Windows2000 下基于 PVM 的并行计算实践研究①

The Practical Study of Parallel Computing Based on PVM System in Windows2000

尚月强　（贵阳 贵州师范大学数学与计算机科学学院　550001）

摘要：为了把网络计算在我国中小研究单位和大专院校推广普及，本文详细介绍了 Windows2000 操作系统下并行环境 PVM 的构建、PVM 并行程序的设计、编写、调试等，并给出了编程实例。实践表明，基于 PVM 的并行计算以其构架简单、使用方便、计算能力大、性能价格比高等优势，特别适合我国中小研究单位和大专院校，必将成为我国网络计算的重要方向，普遍用于各科学领域。

关键词：PVM　Windows2000　并行计算

PVM（Parellel Virtual Machine）是最初由美国 Oak Ridge 国家实验室设计开发的消息传递类并行软件开发环境，它通过 TCP/IP 网络通信协议将一个网络中的各种计算机虚拟成一台并行机来使用，使得大型计算在花费较少的情况下得以实现。具有规模小、编程容易、源代码公开、高度透明和优良的容错性等诸多优点[1]，备受中小研究单位、院校的青睐，得到了广泛的应用。PVM 最先是构建在 Unix 操作系统上的，后来又开发出了基于 Windows 的版本。由于我国绝大多数用户使用的是 Windows2000 操作系统，因此 Windows2000 下基于 PVM 的并行计算研究具有重要的现实意义。

1　Windows2000 下 PVM 并行计算平台构架

1.1　PVM 的安装与配置

最新的 PVM3.4.3 for Win32 版本自带了一个安装程序，遵循其安装向导即可顺利完成 PVM 的安装。不过 PVM 本身没有编译环境，因此安装 PVM 之前必须安装 PVM 支持的语言 C\C++ 或 Fortran 来支持它的编译（笔者选用的是 VC6.0），同时在安装过程中正确设置好环境变量 PVM_ROOT 和 PVM_TMP。需注意的是 PVM 安装程序不会自动生成临时文件夹 Temp，因此用户需预设一个临时文件夹 Temp 来存放 PVM 的两个日志文件 pvmd.uid 和 pvml.uid[2]，否则 PVM 将无法启动。另外还需安装的软件是 rshd（或 rexecd），用于远程调用。

1.2　PVM 的启动及虚拟机的配置

PVM 安装成功后，点击“开始”菜单中 PVM3.4 的 PVM Console 图标，将打开 PVM 控制台，同时启动 pvmd 后台伺候程序，以上工作完成后会出现以下提示符：

```
pvm >_
```

用 add 命令即可增加节点机进 PVM 系统（最先启动 PVM 的计算机称为主机（Master Host），主机上用 add 命令增加的计算机称为节点机或从属机（Slave Host））。需注意的是增加一台节点机进 PVM 系统时首先应启动该节点机上的 rshd，增加命令行除了指出节点机名，还需指出其守护进程的位置，如：

```
add " node2 dx = D: \progra ~ 1 \pvm3.4 \lib \win32 \pvmd3.exe"
```

还需增加其他节点机，重复上述命令即可。同时可用 conf 命令查看 PVM 系统中节点机的配置情况。

当无法增加一台节点机进 PVM 系统时可从以下几方面检查：① 网络是否畅通；② rshd 是否安装并启动；③ PVM 临时目录/Temp 下的文件 pvmd.uid 是否删除，它是由于上次非正常退出 PVM 而遗留下来的。

① 基金项目：贵州师范大学学生科研基金资助项目

2　Windows2000 下基于 PVM 的并行计算编程

2.1　任务的分解与编程模式的选择

并行程序的设计首先是任务的分解,即根据算法的自然特性和数据结构特性等,将整个计算任务分解成多个相对独立的、可并行执行的子任务[1]。最常用的任务分解方法有数据(域)分解(将数据分解,然后对分解的数据并行操作)和功能分解(将程序分为独立的功能模块,然后并行地执行这些功能模块)[3]。采用何种分解方法,可根据任务所涉及的数学公式、数据定义域、算法及通信模式等灵活选择。一般静态数据结构,单一实体的动态数据结构可采用数据分解方法[1]。

任务分解完毕后还要选择编程模式,PVM 提供两种基本编程模式:Master/Slave 和 SPMD。前者需编写两份代码(即 Master 程序和 Slave 程序),由 Master 程序激活一组 Slave 进程,并调度这些进程;后者所有进程地位平等,没有一个 Master 程序在指挥,只需一份程序[4]。采用何种模式,可根据任务的分解策略及硬件环境、I/O 操作等相关因素进行选择。

