LogP模型上的最优播送与求和算法的实现*

寿 标 陈国良

(中国科学技术大学计算机系 合肥 230027)

摘要 与以往的各种并行计算模型相比,LogP模型更真实、更全面地反映了大规模并行计算机MPC(massively parallel computers)的特征.鉴于目前见到的LogP模型上的算法都仅局限于给出设计思想,本文尝试用算法语言来描述LogP模型上的完整的可移植算法.文中针对单项播送与求和这2类基本问题,实现了它们在有任意参数的LogP模型上的最优算法,并对其时间复杂度进行了分析.

关键词 大规模并行计算机,并行计算模型,并行算法,最优播送算法,最优求和算法.

合适的并行计算模型的选取是并行算法设计者面临的首要问题.由于迄今为止的许多并行计算模型对并行机特征的抽象都不够理想,使得在这些模型上设计出的并行算法既不能有效地反映程序的实际执行情况,又难以移植.其主要原因之一在于没有处理好并行机所特有的通信问题.如常用的PRAM模型默认处理机间的通信不需任何代价;BSP模型[1]虽较好地考虑了通信延迟,但在实现h-关系(h-relation)及用一个BSP处理机模拟若干个PRAM处理机时难以保证算法效率.

大规模并行计算机MPC(massively parallel computers)是并行机发展的一种主要趋势,LogP模型[2]就是基于MPC提出的一种新的并行计算模型.该模型认为并行机由许多个本身带有局部存储器的功能强大的处理机组成,处理机之间采用异步的点到点消息传递来进行通信.CM-5,Paragon等机器[3]就是LogP模型的一种实例.LogP模型用L,o,g和P这4个参数来刻画MPC,它对并行机互连网络采用何种拓扑结构没有要求.LogP模型的几个参数的具体含义如下:

L:latency,网络延迟,即从消息进入互连网络开始至到达目的处理机为止所需的最大时间.LogP模型假设任意处理机对之间的网络延迟均为L.在L期间,处理机可以进行本身的计算,从而有可能使计算与通信重叠.

o:overhead,软件开销,即处理机在接收或发送每一个消息时对消息进行处理所需的时间.在这段时间内,处理机不能进行本身的计算.

g:gap,收发间隔,即处理机在连续发送或连续接收消息时所需的最小时间间隔.$1/g$相

* 本文研究得到国家863高科技项目基金资助.作者寿标,1971年生,研究生,主要研究领域为并行算法和并行体系结构.陈国良,1938年生,教授,博士导师,主要研究领域为并行算法和智能计算.

本文通讯联系人:寿标,合肥230027,中国科学技术大学计算机系

本文1995-12-22收到修改稿

当于每个处理机可用的通信带宽.显然,g 至少应等于 o.

P:处理机/存储器模块数.

以下假定各处理机上一次局部运算只需单位时间,并称该单位时间为时间周期,并且 L,o 和 g 均以时间周期为单位进行度量.由于传递一次消息所用时间包括发送时对消息进行处理的时间(o)、消息在互连网上所用时间(L)以及接收时对消息进行处理的时间(o),因此,LogP 模型上的一次消息传递需 $L+2o$ 个时间周期.并行算法设计主要的困难有:如何合理地安排各处理机上的工作负载;如何使通信与计算重叠来提高算法的效率;以及如何用算法语言对算法进行形式化描述.反映到 LogP 模型上,算法设计的关键有 2 点:①要尽量减少处理机间的通信次数;②要充分利用网络延迟 L 这段时间进行计算以提高处理机的利用率.播送与求和操作是许多并行机本身所提供的一种基本功能,其效率直接影响整机性能.文献[4]虽然讨论了 LogP 模型上的最优播送与求和,但仅适合于特殊参数情况($o=0$,$g=1$),同时也没有给出其算法语言描述.本文主要用算法语言实现了 LogP 模型上的完整的最优单项播送算法与求和算法,并给出了它们的时间复杂度.算法对 LogP 模型的 4 个参数的具体取值没有任何限制.下面我们首先给出用于确定 LogP 模型上的最优播送树形状的一个递推关系和其它一些定义,然后分别设计并分析 LogP 上的最优单项播送算法与最优求和算法.

1　有关定义

播送操作是并行机一般都支持的一种基本功能.所谓单项播送问题就是要求在最短时间内将单个消息从任意给定的处理机发送到所有其它处理机中去.下面简称单项播送问题为播送问题.播送算法的设计思想很简单,即应让所有收到消息的处理机都尽可能早和频繁地送消息给还未收到消息的处理机.由于每个处理机只接收一次消息,播送路径是树状的,我们称其为播送树.下面给出用以确定 LogP 模型上最优播送树形状所需的一些定义和定理.文中不加区分地使用结点和处理机.

