分布式系统并行编程的简单方法*

迟学斌

(中国科学院软件研究所 北京 100080)

摘要 本文以并行矩阵乘法为例子,给出在PVM, EXPRESS以及NX环境上的并行程序实现.通过对这个简单例子的分析,归纳出在一般情况下并行程序设计应采取的策略.这里介绍的并行程序设计方法,特别对大型应用问题的并行移植是非常有用的,其目的是为想写并行程序的科研人员提供参考.

关键词 PVM, EXPRESS, NX, 矩阵乘法, 消息传递, 可移植性, 并行程序设计, 分布式系统, MPP.

中图法分类号 TP311.11

分布式并行计算机系统大体上可分为两大类:工作站机群(NOW, COW)和MPP系统.在工作站机群上的并行编程环境有PVM, MPICH, JAVA等,由于该类分布式系统具有充分利用网络上资源的特点,已经受到越来越多的研究开发人员的青睐.而MPP系统与超级计算机系统相比,以其高性能价格比的特点,自80年代以来得到了惊人的发展,目前浮点运算速度最快的已达到万亿次.我国在MPP系统的研制方面也有飞速发展,已经研制出千亿次并行计算机系统,就当今世界计算机的运算速度而言,排名已属世界前列.由此可以看出,分布式并行计算机系统(MPP)的发展是异常迅猛的.

如何使用并行计算机并最大限度地提高它的效率是人们普遍关心的问题,同时对于在并行计算机上做实际应用的广大用户来说,他们也希望所编写的应用程序可以非常容易地从一个并行系统移植到另一个系统.目前国际上出现了PVM和MPI等并行编程环境,得到了并行计算机制造者的认可,目前的并行计算机上都要求有PVM和MPI等并行编程环境,从而给并行程序的移植带来了巨大的方便.那么,是否在分布式并行计算机系统上使用一种并行编程环境就可以了呢?对于不要求高效率的应用者来说,选择一种通用的并行编程环境是可以的,但对于追求高效率的使用者来说,由于通用的并行编程环境在消息传递方面不如并行计算机系统上提供的特殊环境,因此在不同的并行计算机系统上就要考虑使用不同的并行编程环境.在传统的串行计算机上,已经有许多程序设计技术方面的有关研究.[1]本文的出发点是以大规模科学与工程计算问题为背景,因此增加可移植性对编写应用程序尤其重要.

1 行列划分矩阵乘算法

假设有 p 个处理机,用 P_i 表示第 i 个处理机,A, B 和 C 都是 $m\times m$ 矩阵,并且 $m=n\times p$,则数据划分及在处理机中的存放形式为:$A=[A_0^T \quad A_1^T \quad \dots \quad A_{p-1}^T]^T$, $B=[B_0 \quad B_1 \quad \dots \quad B_{p-1}]$, $C=[C_0^T \quad C_1^T \quad \dots \quad C_{p-1}^T]^T$.其中 A_i 和 C_i 是 $n\times m$ 矩阵,B_i 是 $m\times n$ 矩阵,并都存放在 P_i 中.计算 C_i, $i=0,\dots,p-1$ 的方法为:$C_i=[A_iB_0 \quad A_iB_1 \quad \dots \quad A_iB_{p-1}]$.假设在每个结点上的块矩阵仍记为 A, B, C,用 C_j 表示该处理机上 C 的第 j 个列子块,在 P_{me} 上的计算方法如下[2]:

结点 P_{me} 上的并行算法:

```
for i=0 to p-2 do;
    k=(me-i) mod p;Ck=A×B;
    if me mod p=0,then,send B to P(me+1) mod p;receive B from P(p+me-1) mod p;
    else,receive B from P(p+me-1) mod p;send B to P(me+1) mod p;
    end{if}
end{for}
k=(me+i) mod p;Ck=A×B;
```

---

* 本文研究得到国家自然科学基金和国家攀登计划基金资助.作者迟学斌,1963年生,博士,副研究员,主要研究领域为并行计算与软件.

本文通讯联系人:迟学斌,北京 100080,中国科学院软件研究所

本文1997-01-29收到原稿,1997-04-04收到修改稿

在下面的程序实现中,要用到初始化矩阵 A 和 B 的程序,还要用到 BLAS(basic linear algebra subprograms) 中的矩阵乘子程序(Dgemm)和向量复制子程序(Dcopy).