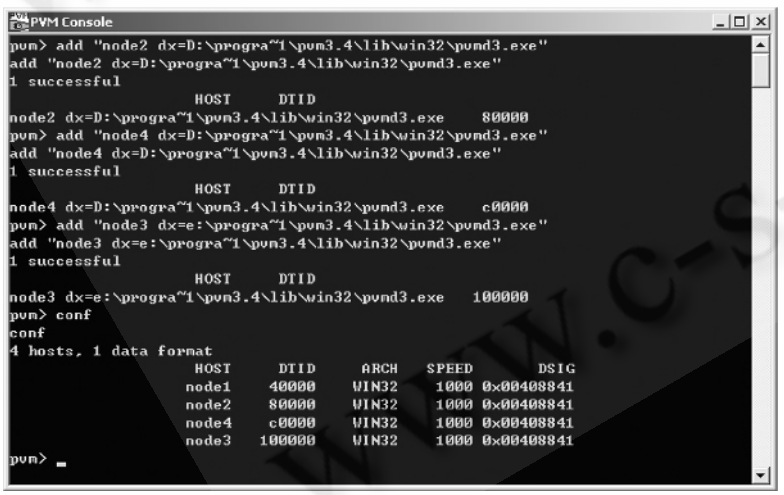

图 1　PVM &C + +编程环境的设置

2.2　并行程序的编写、编译与执行

这里只讨论 Master/Slave 模式。PVM 并行程序的编写要考虑性能、网络和负载平衡三个主要因素。可将原有的串行程序并行化,也可重新编写并行程序。前者因其容易实现,常常是首选方法,但效率不高,因此熟悉并行编程环境后,采用后者能充分开发任务的并行性。PVM 并行程序用 C\C + +或 Fortran 语言编写,通过 PVM 提供的消息传递机制实行并行计算。在编写 PVM 并行程序时,首先应将头文件"pvm3.h"包含进去,又由于 PVM 允许创建与处理器数目毫无相关的任意数量的进程,因此用户无需指定,PVM 能自动将进程分配给合适的处理器进行处理[4]。

PVM 并行程序编写好后,就需进行编译。笔者选用的是 VC6.0 作为 PVM 的编译环境,编译前应将"$PVM_ROOT\include"、"$PVM_ROOT\src"和"$PVM_ROOT\lib\win32"分别加入到 VC6.0 编译器的"include files"和"library files"目录下,可通过"工具"→"选择"→"目录"设置,如图 1 所示。

编译通过后,再将"wsock32.lib"和"libpvm3.lib"(若用了组功能,还需将"libgpvm3.lib")加入到 VC6.0 编译器的"L 对象/库模块"中(通过"工程"→"设置"→"link"设置),点击"运行"即可生成可执行文件,并将其放在 $PVM_ROOT\bin\WIN32 目录下。若 PVM 是同构的,Slave 程序只需编译一次,将生成的 Slave 可执行文件拷贝到各台节点机的上述目录下即可,若不同构,则需将 Slave 程序拷贝到各台节点机重新编译。然后在主机上运行 Master 程序即可得到 PVM 并行程序运行结果。

2.3　PVM 并行程序的调试

PVM 并行程序调试的一般方法是:先确定串行程序的正确性;并行程序在单机上运行通过后再逐步增加处理机数;逐步增加程序中的并行性[1]。同时 PVM 还提供了错误信息输出函数 PVM_perror[5],在程序中加入该函数,可以动态地跟踪了解并行程序的执行情况;还可以使用函数 PVM_setopt[5] 设置某些参数,以了解并行程序运行的详细信息。

3　编程实例

下面是数据分解的 Master/Slave 模式下 C + +语言编程实例:计算两向量 $a,b \in R^{100000}$ 的内积 $\langle a,b\rangle = a_1b_1 + a_2b_2 + \cdots + a_{100000}b_{100000}$:

Master 程序负责发送数据到各处理机,并从各处理机接收部分结果,累加求总和并输出。Slave 程序负责从 Master 程序接收数据,完成所需计算并将计算结

果送回 Master 程序。

Master 程序:nj. cpp

```
#include <iostream. h >
#include " pvm3. h"
#define SLAVE " njslave"
#define SIZE 100000 / * 向量 a,b 的维数 * /
void main( )
{
  int mytid; / * 父进程任务号 * /
  int tids[32]; / * 各从进程任务号 * /
  int n, nproc, numt, i, who, msgtype, nhost,
  narch;
  float a[SIZE],b[SIZE], result[32],total =0.0;
  char buf[100];
  struct pvmhostinfo * hostp;
  mytid = pvm_mytid( ); / * 注册进 PVM 系统,
  获得本进程号 * /
  pvm_config( &nhost, &narch, &hostp ); / * 获
  得当前 PVM 系统中各节点机信息 * /
  cout < < " the number of computers in pvm sys-
  tem is " < < nhost < < endl;
  nproc = nhost * 2; / * 进程数 * /
  if( nproc > 32 ) nproc = 32 ;
  cout < < " Spawning " < < nproc < < " worker
  tasks . . . " ;
  numt = pvm_spawn( SLAVE, ( char * * ) 0, 0,
  " " , nproc, tids); / * 派生 nproc 个任务 * /
  if( numt < nproc ){
    cout < < endl < < " Trouble spawning slaves.
  Aborting. Error codes are: " < < endl;
    for( i = numt ; i < nproc ; i + + )
      cout < < " TID " < < i < < " " < < tids[i] <
  < endl;
    for( i =0 ; i < numt ; i + + ) / * 派生任务
  失败则终止各进程 * /
      pvm_kill( tids[i] );
    pvm_exit( ); / * 退出 PVM 系统 * /
    exit(1); / * 退出程序 * /
  }
  cout < < " SUCCESSFUL" < < endl;
  n = SIZE;
  for( i =0 ; i < n ; i + + ){ / * 初始化向量 a,b
  * /
    a[i] = (i +1)/1000.0f;
    b[i] = (2 * i +1)/1000.0f;
  }
  pvm_initsend( PvmDataRaw ); / * 初始化发送
  消息缓冲区 * /
  pvm_pkint( &nproc, 1, 1); / * 数据打包 * /
  pvm_pkint( tids, nproc, 1);
  pvm_pkint( &n, 1, 1);
  pvm_pkfloat( a, n, 1);
  pvm_pkfloat( b, n, 1);
  pvm_mcast( tids, nproc, 0); / * 向各进程广
  播数据 * /
  for( i =0 ; i < nproc ; i + + ){ / * 接收各进程
  发来的部分和并累加求总和 * /
    pvm_recv( -1, 5 );
    pvm_upkint( &who, 1, 1 );
    pvm_upkfloat( &result[who], 1, 1 );
    pvm_upkstr( buf );
    cout < < " I got " < < result[who] < < " from
  " < < who < < " ( " < < buf < < " )" < < endl;
    total + = result[who];
  }
  cout < < " < a,b > = " < < total < < endl; / *
  输出结果 * /
  pvm_exit( ); / * 退出 PVM 系统 * /
}
```

Slave 程序清单略。

4 结论

从以上编程实例及介绍可知,基于 PVM 的并行计算程序以任务为单位,一个任务就是一个系统进程,每个任务都有一个 task_id 来标识。其编程容易,只需在传统的 C\C + + 或 Fortran 串行程序中插入 PVM 通信原语,即可实现并行计算。加之 Windows2000 操作系统占领了我国近 90% 的市场份额,因此 Windows2000 下基于 PVM 的并行计算以其构架简单、使用方便、计

(下转第 73 页)

(上接第 69 页)

算能力大、性能价格比高等优势,特别适合我国中小研究单位和大专院校,必将成为我国网络计算的重要方向,普遍用于各科学领域。

参考文献

1 吴维刚、董小社、何戈,基于机群系统的 PVM 并行编程技术研究[J],计算机应用研究,2003 年第 2 期:21 – –23。

2 张进波、徐静雯、李元香,Windows 下 PVM 环境的配置及其库函数的时间性分析[J],计算机工程与应用,2001.16:58—59。

3 Barry Wilkinson,Michael Allen 著,陆鑫达译,并行程序设计[M],机械工业出版社,2002.1:88。

4 李代平、张信一、罗寿文等,分布式并行计算技术[M],冶金工业出版社,2004.1:31。

5 Geist A et al. PVM:Parallel Virtual Machine—A Users´Guide and Tutorial for Networked Parallel Computing[M]. Boston:MIT press. 1994.6:38——46。