的源程序片段及其 SPMD 结点程序代码

在图 1(a)中,定义了一个数组 *x*,并将 *x* 以 Block-Cyclic(*k*)的方式分布到 *p* 个处理器上,在图 1(b)中,首先定义了一个数组 *x*,它的下界为 1,而上界由全局数组的上下界和处理器个数及分布参数 *b* 所确定.其次,通过计算 LowerLoopBound(*l*)和 UpperLoopBound(*u*)这两个运行支持函数计算出本结点程序的循环上下界,即根据循环的全局量计算每个结点循环的局部量.接着按照拥有者计算原则为 *x*(*i*) 赋值.值得注意的是,在源程序中,函数 *F*(*i*)中的子变量 *I* 与结点程序中的 *i* 是不一样的,在结点程序中也要通过运行支持函数 GlobalLoopIndex(*i*,*p*)找到局部索引 *i* 所对应的局部下标 *i*$,用它计算 *F*(*i*$).从例 1 可以看出,运行支持函数要依靠数组全局信息完成对局部信息的计算,实现全局和局部的相互转换.从网络并行计算的一般需求来分析,实现每一个结点局部信息的计算,DAD 应提供如下全局信息:

(1) 数组整体信息:包括数组名称、行数、列数、元素类型、类型长度、数组类型等.

(2) 数组分布信息:包括分布类型、块分布时块的大小、循环分布时的循环单位等.由于数组的分布是按维进行的,所以分布信息也需要以维为单位进行组织.

(3) 处理器组织信息:处理器的个数、布局方式等信息.

(4) 分布方式信息:包括数据分布开始的处理器号、数据分布的处理器顺序、参与分配的处理器组的组成、是否通过对齐模板(align template)进行分布等.

(5) 对齐模板信息:如果通过对齐模板进行数据分布,还要记录模板的上下界、对齐计算公式参数等.

(6) DAD 自说明信息:包括本 DAD 的名称、类型等.

这些信息对于编译系统来说,可直接从源程序说明语句中得到.对于其他网络数据并行计算实现方式,由程序员直接写入.

从便于局部信息向全局信息的转换和局部信息的使用角度来看,DAD 还应记录由全局信息计算出来的每一个结点的局部信息,至少包括:

(1) 数组每维信息:该结点数组的上界、下界、步长等信息.

(2) 对齐模板分布每维信息:模板的上界、下界、步长等信息.

(3) 其他信息:如用于通信优化的内存区域指针等.

DAD 的这些内容,是针对网络数据并行计算的一般需求确定的,有了这些信息就可以实现多种形式的局部信息和全局信息的转换了.当然,对于一个具体的系统, 这些信息并不是都需要的.

1.2 DAD的组织

从方便系统使用的角度出发,不同的系统采用了不同的 DAD 结构.如对于并行编译系统的设计、结点程序的生成和通信检测等都是以对局部数据的操作为主的,因此,它的 DAD 设计要考虑使局部信息的使用最为方便,而对于一些数据并行算法库,虽然它也都带有运行支持系统,但算法库的使用对象是应用程序员,要由程序员创建 DAD,为了编程方便,它的 DAD 结构要强调全局信息,尽量少暴露局部信息.这一原则在我们设计的 p-HPF 并行编译器的 DAD 结构和 ScaLAPACK 算法库的 DAD 结构中得到了充分体现,p-HPF 采用的 DAD 结构如下:

```
struct DAD {
  Status status;          //记录数组的状态,表明是核内或核外
  int icla_size;          //若为核外数组,每次引导到内存中的最大尺寸
  int major;              //数组的行或列优先标志
  const int rank;         //数组维数
  Group group;            //记录处理器组信息
  Range* ranges;          //记录数组每维的分布信息
  Stride* strides;        //记录每维的分布方式信息
  }
struct pcrcfDAD:public DAD {
   Range rngs[8];                 //用来记录数组每一维的分布信息
   Stride strs[8];                //数组每维的跳步信息
   int elementLen;                //数组元素的长度
   int elementType;               //数组元素的类型
   int size;
   void* memdata;
   int memHandle;                 //指向用于内存分配的栈首址
   PFILE *pfile;                  //用于存储核外数组的文件指针
   void* data;                    //数组的第一个有效数据指针
   int grpHandle;                 //处理器组句柄,指向用于数据分布的处理器组
   int rngHandles[8];
};
```

