文章编号：1004－5694(2001)03－0001－04

阶段并行模型上的并行算法设计*

尚明生[1]，刘宴兵[2]，孙世新[1]
(1.电子科技大学 计算机系，成都 610054；2.重庆邮电学院，重庆 400065)

摘　要：阶段并行模型是BSP(Bulk Synchronous Parallel)模型的改进，它更接近于表述实际的处理器行为，同时具有编程简单、独立于体系结构和执行性能可预测等特点。NOWs正成为并行计算领域的一个新的发展热点，以太网构成的微机集群系统是NOWs的一种重要实现形式。研究了集群系统中阶段并行模型上的并行算法设计，以FFT算法为例，进行了设计和分析，并给出了测试结果。

关键词：BSP模型；阶段并行模型；集群系统；FFT算法
中图分类号：TP311　　**文献标识码**：A

Designing Parallel Algorithms Based on the Phased Parallel Model

SHANG Ming-sheng[1]，LIU Yan-bing[2]，SUN Shi-xin[1]
(1. *School of Computer Science*，*UEST of China*，*Chengdu* 610054，*China*；2. *Dept. of Computer Science*，*Chongqing University of Posts and Telecommunications*，*Chongqing* 400065，*China*)

Abstract：Phased parallel model is the improvement of BSP model，which is more suitable for describing actual processor's action for it's simple program design，independence of architecture and predictable execution performance. NOWs becomes a new focus in parallel computation now，and the cluster system of PCs on Ethernet is one form of basic realization. This paper is concerned with the design of parallel algorithms in the cluster system on phased parallel model. The design and analysis of FFT algorithm are studied and the experiment results are presented.

Key words：BSP model；phased parallel model；cluster system；FFT algorithm

0　引　言

近年来，工作站机群(NOWs：Network of Workstations)由于具有较多优点而成为并行计算的一个新的发展热点[1]，它采用消息传递方式进行编程。常见的PRAM模型是一种共享存储的同步计算模型，它假定访存时没有通信开销，故不适合消息传递编程。LogP模型虽然比较符合消息传递的实际情况，但用它来指导算法设计需要很强的技巧性，而设计技巧基本上无章可循[2]。其它几种模型如LogGP，C^3等仍不能保证程序确定性，且不易编程实现。

阶段并行模型是块同步并行(BSP：Bulk Synchronous Parallel)模型的改进，它具有BSP模型编程简单、独立于体系结构、较好的可伸缩性和执行性

* **收稿日期**：2001－06－04
基金项目："九五"国防科技预研项目(16.1.4.1)和国家高性能计算基金(99405)
作者简介：尚明生(1973-)，男，重庆大足县人，四川师范学院讲师，现为电子科技大学硕士生，主要研究方向：网络及并行计算。孙世新，教授，博导。

能可预测等特点，并考虑了各种开销，因而更接近于描述实际的并行计算机行为[3]。阶段并行模型能在绝大多数并行体系结构下有效地指导算法的设计与分析。

本文首先介绍 BSP 模型，然后讨论阶段并行模型对 BSP 模型的改进，最后在网络群集系统中，以 FFT 并行算法为例，给出相应的算法设计并在实验室机群系统和曙光 2000 上进行了测试。

1　BSP 模型

BSP 模型是由哈佛大学 Leslie Valiant 提出的，一个 BSP 计算机由 n 个处理机/存储器组成，通过通信网络进行互联，如图 1 所示。

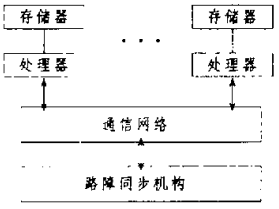

图 1　BSP模型

Fig. 1 BSP model

一个 BSP 程序有 n 个进程，每个驻留在一个节点上，程序按严格的超步(super step)顺序执行，超步间采用路障同步。每个超步分成如下有序的 3 个部分。

(1) 计算：一个或多个处理器执行若干局部计算操作。操作的所有数据只能是局部存储器中的数据，这些数据是在程序启动时或由以前超步中的通信操作存放在局部存储器中的。一个进程的计算与其它进程无关。

