计算机科学 2000Vol. 27№. 8

并行计算模型的层次分析及性能评价*)

The hierarchical Analysis and Performance Evaluating for Parallel Application

刘方爱　乔香珍
(国家智能计算机研究开发中心　中国科学院计算技术研究所　北京 100080)
刘志勇
(国家自然科学基金委员会)

Abstract In this paper, the performance analysis of parallel programs is discussed from user's level and a parallel computing hierarchy is given. After that, some performance properties are analyzed. Based on the parameters in user's program, a framework for parallel algorithm performance evaluation is developed. From this framework, the communication delay between two processors or among a group of processors can be predicted and the speed-up can also be calculated. To support our results, an experiment is done.

Keywords Communication latency, Performance evaluation, Efficient bandwidth, Speedup, Transfer state

1. 引言

如何分析、评价并行程序的性能是并行计算的一个重要研究问题。RAM 模型为串行算法分析提供了理论基础，据此，我们可以用 big-O 模型来分析其算法的复杂性。但是，在并行环境下，由于处理机之间增加了通信，使得并行程序及算法分析问题更加复杂。如何在并行程序和计算机结构之间建立一种联系？许多人为此作过大量的研究，提出了一些并行计算模型，如 LogP 模型[1]、C^3 模型[2]等。但是，在这些模型中，一切分析都是从体系结构的角度出发，这要求程序设计者对计算机的系统结构有深入的了解。另外，这些模型使用复杂，尤其对应用程序设计者来说，他们对系统的硬件细节不见得熟悉，因此使用它们比较困难；再者这些模型缺乏从应用层上对并行系统进行分析的能力。鉴于以上原因，在该文中，我们从应用层对并行计算模型进行了分析。我们首先分析了并行计算的层次问题，然后针对用户层，讨论了其性能评价指标，提出了有效带宽的三种状态，即空余状态、理想状态和拥挤状态的概念；其次，我们从用户层出发，提出了一个性能分析的简单模型，给出了一组公式，利用这组公式，我们可以预测一个节点到另一个节点之间的通信延迟时间，以及整个并行程序的加速比。最后，我们设计了并行程序并在曙光机进行了试验，验证了我们所提出模型的正确性。

2. 并行计算模型的层次

什么是并行计算模型？一般认为：

· 并行计算模型应在并行算法和并行计算机体系结构之间建立一座桥梁，以便用户借此来分析并行算法的性能。

· 并行计算模型应为应用软件开发人员提供一些性能指标，引导他们开发效率更高的程序。

要说明一个并行程序效率的高低，一个主要的指标是并行系统的加速比，围绕加速比的概念，不同的并行计算模型提出了不同的评价指标。但是，我们知道，并行计算机的体系结构千差万别，一个并行算法在一个并行结构上得到较高的性能加速比，而在另一种并行计算机上，其加速比不见得高，因此对并行计算模型的分析离不开硬件系统性能指标。那么，采用哪些性能指标进行分析呢？首先，我们从用户观点来分析并行应用的层次问题，从逻辑上看，并行应用涉及到如下四层，如图 1。

应用层：指用户的工作域，在这一层，用户使用程

*)本文的工作得到国家自然科学基金 69933020 和国家八六三高技术项目的支持。刘方爱　在职博士研究生，主要研究方向为并行算法、性能评价；乔香珍　研究员，主要研究方向为并行计算、并行计算模型和计算机体系结构；刘志勇　研究员，博士生导师，主要研究方向为并行算法、计算机体系结构。