这里定义了两个结构,DAD 结构面向 FORTRAN 接口函数设计使用,pcrcf_DAD 为 C 支持函数使用.两个 DAD 结构包含了数组整体信息、数组分布信息、处理器组信息、分布方式信息等,突出特点是,为了便于编译

器设计的使用,将局部信息组成了 Range,Stride 等专门的类,并设计了专门的成员函数.与 p-HPF 不同,在 ScaLAPACK 中,DAD 是用数组表达的,DAD 的命名规则是 DESC_数组名,如数组 *A* 的 DAD 为 DESC_*A*.数组的各项内容见表 1[6].

Table 1 The DAD Contents of ScaLAPACK

表 **1** ScaLAPACK 的 DAD 内容

DESC_()	Symbolic name①	Scope②	Definition③
1	DTYPE	Global	Descriptor type. DTYPE=1 for dense matrices④
2	CTXT	Global	Context handle, indicating the process grid over which the global matrices is distributed⑤
3	M	Global	Number of rows in the global array⑥
4	N	Global	Number of columns of global array⑦
5	MB	Global	Blocking factor used to distribute the rows of the array⑧
6	NB	Global	Blocking factor used to distribute the rows of the array⑨
7	RSRC	Global	Process row number which the first row of the array is distributed⑩
8	CSRC	Global	Process column number which the first row of the array is distributed⑪
9	LLD	Local	Leading dimension of the local array⑫

①模板名称,②范围,③定义,④数组类型的描述,DTYPE 等于 1 描述致密矩阵,⑤上下文句柄,
标识全局数组分布的进程网格⑥矩阵的行数,⑦矩阵列数,⑧行分布块因子,⑨列分布单位,
⑩行数据开始分布的进程号,⑪列数据开始分布的进程号,⑫局部数组中最大维的长度.

从表 1 可以看出,DAD 展现在用户面前的大多是全局信息,包括数组信息、数组分布信息、分布方式信息、处理器组信息等,它没有使用对齐模板,因此不包括模板信息.其突出特点是提供了几种稀疏类型矩阵(对角阵,带状阵)的描述,从而大大提高了算法库的实用性.

2 DAD 局部信息的计算

如何将数据分布到各个处理结点上以及在不同数据分布方式下局部信息的计算是网络数据并行计算的一个基本问题,局部信息是进行结点程序设计的基础.它的计算与分布方式与结点存储数据的模型密切相关,最常用的分布方式有 Block,cyclic 方式,而存储模型包括稀疏型和致密型两种.本节以 p-HPF 编译系统为例,给出采用稀疏存储模型时 DAD 局部信息的一种计算方法和采用紧凑存储模型时一般数据分布方式为 Block_Cyclic(k) 时的计算方法,并给出计算实例.

2.1 稀疏存储的局部信息的计算

本节首先讨论带对齐模板且接点存储为稀疏方式时全局地址和局部地址的转换方法,然后讨论一般分布时局部信息的计算方法.