(2) 通信：处理器之间相互交换数据，通信总是以点对点的方式进行。

(3) 同步：确保通信过程中交换的数据被传送到目的处理器上，并使一个超步中的计算和通信操作必须全部完成之后，才能开始下一个超步中的任何动作。

BSP 模型总的执行时间等于各个超步执行时间之和，每个超步执行时间确定如下。

(1) 计算时间 w：考虑到负载可能不平衡，或者各个处理器性能差异，w 是指处理器中完成计算操作所需的最大时间。即 $w = \max(w_i)$，w_i 是超步内各个处理器执行时间。由此可见，BSP 模型对异构处理器也适用。

(2) 通信时间 gh：g 与平台有关，与通信模式无关，是衡量网络通信速度的一个参数，可以理解为传送单位长度的消息所需的时间。h 可认为是一个超步中的最大通信量。$h = \max(h_i)$，其中，h 表示处理器 i 发送或接收的数据量。

(3) 同步时间 l：l 是完成一次同步的代价，其下限为网络通信迟延，其值总大于零。

一个超步的总的时间为：$w + gh + l$。总的执行时间为：$(w + gh + l)N$，N 为超步数。如果超步内 3 个操作完全重叠，则超步时间可达到 $\max\{w, gh, l\}$，总的执行时间可达 $\max(w, gh, l)N$。

BSP 模型设计的程序简单，具有良好的结构，它的最大的问题是认为各个超步执行时间完全相同，这虽然减少了算法分析的复杂度，但同时使得 BSP 模型刻划的并行机性能有时不准确。

2　阶段并行模型对 BSP 模型的改进

阶段并行计算模型[3]与 BSP 模型类似，认为一个并行程序由一系列超步组成，每个超步最多有 3 个顺序阶段(可以没有部分阶段，但不能全部没有)，3 个阶段定义如下：并行化阶段 —— 完成并行处理所需的进程创建和分组等进程开销阶段。计算阶段 —— 一个或多个处理器执行若干局部计算操作。交互阶段 —— 完成交互操作所需的通信、同步等操作。

阶段并行计算模型对超步时间开销的计算如下。

(1) 并行化开销 t_p 为创建进程所需时间。

(2) 计算开销用负载大小来表示，记为 w_i。与 BSP 模型不同，在不同的计算阶段可有不同的工作负载，从而执行时间可以不同。

(3) 交互开销 $T_{interact}$ 在不同的超步中可以不同。设 m 为消息长度，t_0 表示启动时间，t_1 表示单位消息的通信时间，则交互开销可以表示为：$T_{interact} = t_0 + t_1 n$。

令 b 为每个处理器工作负载的标准偏差，则 n 个处理器上执行一个超步所需的时间为：

$$t = (w + b\sqrt{2\log n}) + t_0 + mt_1 + t_p$$

由以上公式可以看出，利用阶段并行模型设计算法要从以下方面着手。

(1) 由于平均负载 w 无法减少，故负载平衡偏差 b 应尽量减少，因此在任务分配时要考虑负载平衡。对同构计算机只需按任务总量平均分配到各个处理器即可；对异构网络，则要根据各个处理器的计算能力恰当的分配任务。

(2) 在交互开销方面，由于消息传送在交互阶段集中完成，所有要传送消息已经组合成一个包，故不能减少 t_0，因此要尽量减少消息传送量。在进行数据分配时要考虑到数据相关性，尽可能做到相关数据局部化，从而减少消息传递数量。

(3) 并行化开销 t_p 的减少可通过减少超步数来实现，而超步数的减少可能会影响并行效度，因此要考虑合适的超步数量。

在以太网构成的 NOWs 中，处理器的计算速度一般都大大超过了网络的通信速度，而且由于以太网采用总线拓扑和 CSMA/CD 协议，通信最容易成为瓶颈，因此应特别注意尽量减少通信代价。