序设计语言作为工具，来表达实际问题，如 C、Fortran 等。

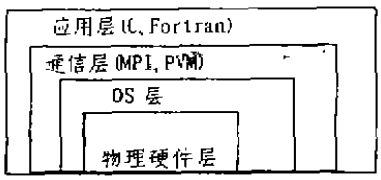

图 1　并行应用的层次结构

通信原语层：提供用户级数据通信的调用接口，用于在并行计算机节点之间传递信息。目前，在分布式内存的并行计算机中，基于消息传递的主要通信接口是 PVM 和 MPI[3]。

os 层：并行应用宿主的操作系统。

物理硬件层：这包含计算机、互联网络等。

一般来说，应用程序的设计者只关心上两层，而对物理硬件层的性能指标不必熟悉。因而，并行计算模型应通过前两层的性能指标来反映。这样，用户便于了解，也便于引导用户编写高效率的程序。通常用户编写并行程序时，要考虑如下问题：程序并行运行的节点个数；并行算法；通信问题（含同步问题）。

对于并行算法中工作量的计算，可以从多种角度来刻画，如并行计算的复杂度或浮点运算次数。

通信问题涉及到通信方式（点点通信、群集通信）、通信模式（阻塞式和非阻塞式通信）和路由方式（store and forward，wormhole）。在这三个方面中，前两个方面是由用户选择的，在用户程序设计时需要用到；而后一种是系统实现时固有的，用户对此无法选择。因此从应用层考虑，路由方式的特性可以不在并行计算模型中反映出来。我们再来看通信模式，在通信模式中，有阻塞式通信和非阻塞式通信两种方式（图 2），其主要区别如下：

很明显，阻塞方式增加了等待时间。但是，在应用程序设计时，当程序使用了非阻塞方式发送时，往往要进行检测，确定发送缓冲区是否可用，因此这也需要一些时间。为了简化分析，我们不区分阻塞和非阻塞发送方式，因此只统一地认为发送数据和接收数据。下面我们进一步细化通信问题，影响通信效率有多种因素，如通信启动时间、信息报文长度、路由算法、通信带宽的拥挤度。但是，在应用层，用户往往关心通信速度。进一步说，在特定并行计算环境下，用户关心：

- 在什么条件下，点点通信达到最佳通信效率？每秒能传输多少字节？
- 当 p 个进程群集通信时，在什么条件下达到最佳通信效率？
- 通信量和通信发送次数对应用系统通信性能的影响如何？

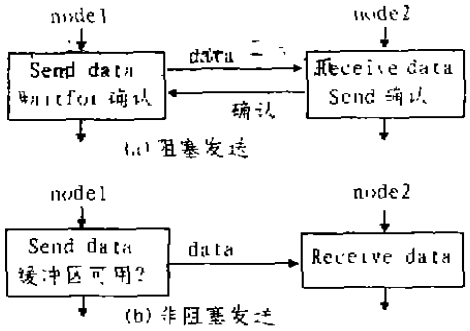

图 2　阻塞发送与非阻塞发送

为了简化问题，假设我们的讨论基于分布存储式并行计算机，采用消息传递的通信方式，其通信接口集有：点点通信：send/receive；群集通信：broadcast，scatter，gather 等，其语义见 MPI[3]。

3. 通信延迟分析

在分布存储式并行系统中，互联网络的带宽有某一确定值，因此，无论在点点通信还是在群集通信中，当通信量达到某一确定值，会取得最佳通信效果（最佳通信流量），这与通信的拥挤度有很大关系。我们知道，在用户层，通信效率受环境因素的影响很大，这些因素包括处理机数、通信函数、信息报长度、通信的拥挤度等。

按照经验，当进程之间进行少量数据传输时，带宽的使用率较低；当进程之间的传输数据达到硬件带宽的最佳值时，进程之间的通信效率最高；当进程之间传输的数据量远远超出其物理带宽时，其传输量进入拥挤状态。下面是我们在 DAWN 1000A 机器上所测得的结果。在图 3 和图 4 中，发送数据包的长度为 2^i，其中 i 为 x 轴的值，bandwidth 的单位为 MB/s，时间为 ms，其数据分别在 2、4、8 个节点所测。从这些数据，我们可以得到如下结论：