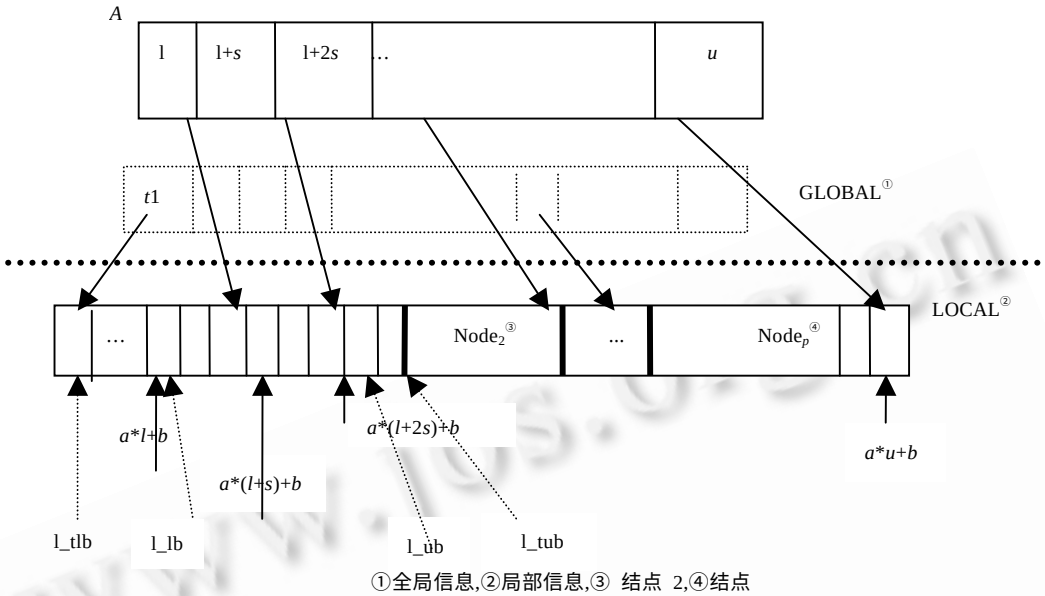
2.1.1 带有对齐模板的分布数据稀疏存储模型

下面是一个用 p-HPF 程序片段描述的数组 *A* 的分布信息和对齐信息.

```
 Real A(l:u:s)
//l,u,s 为整数,u>l,s>0//
 HPF$ processors P(p)
 //声明了 p 个处理器,一维排列//
 HPF$ Template T(t1:t2)
//声明对齐模板//
 HPF$ Align A(a*i+b) with T(i)
  //声明数组和模板的对齐关系//
 HPF$ Distribute (block) onto P::T
//声明将 T 以 Block 方式分布在 P 上//
```

图 2 是数组 *A* 通过模板对齐后分布到各接点上的稀疏存储模型.从图 2 中可以看到,数组 *A* 经对齐模板 *T* 分布到各接点上并不是连续存储的.对齐模板 *T* 并不分配物理空间,它只影响数组元素分布到接点的存储位置.

通过对齐参数 a,b,可调整数组 A 元素在局部结点的存储位置,这有利于我们改善数组访问的局部性,减少通信,提高计算性能.



①全局信息,②局部信息,③ 结点 2,④结点

Fig.2 Sparse store forms for distributed arrays

图 2 分布数组的稀疏存储

2.1.2 基于稀疏存储模型的局部信息计算

设 g_lb,g_ub,step 表示分布数组的全局下界、上界和步长,t_lb t_ub 表示对齐模板的上界和下界,offset stride 表示对齐多项式的系数,discode 表示分布方式,discode=1、2 分别表示 block,cyclic 分布,处理器个数为 p.我们的任务是根据这些全局信息,计算:(1) 对齐模板在每一个处理器上的下界和上界:l_tlb,l_tub,(2) 每一个结点上的第 1 个有效元素的下标 l_lb 和最后一个有效元素的下标 l_ub,即第 1 个有数据的内存地址和最后一个有数据的内存地址.结点 i=1,2,…,p 上的局部信息 l_lb_i l_ub_i l_tlb_i l_tub_i 的计算算法见算法 1.