3 阶段并行模型上的 FFT 算法

FFT 算法具有很好的并行性，在 n 个节点网络上实现 n 个输入的 FFT 算法只需 $O(\log n)$ 步，已经是最优的，且达到线性加速。

$n=8$ 的 FFT 基 2 蝶式计算如图 2 所示。行号为 0～$(n-1)$，列号为 0～$\log(n)$。第 r 行第 i 列的节点分别连到第 r 行第 $i+1$ 列的节点和第 r' 行第 $i+1$ 列的节点上，其中

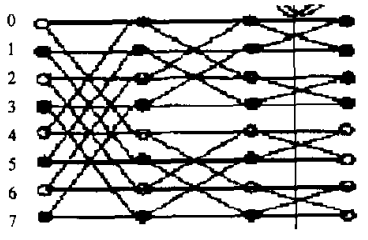

图 2 $n=8, p=2$ 的蝶式计算

Fig. 2 $n=8, p=2$ butterfly computation

$$r' = \begin{cases} r + \dfrac{n}{2^{i+1}}, & \dfrac{kn}{2^i} \leqslant r < (2k+1)\dfrac{n}{2^{i+1}} \\ r - \dfrac{n}{2^{i+1}}, & (2k+1)\dfrac{kn}{2^i} \leqslant r < (k+1)\dfrac{n}{2^i} \end{cases}$$

式中，$k = 0, 1, \cdots, \log n - i$。其串行复杂度为 $n\log(n)$。

在循环分配下，蝶形图的前 $\log(n/p)$ 级全是局部计算，后 $\log(p)$ 级要进行数据交换；成块分配下，前 $\log(p)$ 级要进行数据交换，后 $\log(n/p)$ 级全是局部计算，无数据交换。因此若 $n \geqslant p^2$ 且能被 p 整除，考虑在前 $\log(n/p)$ 级进行循环分配，后 $\log(p)$ 级成块分配。这样只需在前 $\log(n/p)$ 级计算完成后，进行一次数据交换，将循环分配转为成块分配，然后再进行其余 $\log(p)$ 级的计算，便可做到整个 FFT 过程中只需一次数据交换。对于 FFT 算法，假设 $n \geqslant p^2$ 是合理的。

在混合分配策略下，局部计算部分，单个节点上的过程与串行算法完全相同。节点机之间进行数据交换时，只需将 FFT 蝶形图的第 $\log(n/p)$ 级中的第 $k(n/p) + i(k = 0, \cdots, p-1; i = 0, \cdots, n/p - 1)$ 号点传给第 k 号节点即可。由于在后 $\log p$ 级的计算中，每个节点需要 n/p 个初始值，但其中 $1/p$ 的点本来就在其上，所以每个节点共接收 $(n/p)(1 - 1/p)$ 个点。另外，一个节点向其余每个节点传 n/p^2 个点，共发送 $(n/p^2)(p-1) = n/p(1 - 1/p)$ 个消息。

在阶段并行模型中，数据采用混合分配策略，已经做到负载平衡，交互开销传送的数据最少。对于超步的个数，可分为 3 个超步。

(1) 初始化：由 2 个阶段完成，并行化阶段创建 p 个进程，时间开销 t_p；交互阶段将初始数据分配到各个处理器，时间开销为 $t_0 + t_1 n$。

(2) 计算前 $\log(n/p)$ 列：由 2 个阶段完成，计算阶段完成本地数据计算任务，时间开销为 $(n/p)\log(n/p)$；交互阶段进行数据交换，时间开销为 $t_0 + t_1(n/p)(1 - 1/p)$。

(3) 计算后 $\log p$ 列：由 2 个阶段完成，计算阶段完成本地数据计算任务，时间开销为 $(n/p)\log(p)$；交互阶段将计算结果送回主节点，时间开销为 $t_0 + t_1 n$。

整个计算的时间开销为：

$$t = \frac{n}{p}\log n + 3t_0 + \left(2n + \frac{n}{p} - \frac{n}{p^2}\right)t_1 + t_p$$

