

# 异构 BSP 模型及其通信协议

黄伟民, 陆鑫达, 曾国荪

(上海交通大学计算机科学与工程系, 上海 200030)

**摘 要:** 异构并行计算由于其较高性能价格比而在高性能科学计算和通用应用领域受到广泛研究. 但由于异构并行程序设计与性能分析仍处于经验阶段, 开发实用程序较为困难. 本文提出异构环境中的 HBSP 模型, 并导出相应的开销预测方法, 能有效指导异构并行程序的设计与分析. 所设计并实现的 HBSP 模型的通信协议能运行于所有支持 MPICH 软件包的计算平台. 最后以并行 FFT 算法为例, 给出相应的算法设计和实际测试结果.

**关键词:** 异构计算; BSP 模型; 并行计算模型

**中图分类号:** TP393

**文献标识码:** A

**文章编号:** 0372-2112 (2000) 08-0072-04

## Heterogeneous Bulk Synchronous Parallel Model and Its Message Passing Protocols

HUANG Wei-min, LU Xin-da, ZENG Guo-sun

(Dept. of Computer Science, Shanghai Jiao Tong Univ., Shanghai 200030, China)

**Abstract:** Heterogeneous computing has been widely introduced in high-performance scientific computing as well as in "general purpose" applications due to its higher performance cost ratio. However, the application program is difficult to develop because heterogeneous parallel programming and performance analysis are still empirical. In this paper, Heterogeneous Bulk Synchronous Parallel (HBSP) model is proposed and its cost model is derived. The message passing protocol of HBSP is designed and realized in all kinds of machines which support MPICH package. The FFT algorithm is studied as an example and the experimental results of FFT in several heterogeneous systems are discussed.

**Key words:** heterogeneous computing; bulk synchronous parallel model; parallel computing model

### 1 引言

由多个独立计算系统通过网络构成的异构系统群, 可以转变成一个一致的、功能强大的和较高性能价格比的并行计算资源. 近年来, 异构并行计算在高性能科学计算和通用的应用领域受到广泛的研究<sup>[1]</sup>. 在异构网络计算中, 常采用消息传递方式进行编程, 但缺乏一种更自然和简单的计算模型来解决这一问题.

常见的 PRAM 模型是一种共享存储的同步计算模型, 假定访问共享内存为单位时间, 没有通信开销, 故不适合于对基于消息传递的并行计算机. Log P 模型虽然比较符合基于消息传递的并行机的实际情况, 但用它来指导算法设计需要很强的技巧性, 要对每个处理器的局部计算步进行仔细安排, 以在网络通信能力的限制范围内将通信操作和计算操作尽可能重叠起来. 此外还要妥善分配处理器, 以减少对通信带宽的要求和降低远程访问的频率. 因此 Log P 算法的设计技巧基本上无章可循. Log GP<sup>[2]</sup>模型是对 Log P 模型的扩展, 增加了一个参数 G, 用以描述发送长消息时的网络带宽. 与 Log P 相比,

Log GP 算法的执行速度和性能有较大提高, 但随着模型的进一步复杂化, 算法设计和分析也更为复杂. BSP 模型指导下的并行程序设计<sup>[3]</sup>具有编程简单、独立于体系结构和执行性能可预测等特点, 但仍存在要求机间通信满足  $h$ -relation 条件,  $g$  的测算不够精确, 在异构计算环境下不再适合等问题. 针对上述问题, 本文首先给出适合于异构网络计算环境中的 HBSP 模型和开销计算方法. 在第 3 节简要介绍 HBSP 模型的通信协议的设计与实现. 在第 4 节中, 以并行 FFT 算法为例, 给出相应的算法设计和实际测试结果.