2 PVM 环境上的实现[3]

```
      program pmmm
c Variable declaration omited for shortness.
      call pvmfmytid(mytid)
      call pvmfparent(tid(0))
      if (tid(0) .lt. 0) then
        me=0
        tid(0)=mytid
        call pvmfspawn('pmmm',0,'*',np-1,
     &                    tid(1),numt)
        call pvmfinitsend(0,inf)
        call pvmfpack(INTEGER4,tid,np,1,inf)
        call pvmfmcast(np-1,tid(1),0,inf)
      else
        call pvmfrecv(tid(0),0,inf)
        call pvmfunpack(integer4,tid,np,1,inf)
        do 10  i=1,np-1
          if (mytid .eq. tid(i)) me=i
10      continue
      endif
      call initab(...)
      call pdmmm(me,np,tid,m,n,a,b,c)
      call pvmfexit(inf)
      print *,'Program Finished'
      stop
      end

      subroutine pdmmm(me,np,tid,m,n,a,b,c)
c Variable declaration omited for shortness.
      mep1=mod(me+1,np)
      mem1=mod(np+me-1,np)
      do 30 i=0,np-2
        k=mod(me+i,np)*n+1
        call dgemm('n','n',n,n,m,1.0d0,a,
     &             n,b,m,0.0d0,c(1,k),n)
        call pvmfinitsend(0,inf)
        call pvmfpack(REAL8,b,m*n,1,inf)
        if(mod(me,2) .eq. 0) then
          call pvmfsend(tid(mep1),me9,inf)
          call pvmfrecv(tid(mem1),mem1+9,inf)
        else
          call pvmfrecv(tid(mem1),mem1+9,inf)
          call pvmfsend(tid(mep1),me+9,inf)
        endif
        call pvmfunpack(REAL8,b,m*n,1,inf)
30    continue
      k=mod(me+i,np)*n+1
      call dgemm('n','n',n,n,m,1.0d0,a,
     &           n,b,m,0.0d0,c(1,k),n)
      return
      end
```

3 EXPRESS 环境上的实现[4]

```
      program pmmm
c Variable declaration omited for shortness.
      call kxinit
      call kxpara(env)
      me=env(1)
      np=env(2)
      call initab(...)
      call pdmmm(me,np,m,n,a,b,c,w)
      print *,'Program Finished'
      stop
      end

      subroutine pdmmm(me,np,m,n,a,b,c,w)
c Variable declaration omited for shortness.
      call kxgdin(1,np)
      mem1=kxgdno(myid,0,-1)
      mep1=kxgdno(myid,0,1)
      do 30 i=0,np-2
        k=mod(me+i,np)*n+1
        call dgemm('n','n',n,n,m,1.0d0,a,
     &             n,b,m,0.0d0,c(1,k),n)
        if(mod(me,2) .eq. 0) then
          istat=kxwrit(b,m*n*8,mep1,me+9)
          istat=kxread(b,m*n*8,mem1,mem1+9)
        else
          istat=kxread(w,m*n*8,mem1,mem1+9)
          istat=kxwrit(b,m*n*8,mep1,me+9)
          call dcopy(m*n,w,1,b,1)
        endif
30    continue
        k=mod(me+i,np)*n+1
        call dgemm('n','n',n,n,m,1.0d0,a,
     &             n,b,m,0.0d0,c(1,k),n)
        return
        end
```

4 NX 环境上的实现[5]

```
      program pmmm
c Variable declaration omited for shortness.
      np=numnodes()
      me=mynode()
      call initab(...)
      call pdmmm(me,np,m,n,a,b,c,w)
      print *,'Program Finished'
      stop
      end

      subroutine pdmmm(me,np,m,n,a,b,c,w)
c Variable declaration omited for shortness.
```

```
      mep1=mod(me+1,np)
      mem1=mod(np+me-1,np)
      do 30 i=0,np-2
        k=mod(me+i,np)*n+1
        call dgemm('n','n',n,n,m,1.0d0,a,
     &             n,b,m,0.0d0,c(1,k),n)
        if(mod(me,2).eq.0) then
          call csend(me-9,b,m*n*8,mep1,0)
          call crecv(mem1+9,b,m*n*8)
        else
          call crecv(mem1+9,w,m*n*8)
          call csend(me+9,b,m*n*8,mep1,0)
          call dcopy(m*n,w,1,b,1)
        endif
30    continue
        k=mod(me+i,np)*n+1
        call dgemm('n','n',n,n,m,1.0d0,a,
     &             n,b,m,0.0d0,c(1,k),n)
      return
      end
```

5　统一的实现方案

在上面给出的各种程序实现中，其结构大致相同，差别最大之处在产生进程并得到进程个数和局部进程编号上，再就是，通讯函数有很大差别. 为讨论方便起见，假设创建并行环境的函数为 setpar，发送消息的函数为 mysend，接收消息的函数为 myrecv. 由于 EXPRESS 和 NX 环境在实现上述 3 个函数时是非常简单的，故此只选择 PVM 进行分析. 创建并行环境的函数 setpar 有两个参数，me，np，其中为 me 逻辑进程号，np 表示全部进程个数. 在 PVM 环境上的简单实现如下：

```
      subroutine setpar(np,me)
c Variable declaration omited for shortness.
      call pvmfmytid(mytid)
      call pvmfparent(tid(0))
      if(tid(0).lt.0) then
        me=0
        tid(0)=mytid
        open(2,file='setpar.dat')
        read(2,*) name
        close(2)
        call pvmfspawn(name,0,'*',np-1,
     &                 tid(1),numt)
        call pvmfinitsend(0,inf)
        call pvmfpack(INTEGER4,tid,np,1,inf)
        call pvmfmcast(np-1,tid(1),0,inf)
      else
        call pvmfrecv(tid(0),0,inf)
        call pvmfunpack(INTEGER4,tid,np,1,inf)
        do 10 i=1,np-1
          if(mytid.eq.tid(i)) me=i
10      continue
      endif
      return
      end
```