加速比 $S_p = O(n)$，效率 $E_p = O(1)$。

4　实验结果

我们在UNIX操作系统的环境下使用并行计算平台 PVM 对设计的算法编程实现，并测试了其结果。实验室机器配置为 CPU：k6-266；内存为 32 M；操作系统为 SCO UNIX 5.0.02；语言为 C 语言；并行计算环境为 PVM 3.3.11。网络为 10 Mbit/s Ethernet。程序采用 Master/Slave 结构，选定其中的一台为主节点机，而剩下的则为从节点机。表 1 是在 PC 机群上的实验结果，表 2 是在曙光 2000 上的测试结果。

表 1　FFT 变换在 PC 机群中的测试结果

Tab. 1 The testing results of FFT transformation in PC group

问题规模 N（2 的幂）	串行时间(s)	并行(机器数=2)			并行(机器数=4)		
		时间(s)	平均加速比	平均效率	时间(s)	平均加速比	平均效率
13	0.046	0.056	0.710	0.355	0.030	1.332	0.333
14	0.090	0.102	0.880	0.440	0.060	1.504	0.376
15	0.200	0.172	1.164	0.582	0.104	1.916	0.479
16	0.490	0.393	1.246	0.623	0.213	2.304	0.576
17	1.200	0.762	1.574	0.787	0.448	2.680	0.670
18	2.650	1.864	1.422	0.711	0.906	2.920	0.730
19	5.690	3.887	1.464	0.732	1.998	2.848	0.712

表 2　FFT 变换在曙光 2000 上的测试结果

Tab. 2 The testing results of FFT transformation in shuguang2000

问题规模 N（2 的幂）	串行时间(s)	并行(处理器个数=2)			并行(处理器个数=4)		
		时间(s)	平均加速比	平均效率	时间(s)	平均加速比	平均效率
13	0.018	0.021	0.861	0.430	0.015	1.360	0.340
14	0.044	0.041	1.090	0.545	0.022	1.968	0.492
15	0.096	0.071	1.344	0.672	0.037	2.388	0.597
16	0.233	0.157	1.486	0.743	0.098	2.776	0.694
17	0.586	0.352	1.666	0.833	0.185	3.156	0.798
18	1.333	0.739	1.804	0.902	0.419	3.403	0.851
19	3.034	1.756	1.728	0.864	0.926	3.252	0.813

6　结束语

阶段并行模型介于完全串行的模型和严格同步的模型之间，它考虑了所有的开销类型：负载不平衡开销、交互开销和并行化开销，因而能较准确的刻划实际并行机的行为，同时具有很好的可扩展性，它的程序和串行程序有类似的风格，易于进行并行算法分析、设计和实现，非常适宜于群集系统。

对异构系统，由于涉及到各个处理器能力差异，需要更多的平衡策略，已有的一些文献对此进行了讨论，但基本上都假设任务可以均匀划分（实际上在很多问题上任务是不能均匀划分的），而且得到的结果是在各个处理器速度相同时达到最优，因此如何在任务不可均匀划分的异构处理器上进行基于阶段并行模型的算法设计将是我们今后研究的重点。

参　考　文　献

[1] 李晓梅，莫则尧，胡庆凤，等. 可扩展并行算法的设计与分析[M]. 北京：国防工业出版社，2000.

[2] 黄伟民，陆鑫达，曾国荪. 异构 BSP 模型及其通信协议[J]. 电子学报，2000，28(8)：72-75.

[3] 黄铠，许志伟. 可扩展并行计算技术、结构与编程[M]. 北京：机械工业出版社，2000.

[4] 刘久星，孙永强. PRAM，BSP 和 LogP 并行模型之间的关系及其比较[J]. 小型微型计算机系统，1999，20(11)：824-827.

[5] 雷州，徐志伟，祝明发. 系统负载与并行程序运行时间的关系[J]. 计算机研究与发展，2000，37(7)：813-817.

[6] 李晓峰，寿标. LogP 模型的改进与 FFT 算法的优化设计[J]. 计算机研究与发展，1996，33(6)：438-444.

[7] 许锦波，顾乃杰，陈国良. C^3 模型上 FFT 算法的设计与分析[J]. 计算机研究与发展，1997，34(Suppl)：59-64.

[8] 孙家旭，张林波，迟学斌，等. 网络并行计算与分布式编程环境[M]. 北京：科学出版社，1996.

（编辑：何先刚）