(1)*有效带宽问题*　当数据包长度一定时，随着通信节点的增加，有效带宽逐渐下降，呈现有规律的变化。

(2)*通信启动时间问题*　无论是点点通信还是群集通信，从发出通信命令到报文在通信线路上开始传输为止，称为通信启动时间。从测试数据可以看出，在少量数据进行通信时，通信启动时间是一个主要因素，因此，减少通信次数是降低通信延迟的主要途径。在图 4 中，第一次发送延迟时间较长，是因为 MPI 做了一些初始化准备工作。

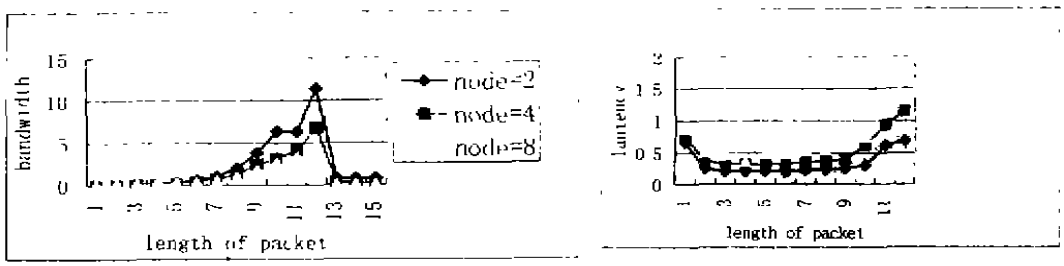

图 3　包长、带宽与节点的关系　　图 4　包长与时间的关系

(3)群集通信　从图 3 可以看出，群集通信的有效带宽会达到三种状态：

①带宽空余状态。它对应传输少量数据的情况，其主要时间用在通信启动时间上，从图 3 和图 4 可以看出，当包长小于 256 字节时，有较多的空余带宽。表 1 给出了达到空余状态的传输包长。

表 1　DAWN 1000A 数据传输状态说明

通信节点数	空余状态数据长度	理想状态数据长度	拥挤状态数据长度
2	4－128 Byte	256－4K Byte	8K 左右或大于 8K
4	4－256 Byte	512－8K Byte	16K 左右或大于 16K
8	4－512 Byte	1024－8K Byte	16K 左右或大于 16K

②理想传输状态。在这种状态下，通信达到理想传输效果，能充分利用硬件带宽。在以上测试数据情况下，达到理想状态的传输数据见表 1。其中，当传输数据在 4K 左右时，达到最佳传输状态。

③拥挤状态。从测试数据可以看出，当传输的包长达到一定值时，其传输时间急剧增加，而传输带宽急剧地降低，这表明传输达到拥挤状态，见图 3。

由此可以看出，在传输少量数据时，应尽量减少通信次数；在传输大量数据时，应尽可能使有效带宽达到最佳值，取得最佳传输效果，避免传输进入拥挤状态。

4. 并行程序设计原则

基于以上的分析，当程序设计者对应用层的指标体系了解以后，就可以以此为基础，设计出效率更高的应用程序。

1)减少通信次数原则　当传送少量数据时，启动时间在通信延迟中起主导作用，因此应尽可能减少通信次数。这可以由通信原语提供的通信机制来实现。例如：

原程序段：
```
for I := 1 to 1000
    send(a[I],1,k)
endfor
```

优化后的程序段
```
send(a,1000,k)
```

原程序段向处理机 k 发送 1000 个数据，每次发送一个，共发送 1000 次，而优化后的程序段一次发送 1000 个数据，其效率会远远优于前者。对于离散数据，在发送进程进行 pack，然后发送；在接受进程进行 unpack，这样也会大大提高效率。

2)传输大量数据原则　对于大量数据的传输，应尽可能在理想状态下传输，即对传送包长和处理机的个数进行仔细的考虑，避免进入拥挤传输状态。

5. 性能分析公式

要评价一个用户程序，目前有多种性能评价模型。按照文[4]中的模型，并行程序的时间可以如下表示：

$$T = T_{comp} + T_{par} + T_{interact}$$

其中，T_{comp}、T_{par}、$T_{interact}$ 分别是计算时间、并行化时间和交互操作时间，计算时间是解题所用的时间；并行时间指进程建立、撤销等所需的时间；交互操作时间指通信准备、通信、同步所需的时间。假设进程采用在任务开始时一次产生、完成任务后撤销模型，因此，我们对进程产生、撤销所需的时间不进行仔细讨论，在此主要讨论进程交互时间。