定义 1. 设处理机 i 的接收延迟为处理机首次收到播送消息所需的时间,播送算法 A 的运行时间 T_A 为 $\max\limits_{0\leqslant i\leqslant P-1}\{D_i(A)\}$,播送问题的复杂度 $C(P;L,o,g)$ 为 $\min\limits_{A}\{T_A\}$.若 $T_A=C(P;L,o,g)$,则称 A 是最优播送算法.

定义 2. 设通用最优播送树 $GBT(P;L,o,g)$ 为一带标号的有序无限树,其根结点的标号为 0,标号为 T 的结点其孩子的标号分别为 $T+L+2o+ig$,$i\geqslant 0$.

定理 1.[4] 设 $B(P;L,o,g)$ 为 $GBT(P;L,o,g)$ 的包含所有 P 个最小标号结点(即 $B(P;L,o,g)$ 中的其它结点的标号均大于 $GBT(P;L,o,g)$ 中结点的标号)的子树,则 $B(P;L,o,g)$ 必是 P 个处理机的最优播送树 $OBT(P;L,o,g)$.

依据 $OBT(P;L,o,g)$,可以计算出在 $T\geqslant 0$ 个时间周期内播送算法所能播送到的处理机个数 $P(T;L,o,g)$.定理 2 给出了具体的计算公式.注意在 LogP 模型上传递 1 次消息所需时间为 $L+2o$,处理机只能以间隔 g 的速度向外发送消息.

定理 2. 对任意的 $T\geqslant 0$,$L>0$,$o>0$ 和 $g>0$,均有 $P(T;L,o,g)=f_T$,其中函数 f 的一般形式 f_n 满足下面的递推关系:

$$f_n=\begin{cases}1, & 0\leqslant n<L+2o,\\ 1+\lfloor n/(L+2o)\rfloor, & L+2o<g \text{ 且 } L+2o\leqslant n<g,\\ f_{n-g}+f_{n-2o-L}, & n\geqslant\max\{g,L+2o\}.\end{cases}$$

证明：① 当$0\leqslant n<L+2o$时，根处理机没有时间发送消息，$f_n=1$.

② 当$L+2o<g$且$L+2o\leqslant n<g$时，收到消息的处理机只能发送1次消息，没有时间发送第2次，此时播送树退化为线性阵列，故$f_n=1+\lfloor n/(L+2o)\rfloor$.

③ 当$n\geqslant\max\{g,L+2o\}$时，令$k=\lfloor(n-L-2o)/g\rfloor$，则$k+1$为$OBT$树根结点的孩子个数. 因$OBT$树根结点的各子树仍为$OBT$树，有$f_n=1+\sum_{m=0}^{k}f_{n-L-2o-mg}$. 解递推关系可得$f_n-f_{n-g}=f_{n-L-2o}+f_{n-g-L-2o-kg}$，又$n-g-L-2o-kg<0$，有$f_{n-g-L-2o-kg}=0$，从而得到$f_n=f_{n-g}+f_{n-L-2o}$. □

由定理2可知在$OBT(P;L,o,g)$上播送消息所需时间T满足关系式$P=f_T$. 下面我们把播送树里各结点可用以播送消息的时间称为该结点的有效时间. 播送树的各结点可以用二元组$(P_i,T_i)(0\leqslant i<P)$来唯一地代表，其中$P_i$和$T_i$分别为根据下面的定义3和定理3计算得到的结点处理机编号和结点有效时间.

定义 3. 对树$OBT(P;L,o,g)$，其各结点的处理机编号用下面的规则来确定：设P_i为任一内结点的处理机编号，以该结点的孩子为根的各子树的结点数分别为$C_0,C_1,\ldots$，则P_i的第$k(k\geqslant 0)$个孩子结点的处理机编号用下述方法确定：当$k=0$时，编号为$(P_i+1)\ mod\ P$；当$k\geqslant 1$时，编号为$(P_i+1+\sum_{m=0}^{k-1}C_m)mod\ P$.