### 2 HBSP 模型和开销计算

HBSP 模型中计算由一系列连续的超步组成. 每个超步中, 每个处理单元使用超步开始前已存放在本地的数据进行计算和处理, 这些任务包括: (1) 对存放在本地的数据进行计算; (2) 消息传递; (3) 消息接收. 对于异构计算系统  $P = \{P_1, P_2, \dots, P_p\}$ , 其中处理机数  $p = |P|$ , 处理机速度参数集合  $S = \{S_1, S_2, \dots, S_p\}$ , 任务分配  $W = \{w_1, w_2, \dots, w_p\}$ , 通信量分配

集合  $H = \{h_1, h_2, \dots, h_p\}$ . 令  $s_j$  为处理机  $P_j$  的计算速度,  $b_j$  为处理机忙闲度,  $s_j = s_j \cdot b_j$ , 为实际计算速度.  $w_j^t = w_j / s_j$ , 为处理机  $P_j$  完成任务  $w_j$  所需时间.  $g^t$  为发送一个字节消息所需时间,  $L^t$  为处理机完成 Barrier 同步所需时间.

这样,第  $i$  个超步的执行时间

$$T(x) = T_{\text{localproc}_i} + T_{\text{sync}_i} = \max_{j=1}^p w_j^t + h_j \cdot g^t + L^t \quad (1)$$

每个超步的开销由每个处理器上本地处理开销(包括局部计算和机间通信)的最大值和超步结束时的同步开销两部分组成:

$$C(x) = \max_{i=1}^p w_i + h_i \cdot g + L \quad (2)$$

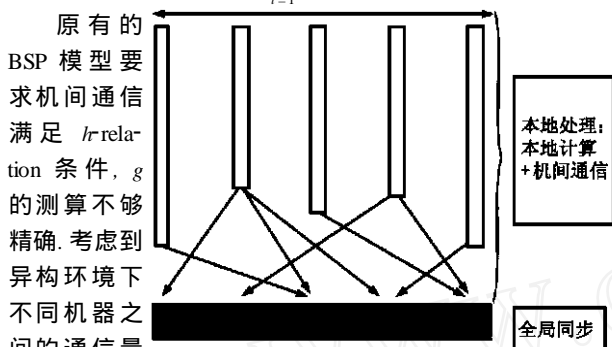


图 1 异构 BSP 模型和开销计算

现有的并行机对于长消息提供特殊的支持,故发送长消息的带宽要远大于发送短消息的带宽.为此,我们在 HBSP 模型中,用线性公式来描述发送  $n$  字节长消息所用的时间:  $t = t_0 + t_b \cdot n$ , 其中  $t_0$  是将一个消息发往目标处理机的通信启动开销时间,  $t_b$  是传送每字节所用时间(参见第 3 节).

不失一般性,假设处理机集合  $P = \{P_1, P_2, \dots, P_p\}$  通过同一网络连接而构成一个异构系统,各处理机发送单位字节所需的通信开销  $g$  基本相同.设处理机实际速度参数为  $S = \{S_1, S_2, \dots, S_p\}$ , 令最慢处理机的速度  $S_{\min} = \min_{i=1}^p S_i$ , 则处理机完成 Barrier 同步所需时间为  $L^t = L / S_{\min}$ .

令处理机相对速度比率  $P^* = (P_1^*, P_2^*, \dots, P_p^*)$ , 其中  $P_i^* = S_i / S_{\min}$ , 则由(1)可得:

$$T(x) = \left( \max_{j=1}^p w_j / P_j^* + (p-1) \cdot h_0 / P_j^* + h_j \cdot g + L \right) / S_{\min}$$

通信启动开销  $(p-1) \cdot h_0$  只与处理机数目相关,在问题规模扩大时可以忽略,上式就简化为:

$$T(x) = \left( \max_{j=1}^p w_j / P_j^* + h_j \cdot g + L \right) / S_{\min} \quad (3)$$

处理机  $P_i$  利用率

$$U_i = \frac{w_i^t}{\max_{j=1}^p w_j^t + h_j \cdot g^t + L^t} = \frac{w_i / P_i^*}{\max_{j=1}^p w_j / P_j^* + h_j \cdot g + L} \quad (4)$$

根据 HBSP 模型,我们可得出编写高效异构 BSP 程序的策略:

- (1) 使超步的数目最小化,从而可以减少  $L$  的次数.
- (2) 合并发往相同处理器的多个消息,使每个超步的通信

启动开销至多为  $p-1$  次.