算法 **1**. 数据在接点稀疏存储时的局部信息计算

case (discode=block) and (MOD((t_ub−t_lb+1),p)=0) //块分布,各结点上元素数相同

q=(t_ub-t_lb+1)/p //各结点上的元素个数

l_tlb_i=t_lb+(i−1)*q (i=1,2,…,p)

l_tub_i=t_lb+i*q−1 (i=1,2,…,p) //计算模版在各结点上的下界和上界

if ((g_lb*stride+b)>=t_lb) and (g_ub*stride+b)<=t_ub)) then

对每一个 l_tlb_i,在[g_lb,g_ub]中找满足 stride*x_i+b>=l_tlb_i 最小的 x_i

l_lb_i=(stride*x_i+b)−l_tlb_i+1

对每一个 l_tub_i,在[g_lb,g_ub]中找满足 stride*y_i+b<=l_tub_i 最大的 y_i

l_ub_i=(stride*x_i+b)□l_tlb_i+1 //计算每个结点上第一个和最后一个有效元素

endif

case (discode=block) and (MOD((t_ub−t_lb+1),p)<>0) //块分布,各结点上元素数不同

q= floor((t_ub−t_lb+1)/p)

l_tlb_i=t_lb+(i−1)*q (i=1,2,…,p−1)

l_tub_i=t_lb+i*q−1 (i=1,2,…,p−1) //计算模版在 1,2,…,p−1 个结点上的上下界

l_tlb_i=t_lb+(i−1)*q (i=p)

l_tub_i=t_ub (i=p) //计算模板在 p 结点上的上下界

l_lb,l_ub 的计算同上一种情况.

case (discode=cyclic) and MOD((t_ub−t_lb+1),p)=0 //循环分布,结点上元素数相同

l_tlb$_i$=t_lb+i−1 (i=1,2,…,p)

l_tub$_i$=t_ub−(p−i) (i=1,2,…,p) //计算模板在各结点上的上下界

o=stride*g_lb+offset−t_lb+1

if MOD(GCD(stride,p),ABS(i−o))=0 then

找出所有满足方程 o+stride*m_i=i+n_i*p 且 n_i>=0, i+n_i*p<=t_ub−t_lb+1,

0=<m_i<(g_lb−g_ub)/step m_i<>n_i 的数对(m_i,n_i)

if 存在这样的数对 then

l_lb$_i$=MIN(n_i)+1

l_ub$_i$=MAX(n_i)+1 //计算各结点上的第 1 个和最后一个有效元素

else

没有有效元素分布在该结点上,置 l_lb$_i$=0,l_ub$_i$=−1.

End if

else

没有有效元素分布在该结点上,置 l_lb$_i$=0,l_ub$_i$=−1.

endif

case (discode=cyclic) and MOD((t_ub−t_lb+1),p)<>0 //循环分布各结点上元素数不同

q=floor((t_ub−t_lb+1),p) r=(t_ub−t_lb+1)−p*q

l_tlb$_i$=t_lb+i−1 (i=1,2,…,p) //模板在各结点的下界.

l_tub$_i$=t_ub−(r−i) (i=1,2,..r) //模板在 1,2,…,r 结点的上界.

l_tub$_i$=(t_ub−r)−(p−i) (i=r+1,r+2,…,p) //模板在 r+1,r+2,…,p 结点的上界.

l_lb$_i$ 和 l_ub$_i$ 的计算和上一种 cyclic 的情况一样.计算实例如下:

把一个数组 a[1:20:1],按 Cyclic 的顺序,通过对齐模版 t[1:60]和对齐多项式 2*i+1 分布到 p=4 个处理器上.此时全局信息为

p=4,g_lb=1,g_ub=20,t_lb=1,t_ub=60 stride=2,

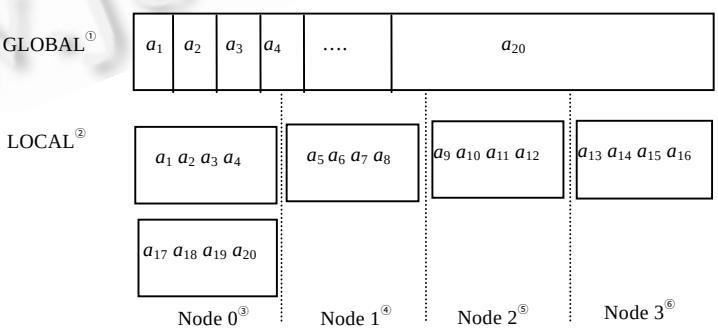
offset=1,discode=2 (cyclic).