需要建立文件 setpar.dat，其内容只是子进程的可执行文件名，再就是 tid 数组要放到 common 块中用 save 定义，以备在消息传递的其它函数中使用. 消息传递函数 mysend 和 myrecv 可按下述方式实现：

```
      subroutine mysend(buf,bsize,desnode,tag)
c Variable declaration omited for shortness.
      call pvmfinitsend(0,info)
      call pvmfpack(byte1,buf,bsize,1,inf)
      call pvmfsend(tid(desnode),tag,inf)
      return
      end
```

```
      subroutine myrecv(buf,bsize,srcnode,tag)
c Variable declaration omited for shortness.
      call pvmfrecv(tids(srcnode),tag,inf)
      call pvmfunpack(byte1,buf,bsize,1,inf)
      return
      end
```

在上述两个子程序中，参数 buf 表示要发送或接收消息的数组，bsize 是 buf 按字节数的长度，tag 表示发送或接收消息的标签，desnode 表示消息要发送到的逻辑进程号，srcnode 表示消息是从何逻辑进程号发来的. 这种做法在消息传递过程中有时会损失一些效率，但比起来不同环境带来的影响要小得多. 在移植过程中，只需要修改几个简单的函数即可，从而提高了移植效率和减少在移植过程中产生的错误，可以说是一种比较好的编程方法. 这里所考虑的是在编写并行程序中使用最频繁的两个通讯函数和一个创建进程的函数，对于一些其它的功能函数，原则上是可以用类似的方法进行处理的，但是每种并行环境都有其特点，不可能用另一种并行环境完全代替，这也是它们各自存在的基础. 比如 PVM 环境中没有提供异步发送(non-Blocking)消息的功能，而在 EXPRESS 和 NX 环境中都提供了此功能；再如 PVM 环境中提供了数据打包(Packing)和任务组的功能，而 EXPRESS 和 NX 环境中都没有提供. 对于使用此类功能函数的并行程序在移植过程中将有很多麻烦，甚至要对实现方法进行彻底的改造.

参考文献

1　Feng Yu-lin, Zhong Cui-hao, Chen You-jun. Programming methdology. Beijing: Beijing Science and Techrology Press, 1989

2　Sun Jia-chang, Zhang Lin-bo, Chi Xue-bin *et al*. Network parallel computing and distributed programming environment.

Beijing: Science Press, 1996
3 Geist A, Beguelin A, Dongarra J *et al*. PVM: parallel virtual machine. The MIT Press, 1994
4 Hitachi computers user's guide——parallelware (FORTRAN). Hitachi, Ltd., April 1995
5 Paragon TM user's guide. Intel Corporation, June 1994

A Simple Method of Parallel Programming on Distributed Memory System

CHI Xue-bin

(*Institute of Software The Chinese Academy of Sciences Beijing* 100080)

Abstract In this paper, the author takes parallel matrix multiplication as an example to see how their parallel implementations are under PVM, EXPRESS and NX environments. By analyzing this simple problem, the strategies of parallel programming are summarized for general purposes. The given parallel programming method is specially useful for parallel porting of a large scale application. The purpose of this paper is to provide a reference for scientists who want to write parallel programs.

Key words PVM, EXPRESS, NX, matrix multiplication, message passing, portability, parallel programming, distributed system, MPP.

Class number TP311.11

第 10 届中国计算机网络与数据通信学术会议

征 文 通 知

根据中国计算机学会学术活动计划安排,计算机网络与数据通信专委会拟定于 1998 年 10 月下旬或 11 月上旬在东南大学召开第 10 届中国计算机网络与数据通信学术会议,并邀请著名网络专家作专题报告,邀请国内外著名网络厂商介绍最新网络技术和产品。

一、征文内容

高性能计算机网络	体系结构及协议	协议工程
形式描述技术(FDT)	网络智能化	网络安全
网络管理	CSCW	多媒体通信
Internet/Intranet	Web 技术	EDI
远程教育和诊断系统	工厂和企业网络	网络应用系统
ATM	ISDN	数据传输与交换
互联与路由技术	通信网组网技术	LAN 技术
WAN 技术		

二、征文通知

1. 论文必须未公开发表过,字数一般不得超过 6000 字;

2. 论文应包括题目、作者姓名、作者单位、摘要、关键词、正文和参考文献,另附作者姓名、单位、地址、邮编、电话、传真及 E-mail 地址;

3. 论文一律为 A4 激光打印稿,一式 3 份;

4. 征文请寄:210096 南京市 东南大学计算机系 顾冠群 罗军舟
电话:(025)3792359 传真:(025)3793072

三、重要日期

1. 截稿日期:1998 年 5 月 31 日
2. 录用通知:1998 年 6 月 30 日
3. 正式文稿:1998 年 8 月 10 日

四、论文刊出

大会录用的论文将收入第 10 届中国计算机网络与数据通信学术会议论文集,优秀论文还将推荐到《软件学报》.