(3) 由于每个超步中的 Barrier 同步必须等待最慢的进程,要使  $T(x)$  最小,就应使各处理机的本地计算时间和处理机间通信时间之和尽量均衡.

显然有:

$$\begin{cases} \sum_{i=1}^p w_i = W \\ W_i / P_i^* + h_i \cdot g = W_j / P_j^* + h_j \cdot g, \forall i, j \leq p \end{cases} \quad (5)$$

上式成立时的任务分配  $W$  使  $T(x)$  最小.

### 3 HBSP 模型的通信协议及其实现

为实现 BSP 模型,在 1998 年 7 月发布的 BSPlib V1.4 软件包上,进程间的消息传递是由 5 条原语实现的,其中发送消息的两条原语为: `bsp_set_tagsize(int *tag, nbytes)`, 用于设置消息的标记位; `bsp_send(int pid, const void *tag, const void *payload, int payload, nbytes)`, 用于发送消息到远程队列中.接收消息需要以下三条原语:首先调用 `bsp_qsize(int *mmessages, int *accum, nbytes)`, 返回队列中的消息数;然后调用 `bsp_get_tag(int *status, void *tag)`, 取出消息中的标记位;最后调用 `bsp_move(void *payload, int reception, nbytes)`, 从队列中取出消息.超步之间的 Barrier 同步是由 `bsp_sync(void)` 原语实现,对所有进程进行全局同步.

在 BSPlib 中,消息的发送和接收使用起来比较复杂,在一个超步内消息的标记部分需一次设定,大小不能改变,消息收发原语使用起来较为复杂;同时 Barrier 同步是对所有进程进行全局同步,不能按照程序设计的需要对特定的进程进行同步.为解决这一问题,我们设计并实现了 HBSP 模型的通信协议,该协议支持异构环境,能运行于所有支持 MPICH 的计算平台,并具有较好的执行效率和可移植性.在 HBSP 通信协议中,进程间的消息传递由 2 条原语实现: `hbsp_send(buf, count, datatype, tag, dest)` 用于发送消息到其它进程中, `hbsp_recv(buf, count, datatype, tag, source)` 用于从其它进程中接收消息.其中 `buf` 为指针,指向发送或接收消息缓冲区的起始地址; `count` 为整型变量,说明发送或接收消息的数量; `datatype` 说明发送或接收消息的类型; `dest` 说明消息发送的目的地, `source` 说明从哪个进程接收消息; `tag` 为消息的标记位,用于区分发往同一个进程的两个消息.超步之间的 Barrier 同步是由 `hbsp_sync(n, ranks)` 原语实现.其中 `ranks` 为数组类型,用于存放需要同步的进程的序列号, `n` 说明 `ranks` 数组中的进程数目,当  $n < 0$  时,将对所有进程进行全局同步.

表 1 HBSP 的通信开销

处理机	发送 char 的斜率	发送 int 的斜率	发送 double 的斜率
USparc-Ultra30	0.114	0.406	0.756
USparc-USparc	0.112	0.400	0.746
E450-USparc	0.118	0.401	0.756

在由 {SunE450, SunUltra30, SunUltraSparc} 等多种机器通过 100Mb Ethernet 构成的异构计算系统上,采用 HBSP 通信原语测试不同性能处理机间的通信速度和通信启动开销.图 2 中 HBSP 通信原语的实际测试结果表明发送不同类型消息时(从

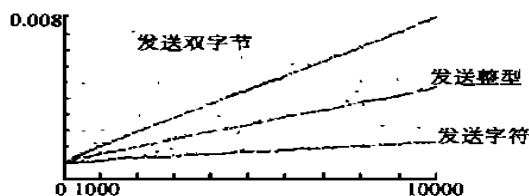
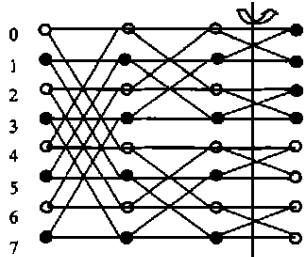