根据以上算法计算可得如下结果:

结点	l_tlb	l_tub	l_lb	l_ub
1	1	57	2	11
2	2	58	0	-1
3	3	59	1	10
4	4	60	0	-1

2.2 数据分布为**Block_Cyclic(*k*)**时局部信息的计算

Block_Cyclick(k)数据分布是一种一般的数据分布形式,如 Block_Cyclic(l)就是上面介绍的 Cyclic 分布,而当 k=块分布时的块长时,就是 Block 分布了.图 1(a)中给出了一个采用 Block_Cyclic(k)分布的程序,我们采用紧凑型存储方式,即局部数据连续存储在数组中(如图 3 所示),以此为例,给出图 1(b)结点程序中局部参数的计算方法如下.这个例子中没有使用对齐模板.



①全局信息,②局部信息,③结点 0,④结点 1,⑤结点 2,⑥结点 3.

Fig.3 Example of Block_Cyclic(k) distribution

图 3 Block_Cyclic(k)分布例

Ower($X(i)$=floor((i–l)/k) MOD p //计算 $x(i)$所在的处理器

LowerLoopBound(l_i,p)= //局部循环的下界,即 p 上的第 1 个有效元素

l_1=l_i–l

if (p<floor(l_1/k) MOD p) then return ceiling(l_1/(k*p))*k+1

else if (p>floor(l_1/k) MOD p) then return floor(l_1/(k*p))*k+1

else return floor(l_1/(k*p))*k+(l_1 MOD k)+1

endif

endif

UpperLoopBound(u_i,p)= //局部循环的上界,即 p 上的最后一个有效元素

l_1=u_i–l

if (p<floor(u_1/k) MOD p) then return ceiling(u_1/(k*p))*k

else if (p>floor(u_1/k) MOD p) then return floor(u_1/(k*p))*k

else return floor(u_1/(k*p))*k+(u_1 MOD k)+1

endif

endif

LocalLoopIndex(i)=floor((i–l)/k*p)*k+((i–l) MOD k)+1 //全局下标向局部下标的转换

GlobalLoopIndex($i$$,p)=floor(($i$$–1)/k)*p*k+(($i$$–l) MOD k)+l

/局部下标向全局下标转换

计算实例如下:

Owner(18)=floor((18–1)/4) mod 4=0 //a_{18} 分布在第 1 个结点上.

Owner(11)=floor((11–1)/4) mod 4=2 //a_{11} 分布在第 3 个结点上.

LowerLoopBound(1,0)=LowerLoopBound(1,1)=LowerLoopBound(1,2)=

LowerLoopBound(1,0)=1 //即所有结点的下界均从 1 开始.

UpperLoopBound(20,0)=floor(19/(4*4))*4+(19 mod 4)+1=8

//即第 1 个结点上的循环上界为 8

UpperLoopBound(20,0)=UpperLoopBound(20,1)=

UpperLoopBound(20,2)=UpperLoopBound(20,3)=floor(19/(4*4))*4=4

//即第 2,3,4 个结点上的循环上界为 4

值得指出的是,Block_Cyclic 分布的作用不仅仅在于它是 Block,Cyclic 分布方式的推广,对于矩阵计算等一类工程应用中最常见的算法,它具有特殊的作用.因为大矩阵的计算往往是分块进行的,通过 Block_Cyclic 实现二维矩阵块在不同结点上的分布,有利于提高矩阵计算的高效率.当然分布块的大小、维数等的确定,应根据实际的算法确定.在 ScaLAPACK 就采取了行列块分布的方式.设图 1 中数组为 x(1:20),p=4,k=4 则分布情况如图 3 所示.