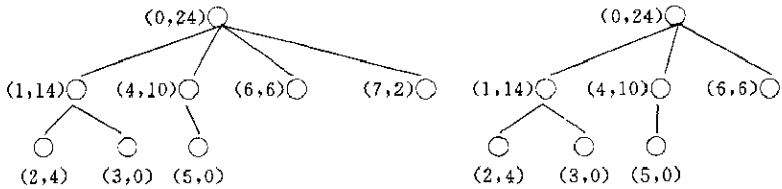
定理 3. 对任意的$OBT(P;L,o,g)$，可以用二元组$(P_i,T_i)(0\leqslant i<P)$来唯一地表示它的各结点. 其规则如下：令根结点的处理机编号为P_0，根结点的有效时间T_0为在该播送树上播送消息给所有处理机所需的时间；设某内结点的处理机编号为P_i，有效时间为T_i，则它的第$k(k=0,1,\ldots)$个孩子结点的处理机编号为$(P_i+1+f_{T_i}-f_{T_i-kg})\ mod\ P$(函数$f$如定理2所定义)，有效时间为$T_i-L-2o-kg$.

证明：当$k=0$时，显然成立. 若$k\geqslant 1$，由定义3，第k个孩子结点的处理机编号为$P_i+1+\sum_{m=0}^{k-1}f_{T_i-L-2o-mg}=P_i+1+f_{T_i}-f_{T_i-kg}$. □

由$OBT(P;L,o,g)$的定义可知，其P值范围限于f_n序列. 为此需构造含有任意P个处理机的最优播送树$T^*(P;L,o,g)$.

定义 4. 对满足$f_{n-1}<P\leqslant f_n$的任意处理机个数P，当$P=f_n$时，令$T^*(P;L,o,g)=OBT(P;L,o,g)$；当$f_{n-1}<P<f_n$时，令$T^*(P;L,o,g)$为由$OBT(f_n;L,o,g)$的前序遍历(深度优先)序列的前$P$个结点构成的子树.

这里采用前序遍历是为了便于下面的算法设计. 至此，对任意的$P>0$，根据定理3和定义4已可求出其$T^*(P;L,o,g)$，它的各结点的二元组表示都是唯一的. 图1(a)和(b)分别为$P=8$和$P=7$时的最优播送树$T^*(8;6,2,4)$和$T^*(7;6,2,4)$，其中根结点设为0号处理机. 注意$T^*(7;6,2,4)$砍去了$T^*(8;6,2,4)$里的结点(7,2).



(a) $T^*(8;6,2,4)$　　(b) $T^*(7;6,2,4)$

图1

2　最优单项播送算法

有了以上的一些有关最优播送树的定义和性质,可以设计出下面的最优播送算法1.这里采用SPMD[5](single program multiple data)编程方式,处理机间的通信采用消息传递的方法来进行.

算法1. 从处理机P_0播送消息至其余各处理机

输入:msg,L,o,g,P,P_0及递推关系$f_n=f_{n-g}+f_{n-L-2o}$.

输出:各处理机均收到消息msg

```
Broadcast(L,o,g,P,P0,msg,fn)
(1) Pi=GetMyProcId();                                  /* 获取进程所在处理机的处理机编号 */
(2) if(Pi=P0) {                                        /* 判断本处理机是否为根处理机P0 */
    (2.1) Find T which satisfy fT-1<P≤fT;              /* 计算P0的有效时间T */
    (2.2) for (k=0;k<⌊(T-L-2o)/g⌋+1;k++) {             /* 向所有孩子发送消息 */
          if (P-1-fT+fT-kg>0)                          /* 若第k个孩子存在,则向它发送消息 */
            send(msg,T-L-2o-kg,(P0+1+fT-fT-kg) mod P, min{P-1-fT+fT-kg,fT-L-2o-kg});
          }
    }
(3) else {                                             /* 非根处理机上的处理 */
    (3.1) recv(msg,Ti,src_proc_id,Ni);                 /* 从父处理机接收消息 */
    (3.2) for (k=0;k<⌊(Ti-L-2o)/g⌋+1; k++) {           /* 向所有孩子发送消息 */
        if (Ni-1-fTi+fTi-kg>0)                         /* 若第k个孩子存在,则向它发送消息 */
          send(msg,Ti-L-2o-kg,(Pi+1+fTi-fTi-kg) mod P, min{Ni-1-fTi+fTi-kg,fTi-L-2o-kg});
        }
      }
    }
```

算法中发送消息所用send函数的定义为:send(消息、目的结点有效时间、目的结点编号、以目的节点为根的子树的结点总数);接收消息所用recv函数的定义为:recv(消息、本结点有效时间、源结点编号、以本结点为根的子树的结点总数).

定理4. 算法1花费的时间T_B满足下述关系式:$\min\{g,L+2o\}\log(P)\leqslant T_B<\max\{g,L+2o\}\log(4P)$.

证明:分2种情况.