图2 HBSP 通信性能测试结果

char, integer 到 double), 通信时间基本是消息大小的线性函数。表 1 中当 HBSP 通信原语在不同类型处理机之间发送消息时

(E450, UltraSparc, Ultra30 之间), 表示通信速度的斜率基本变化不大, 说明在处理机集合通过同一网络连接而构成的异构系统中, 各处理机的通信开销基本相同。限于篇幅, 我们将在其他文章中详细介绍 HBSP 模型的通信协议及其实现方法, 这里不再赘述。

图3  $n=8, p=2$  的 FFT 蝶式计算

#### 4 HBSP 模型上的 FFT 算法设计和分析

对于图 3 所示的  $n=8$  的 FFT 蝶式计算图, 行号为 0 到  $(n-1)$ , 列号为 0 到  $\log(n)$ . 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  $i+1$  列节点,  $r = r + \frac{n}{2^{i+1}}, kn/2^i \leq r < (2k+1)n/2^{i+1}$ , 其中  $k=0, 1, \dots, \log n - i$ .  $n$  点 FFT 算法的串行时间复杂度为  $O(n \log n)$ .

设处理机数为  $p$ , 通常的 BSP 算法将  $n$  点的蝶式计算图

分成  $\lceil \log n / \log d \rceil$  ( $d = n/p$ ) 个连续的超步, 每个超步由  $n/d$  个各自独立的  $d$  点 FFT 蝶式图组成. 超步内每个处理机有  $d \log d$  个局部操作, 并且每个处理机发送和接收  $d$  个消息. 整个计算开销为  $\lceil \log n / \log d \rceil \cdot (d \log d + d \cdot g + L) = d \log n + d \cdot g \log n / \log d + L \log n / \log d$ . 为减少同步与通信开销, 我们将蝶形计算图的前  $\log(n/p)$  列作为一个超步, 而后  $\log p$  列的蝶式计算作为第二个超步. 对第一个超步的起始计算节点, 采用循环指派的方法, 将节点  $i$  分配给处理机  $i \pmod{p}$ . 对第二个超步采用按块分配的方法, 节点  $i$  分配给处理机  $[i * p / n]$ . 每

对处理机之间需要收发  $\frac{n}{p} - \frac{n}{p^2}$  个消息. 整个计算开销为:  $\frac{n}{p} \cdot \log n + \left( \frac{n}{p} - \frac{n}{p^2} \right) \cdot g + L$ .

在处理机速度相同的并行机上所需的计算时间可直接从上述开销中得出, 当处理机速度不同时, 任务的完成时间实际上是由速度最慢的处理机的完成时间所决定的, 即:  $T(x) = C(x) / \min\{S_i\}$ , 所以处理机  $P_i$  的利用率  $U_i = \left( \frac{n}{p} \log n \cdot \frac{1}{P_i^*} \right) / C(x)$ .

为使异构计算系统中并行 FFT 算法的  $T(x)$  最小, 需要使各处理机的本地计算时间和处理机间通信时间之和尽量均衡. 由于总的任务负载为  $n \log n$ , 由式 (5) 我们得出使  $T(x)$  最小的任务分配方案  $w$ : 每个处理机  $P_i$  分配节点数为  $(P_i^* \cdot n) / p$ , 每个处理机  $P_i$  收发消息数  $P_i^* \cdot \frac{n}{p} \cdot \frac{p}{p-1} = P_i^* \cdot n \cdot (p - P_i^*) / p^2$  时, 算法开销为:

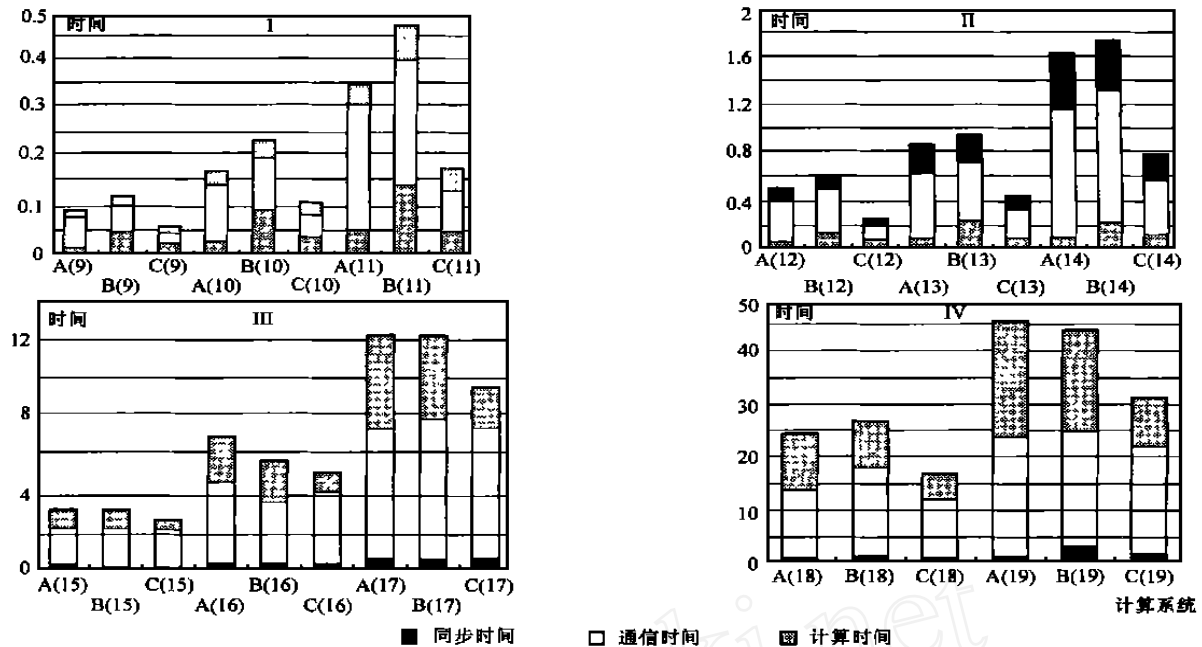
$C(x) = \max_{i=1}^p \left\{ \frac{n \cdot P_i^*}{P} \log n / P_i^* + \frac{P_i^* \cdot n \cdot (p - P_i^*)}{p^2} \cdot g \right\} + L$ , 处理机利用率  $U_i = \left( \frac{(P_i^* \cdot n)}{p} \log n \cdot \frac{1}{P_i^*} \right) / C(x)$ .

表2 异构系统中并行 FFT 算法的预测性能

计算系统	{USparc, USparc, USparc, USparc} 基于原 BSP 模型	{E450, USparc, USparc, USparc} 基于原 BSP 模型	{E450, USparc, USparc, USparc, USparc} 基于 HBSP 模型
计算性能	$P^* = \{4, 1, 1, 1\}$	$P^* = \{4, 1, 1, 1\}$	$P^* = \{4, 1, 1, 1, 1\}$
任务分配 $w$	$\left\{ \frac{n \log n}{4}, \frac{n \log n}{4}, \frac{n \log n}{4}, \frac{n \log n}{4} \right\}$	$\left\{ \frac{n \log n}{4}, \frac{n \log n}{4}, \frac{n \log n}{4}, \frac{n \log n}{4} \right\}$	$\left\{ \frac{n \log n}{2}, \frac{n \log n}{8}, \frac{n \log n}{8}, \frac{n \log n}{8}, \frac{n \log n}{8} \right\}$
消息收发 $H$	$\left( \frac{3n}{16}, \frac{3n}{16}, \frac{3n}{16}, \frac{3n}{16} \right)$	$\left( \frac{3n}{16}, \frac{3n}{16}, \frac{3n}{16}, \frac{3n}{16} \right)$	$\left( \frac{n}{4}, \frac{7n}{64}, \frac{7n}{64}, \frac{7n}{64}, \frac{7n}{64} \right)$
算法开销	$\frac{n \log n}{4} + \frac{3ng}{16} + L$	$\frac{n \log n}{4} + \frac{3ng}{16} + L$	$\frac{n \log n}{8} + \frac{ng}{4} + L$
处理机利用率	$\log n \gg g$ 时约为 $(1, 1, 1, 1)$	$\log n \gg g$ 时约为 $(1/4, 1, 1, 1)$	$\log n \gg g$ 时约为 $(1, 1, 1, 1, 1)$