假设并行程序单节点运行所需时间为 t_1，n 个节点运行应用所需时间为 t_n。当 n 个节点同时运行时，可以认为每个节点的计算工作量相同，因而每个节点的计算时间为：t_1/n。也就是说，当 n 个节点并行计算时，每个节点所需的计算时间为：t_1/n。

在文[4]中，Hockney 给出了点对点通信的时间延迟公式，即：$t(m) = t_0 + m/r_\infty$。其中，t_0 是通信启动时间(μs)，m 是通信量(字节)，r_∞ 是异步带宽(MB/s)，即峰值状态下每秒传输的字节数(MB)。

在文[5]中，对于 Hockney 模型进行了扩充，应用到群集通信中，其通信时间与通信量的关系为：$t(m, n) = t_0(n) + m/r_\infty(n)$，其中 m 是通信量，n 是通信节点数。例如 IBM SP2 的测量结果为：

send/receive　　$t(m) = 46 + 0.035m$
Broadcast　　$t(m,n) = 52\log n + (0.029\log n)m$

根据我们在曙光机上的测量结果，对于点点通信和 BCAST 通信，其通信延迟时间(μs)和通信量 m 及

通信节点数 n 的关系为：

send/receive　　$t_p(m)=t_0+m/r_\infty=62+0.57m$

在理想传输状态下，根据我们测量分析可知 Broadcast 与 send 通信时间的关系为：

$$t_b(m,n)=t_p(m)\log(n) \quad (1)$$

其中 $t_p(m)$ 和 $t_b(m,n)$ 分别表示当包文长度为 m 时，点点通信和 n 个节点间的 Broadcast 通信的时间延迟。现在我们回到评价应用系统的效率上来，根据上面的结果，从节点 i 到节点 j 点点通信所需总的通信时间为：

$$T_{pij}=\sum(t_0+m_{pij}/r_\infty)=c_{pij}t_0+(\sum m_{pij})/r_\infty$$
$$=c_{pij}t_0+M_{pij}/r_\infty$$

其中 c_{pij} 是节点 i 到节点 j 的点点通信次数，t_0 是启动通信时间，M_{pij} 是点点通信总量。上式说明，为了减少通信时间，应减少通信次数。与此类似，我们有 BCAST 所需的通信时间为：

$$T_{bij}=\sum(t_0(n)+m_{bij}/r_\infty(n)) \quad (1)$$

在理想状态下，按照(1)式，我们有：

$$T_{bij}=T_{pij}*\log(n)=(c_{bij}t_0+M_{bij}/r_\infty)\log(n)$$

所以，节点 i 到节点 j 总的通信时间为：

$$T_{interact}(i,j)=T_{pij}+T_{bij}=c_{pij}t_0+M_{pij}/r_\infty+(c_{bi}t_0$$
$$+M_{bij}/r_\infty)\log(n)=(c_{pij}+c_{bij}\log(n))t_0$$
$$+(M_{pij}+M_{bij}*\log(n))/r_\infty \quad (2)$$

式(2)是在两个节点计算量均衡的前提下，由节点 i 到节点 j 的通信延迟估算公式，上式揭示了节点 i 到节点 j 的通信延迟与点点通信次数(c_{pij})、点点通信数据量(M_{pij})、群集通信次数(c_{bij})、群集通信数据量(M_{bij})、通信带宽以及通信启动时间之间的关系，利用该等式，我们可估算通信时间。我们继续分析，节点 i 到组内所有节点的通信延迟为：

$$T_{interact}(i)=\sum T_{interact}(i,j)$$
$$=(c_{pi}+c_{bi}*\log(n))t_0+$$
$$(M_{pi}+M_{bi}*\log(n))/r_\infty \quad (2')$$

