基于 MPI 的背包问题并行程序设计与实现

张居晓
（南京特殊教育职业技术学院文理学院，南京 210038）

摘 要：MPI（Message Passing Interface）是消息传递并行程序设计的标准之一，概述了 MPI 的概念和组成，着重介绍了支持并行程序设计的消息传递接口（MPI）以及在 MPI 环境下的并行程序设计方法，并给出一个 MPI 并行程序设计实例，说明了 MPI 的程序设计流程和普通串行程序设计之间的关联。

关键词：MPI；并行程序设计；背包问题
中图分类号：TP311.11 **文献标识码**：A **文章编号**：1007-9599（2011）21-0159-02

MPI-based Parallel Programming and Implementation of Knapsack Problem

Zhang Juxiao
(Arts and Sciences of Nanjing Technical College of Special Education,Nanjing 210038,China)

Abstract:MPI(Message Passing Interface)message passing parallel programming is one of the criteria,outlined the concept and composition of MPI,focuses on support for parallel programming Message Passing Interface(MPI)and MPI parallel programming environment method,and gives a MPI parallel programming examples to illustrate the design of MPI processes and procedures common link between the serial programming.

Keywords:MPI;Parallel programming;Knapsack problem

一、MPI 原理和特征

MPI（Message Passing Interface）是消息传递并行程序设计的标准之一，MPI 正成为并行程序设计事实上的工业标准。对于 MPI 的定义，必须明确以下几点：①MPI 是一个库，而不是一门语言；②MPI 是一种标准或规范的代表，而不是特指某一个对它的具体实现；③MPI 是一种消息传递编程模型，并成为这种编程模型的代表和事实上的标准。

MPI 作为一个并行程序库的开发平台，为用户编写和运行程序提供了便利的条件。由于 MP I 是基于消息传递机制构建的系统，因此它在体系结构为分布存储的并行机中有很宽阔的应用空间，它可以应用在各种同构和异构的网络平台中。它的编程语言可以为 Fortran77/90、C/C++。在 Fortran77/90、C/C++语言中都可以对 MPI 的函数进行调用，它作为一种消息传递模式的并行编程环境，MPI 并行程序要求将任务进行划分，同时启动多个进程并发的执行，而各个进程之间通过 MPI 的库函数来实现其中的消息传递。MPI 与其它并行编程环境相比，显著的特点有：（1）可移植性强，能同时支持同构和异构的并行计算；（2）可伸缩性强，允许并行结构中的节点任意增加或减少；（3）能很好的支持点对点通信和集体通信方式；（4）对 C 语言和 Fortran 语言的支持，使其能很好的满足各种大规模科学和工程计算的需要。这样，以 MPI 作为公共消息传递接口的并行应用程序就可以不作任何改动的移植到不同种类和型号的并行机上，也能够正常运行，或者移到网络环境中也一样。

二、MPI 的基本函数

MP I 为消息传递和相关操作提供了功能强大的库函数，从理论上来说，MPI 的所有通信功能都可以用它的 6 个基本调用来完成，即使用这 6 个函数可以实现所有的消息传递并行程序。这六个函数分别为（C 语言的调用格式来描述）：

（一）MPI 初始化

```
int MPI_Init (int *argc,  char ***argv)
```

MPI_INIT是MPI程序的第一个调用它完成MPI程序所有的初始化工作所有MPI程序的第一条可执行语句都是这条语句。

（二）MPI 结束

int MPI_Finalize (void) 是MP I程序的最后一个调用，它结束MP I程序的运行，它是MP I程序的最后一条可执行语句，否则程序的运行结果是不可预知的。

（三）当前进程标识

```
int MPI_Comm_rank (MPI_Comm comm,  int *rank)
```

这一调用返回调用进程在给定的通信域中的进程标识号，有了这一标识号，不同的进程就可以将自身和其它的进程区分开，实现进程的并行和协作。

（四）通信域包含的进程数

```
int MPI_Comm_size (MPI_Comm comm,  int *size)
```

这一调用返回给定的通信域中所包含的进程的个数，不同的进程通过这一调用得知在给定的通信域中一共有多少个进程在并行执行。

（五）消息发送

```
int MPI_Send (void* buf,  int count,  MPI_Datatype datatype,  int dest,  int tag,  MPI_Commcomm)
```

该调用将发送缓冲区中的 count 个 datatype 数据类型的数据发送到目的进程，目的进程在通信域中的标识号是 dest，本次发送的消息标志是 tag，使用这一标志，就可以把本次发送的消息和本进程向同一目的进程发送的其它消息区别开。

（六）消息接收

```
int MPI_Recv (void* buf,  int count,  MPI_Datatype datatype,  int source,  int tag, MPI_Comm comm,  MPI_Status *status)
```

该调用从指定的进程 source 接收消息，并且该消息的数据类型是消息标识要和本接收进程指定的 data2tyPe 和 tag 相一致，接收到的消息所包含的数据元素的个数最多不能超过 count。

三、背包问题分析

问题描述：有不同价值、不同重量的物品 n 件，求从这 n 件物品中选取一部分物品的方案，使选中的物品价值总重量不超过指定的限制重量，但是物品的价值之和最大。

这里我采用了比较简单的动态规划法来解决，动态规划简而言之就是将大问题分解为小问题，并综合小问题的解导出大问题的方法。为了节约重复相同自问题的时间，引入一个数组，不管它们是否有用，把所有子问题的解存于该数组中，这就是动态规划的基本方法。这个方法刚好在并行计算中能够很好的使用。

输入：各种物品的价值 p1，p2……pm，各种物品的重量 w1，w2……wm，背包容量 c。

输出：使放入背包的总价值最大的物品编号。

算法：

```
for i=0  to  c
    f[0][i]=0
endfor
for I=1 to m
   f[i][0]=0
   for j=1 to c          ***
if wi<=j
if  pi+f[i-1][j-wi]>f[i-1][j]
```