<例>. 计算系统由 Sun E450 和 4 台 Sun UltraSparc (USparc) 构成, 其中 E450 含有 4 个 CPU, 主频为 300MHz, UltraSparc 主频为 143MHz. E450 作为服务器, 实际计算能力约为 USparc 的四倍. Sun USparc 的速度为  $S_{USparc}$ , 表 2 给出在 BSP 和 HBSP 模型上设计 FFT 算法的结果. 我们采用 HBSP 原语编制了并行 FFT 程序并在上述环境中进行实际测试, 图 4 给出 FFT 程序计算、通信和同步时间测试结果, 其中 X 轴表示三种计算系统及其所计算 FFT 问题规模的对数 (括号内, 从 9 到

19), Y 轴为时间. 为清晰起见, 我们将图分解成、和四个部分. 计算时间基本上达到线性加速, 通信时间三种并行机基本相同; 由 E450 和 3 台 USparc 构成的计算系统同步时间最大, 这主要是由于算法设计时未考虑机器速度的差异, 导致同步时间增大. 由于 FFT 算法属于通信密集型计算任务, 并行计算时较难达到线性加速比, 在问题规模较小时通信时间甚至超过计算时间. 但随着问题规模的增大, 并行处理的优越性逐渐显现. 实际应用中的 FFT 常用来处理卫星图片等巨量数



注 A: 4 台 USparc; B: E450 & 3 USparc; C: E450 & 4 USparc 括号内数值为 FFT 问题规模的对数

图 4 异构系统中并行 FFT 算法计算、通信和同步时间的实际测试结果

据,将会取得较好的加速比。总结起来,性能预测和实际测试结果基本相同,说明 HBSP 模型能够有效地指导和预测异构并行算法的设计与实现。

## 5 结束语

近年来,异构并行计算在高性能科学计算和通用的应用领域受到广泛的研究。本文给出异构计算环境中的 HBSP 模型和开销计算方法,简要介绍了 HBSP 的通信协议及其实现。以 FFT 算法为例,给出相应的算法设计和分析结果,并在异构环境中实际测试,得出结论:HBSP 模型能够有效地指导和预测异构环境中并行算法的设计与实现。

原 BSP 模型要求处理机间通信满足  $1/r$  relation 关系,HBSP 模型中则不再有此限制,从而为算法和程序设计提供了更多的自由。在处理机速度相同的情况下,即  $P_i^* = 1 (i = 1, \dots, p)$  时,且各处理机上所分配的计算量通信量完全均衡时,式(1)和(3)可转化为 BSP 的标准开销模型<sup>[3]</sup>。

## 参考文献:

- [1] V. Sunderam. Heterogeneous network computing: The next generation [J]. Journal of Parallel Computing, 1997, 23: 121 - 135.

- [2] A. Alexandrov. LogGP: Incorporating long messages into the LogP model [J]. In 7th Annual Symp. on Parallel Algorithms and Architectures, 1995: 159 - 168.
- [3] W. F. McColl. Scalability, Portability and Predictability: The BSP Approach to Parallel Programming [J]. Journal of Future Generation Computer Systems, 1996, 12: 265 - 272.

## 作者简介:



黄伟民 1971 年生, 1991 年获武汉大学学士学位, 1996 年获中国科技大学硕士学位, 1999 年获上海交通大学博士学位, 主要研究领域为并行与分布式处理。

陆鑫达 上海交通大学计算机科学与工程系教授, 博士生导师。1964 年哈尔滨工业大学计算机专业研究生毕业。1979 ~ 1981 年为英国纽卡舍尔大学计算机系访问学者, 1987 ~ 1990 年为德国 GMD-FIRST 计算机研究所和柏林工业大学客座首席科学家。主要研究方向为高性能计算、网络并行计算和优化编译等。