其中，c_{pi}、M_{pi}、c_{bi}、M_{bi} 分别为节点 i 到组内所有节点的点点通信次数、点点通信量、群集通信次数和群集通信量，$T_{interact}(i)$ 就是该节点的通信延迟时间，利用式(2)和(2')，我们就可以估算一个节点的通信延迟时间。我们进一步分析，若 c_i 为节点 i 到所有节点的通信次数，M_i 是节点 i 到所有节点的总通信量，那么：节点 i 的通信延迟为到所有节点通信延迟之和：

$$T_{interact}(i)\leqslant\Sigma(1+\log(n)(c_it_0+M_{ij}/r_\infty)$$
$$\leqslant(1+\log(n))(c_it_0+M_i/r_\infty) \quad (2'')$$

从(2'')式看出，某个节点的通信延迟和该节点的通信次数以及该节点总通信量的关系。假设 n 个节点的计算工作量相同，忽略并行时间。这样一来，整个并行系统的运行时间为：

$$T=T_{comp}(i)+\max(T_{interact}(i))\leqslant t_1/n+(1+\log(n))$$
$$\max(c_it_0+M_i/r_\infty)$$

假设，在节点 k 上得到最大值，则：

$$T=T_{comp}(i)+\max(T_{interact}(i))\leqslant t_1/n+(1+\log(n))$$
$$(c_kt_0+M_k/r_\infty)$$

那么系统的加速比为：

$$P=\frac{\text{单结点时间}}{\text{n 个结点时间}}=\frac{t_1}{t_1/n+(1+\log(n))(c_kt_0+M_k/r_\infty)}$$
$$=\frac{n}{1+\frac{n(1+\log(n))(c_kt_0+M_k/r_\infty)}{t_1}} \quad (3)$$

这样，我们得到了加速比与处理机节点数 n、最大节点的通信数据量和通信次数等有关参数的关系。据此模型，我们可以对并行应用程序进行分析。由于在某一机器上 t_0 和 r_∞ 可以具体测出，因此只要求出单机运行时间、发送的数据量和发送次数，就可以估算出其加速比。

根据以上评价公式，我们用矩阵乘积算法进行了验证，其矩阵为 1024×1024，和单节点运行时间(462.0424 秒)进行比较，得到如下测试结果，可以看出，其误差在 15%以下。

节点数	2	4	8	16
通信次数	3072	5120	9216	17408
通信量	20971520	23068672	24059904	24641536
估算加速比	1.972233	3.81055453	6.910847	10.21886
实测时间	239.0082	140.112	76.02739	51.63989
实测加速比	1.933165	3.29766465	6.077315	8.947392
误差估算%	0.02021	0.1555312	0.13715	0.1421

结论　在本文中，我们首先提出了并行计算模型的层次问题，然后从应用程序设计人员的角度，分析了并行计算模型的主要参数。其次，以曙光机为例，通过对有关数据的测试及分析，对带宽的传输状态分为空余状态、理想状态和拥挤状态，在此基础上讨论了并行程序设计原则。接下来，以性能加速比为出发点，讨论了通讯时间问题，并提出了一组性能评价公式。通过验证，这组公式能较好地反映并行应用的实际情况。

参考文献

1 Culler D, et al. LogP: Towards a realistic model of parallel computation In: Proc 4th ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming. 1993

2 Hambrusch S E, Khokhar A A. C^3: A Parallel Model for Coarse-Grained Machines. J of Parallel and Distributed Computing, 1996, 32

3 Gropp W, Lusk E. A high-performance portable implementation of the MPI message passing interface standard. Parallel Computing, 1996, 22

4 Hockney R W. A Framework for Benchmarks. Advances in Parallel Computing, 1993, 8

5 Hwang Kai, Xu Zhiwei Scalable Parallel Computing. China Machine Press, 1999