3 对 DAD 标准化的讨论

大规模并行应用系统的开发,要求有工程化的方法,要求使用各种语言的串行或并行的算法库,由于不同的语言编译器和其网络数据并行计算系统的实现采用不同的运行支持系统,所以 DAD 的格式也不同.如 p-HPF 并行编译器采用的是 Adlib 运行支持库,数据并行算法库 ScaLAPACK 采用的是 BLACS 运行支持库.这为并行应用系统的可移植性带来了极大的困难.也为一些并行算法库的应用带来很大麻烦.如当 ScaLAPACK 算法库被 HPF 程序调用时,HPF 必须将自己参数的 DAD 格式转变成 ScaLAPACK 的格式.再如,由于运行支持库的 DAD 格式不同,用一个 HPF 编译器编译生成的运行程序不能在装有另一个运行支持系统的环境中运行,这也对并行运算系统的可移植性和重用性造成影响.为了克服这些缺点,需要有一个 DAD 标准,通过标准屏蔽掉不同运行支持系统的差异,对提高并行模块的通用性、程序的可移植性有重要意义.当然,提出这样一个标准,要充分考虑

各个运行支持系统的共性和特性,做到尽量方便各系统的使用,这正是我们下一步工作的方向.

References:

[1] Schreiber, R.S. The Data Parallel Programming Model. Berlin: Spring-Verlag, 1996. 27 ~ 41.

[2] Hall, M.W. Interprocedure compilation of Fortran D for MIMD distributed-memory machines. Technical Report, CRPC-TR 91195, Houston: Rice University, 1991.

[3] Brandes, T. HPFIT: a set of integrated tools for the parallelization of applications using high performance Fortran. Parallel Computing, 1997,23(10):89~105.

[4] Du, Zhi-hui, Ding, Wen-kui, Zheng, Geng-bin, *et al*. Research and implementation of an HPF compilation system. Journal of Software, 1999,10(1):60~68 (in Chinese).

[5] Wang, Jian-ping, Cheng, Xu, Ding, Wen-kui, *et al*. The implementation strategy of communication in HPF compiler and related algorithms. Chinese Journal of Computers, 1999,22(5):486~496 (in Chinese).

[6] Blackford, L.S. ScaLAPACK User's Guide. Philadelphia: Society for Industry and Application Mathmatics, 1997.

附中文参考文献:

[4] 都志辉,丁文魁,郑耿斌,等.一个 HPF 编译器的设计与实现.软件学报,1999,10(1):60~68.

[5] 汪剑平,程旭,丁文魁,等.HPF 编译器中的通信实现策略及其相关算法.计算机学报,1999,22(5):486~496.

Research on the Distributed Array Description in Parallel Computing Based on Networks*

HU Chang-jun[1], DING Wen-kui[2], HUANG Qi-jun[2], XIANG Hua[2], XU Zhuo-qun[2]

[1](*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*);

[2](*Department of Computer Science and Technology, Beijing University, Beijing* 100871, *China*)

E-mail: huchangjun@tsinghua.edu.cn

http://www.cs.tsinghua.edu.cn

Abstract: How to describe the distributed array is a key point in parallel computing based on networks. Based on the general requirement of the parallel implementation, this paper first discusses the structure and necessary components of DAD (distributed array descriptor). Then a certain DAD structure used by p-HPF parallel compiling system is introduced in detail. Further, for the sparse and the dense storage model, the methods are given out to convert global arrays to local ones in three distribution Blocks, the Cyclic and the Block_Cyclic(*k*) respectively. These methods have been implemented in p-HPF compiler and proved to be effective. Finally, the importance of the distributed array description standard for portability and reuse of the parallel computing systems is discussed.

Key words: distributed array description; network parallel computing; HPF language; parallel compiling

* Received March 16, 2000; accepted September 25, 2000

Supported by the National Natural Science Foundation of China under Grant No.60173004; the National High Technology Development 863 Program of China under Grant No.863-306-ZT01-02-03