①当$g\geqslant L+2o$时,由递推关系可得$f_T=f_{T-g}+f_{T-L-2o}\geqslant 2f_{T-g}=2^{\lfloor T/g\rfloor}>2^{T/g-1}$,两边取对数有$T<g\log(2f_T)$.又$f_T=f_{T-g}+f_{T-L-2o}\leqslant 2f_{T-(L+2o)}\leqslant 2^{\lfloor T/(L+2o)\rfloor}\leqslant 2^{T/(L+2o)}$,两边取对数有$T\geqslant(L+2o)\log(f_T)$.综上得$(L+2o)\log(f_T)\leqslant T<g\log(2f_T)$,即$\min\{g,L+2o\}\log(f_T)\leqslant T<\max\{g,L+2o\}\log(2f_T)$.

②当$g<L+2o$时,由递推关系可得$f_T=f_{T-g}+f_{T-L-2o}>2f_{T-(L+2o)}=2^{\lfloor T/(L+2o)\rfloor}>2^{T/(L+2o)-1}$,两边取对数有$T<(L+2o)\log(2f_T)$.又$f_T=f_{T-g}+f_{T-L-2o}<2f_{T-g}\leqslant 2^{\lfloor T/g\rfloor}$

$\leqslant 2^{T/g}$，两边取对数有 $T \geqslant g\log(f_T)$. 综上得 $g\log(f_T) \leqslant T < (L+2o)\log(2f_T)$，即 $\min\{g, L+2o\}\log(f_T) \leqslant T < \max\{g, L+2o\}\log(2f_T)$.

由①、②可以得到 $\min\{g, L+2o\}\log(f_T) \leqslant T < \max\{g, L+2o\}\log(f_T)$. 又 $P \leqslant f_T \leqslant 2f_{T-\min\{g,L+2o\}} \leqslant 2f_{T-1} < 2P$，代入上式有 $\min\{g, L+2o\}\log(P) \leqslant T = T_B < \max\{g, L+2o\}\log(4P)$. 命题得证. □

3 最优求和算法

Log$P$ 上的求和也就是将操作数分配到各个处理机上，在最短时间内求出所有操作数之和. 为此，必须细致地给各处理机分配最佳的操作数个数. 由于并行机上的求和过程也就是将各操作数从所有处理机汇总到某一特定处理机的过程，因此，它是播送的逆过程，其通信依赖图也是树形的. 需要进一步确定的是 Log$P$ 模型上最优求和树 $S^*(P;L,o,g)$ 的形状以及相应的任务分配方案.

为尽快求出所有操作数之和，各结点除准备接收或发送消息时不能进行计算外，其余时间应尽可能参与计算，使得计算与通信重叠（L 期间可进行计算）. 因此，非叶结点上局部求和与接收消息是穿插进行的. 由于求和树中非叶结点在求局部操作数之和的同时，还需从其它结点接收发送过来的局部和并累加到自身的局部和上，也就是说处理机每接收 1 次消息都需用 1 个额外的时间周期（本来可用以局部计算，假设 1 次求和需 1 个时间周期）来处理它，其效果相当于消息在网络上花费了 $L+1$ 个时间周期，故最优求和树 $S^*(P;L,o,g)$ 也就是最优播送树 $T^*(P;L+1,o,g)$（注意：以下在计算 f_t 时，网络延迟应采用 $L+1$ 而不是 L）.

最优求和树 $S^*(P;L,o,g)$ 各结点的二元组表示 (P_i,T_i) 可根据 $T^*(P;L+1,o,g)$ 各结点的二元组表示来得出. 在设计求和算法时，我们从另一角度考虑问题，即讨论如何在给定的时间周期内求出尽可能多的操作数的总和. 下面的引理 1 用于确定最优求和树 S^* 的各结点上的操作数个数.

引理 1. 对任意的最优求和树 $S^*(P;L,o,g)$，设各结点用二元组 (P_i,T_i) 表示. 为在根结点的有效时间里求出尽可能多的操作数之和，叶结点上的操作数个数 N_i 应为 T_i+1，非叶结点上的操作数个数 N_i 应为 $T_i-(\lfloor (T_i-(L+2o+1))/g \rfloor+1)(o+1)+1$.

证明：叶结点时显然成立. 对非叶结点，其孩子结点的总数为 $\lfloor (T_i-(L+2o+1))/g \rfloor+1$. 又处理机每次接到消息后都需进行一次局部求和，即每次接收都需占用处理机 $o+1$ 的求和时间，故非叶结点可用以求和的时间为 $T_i-(\lfloor (T_i-(L+2o+1))/g \rfloor+1)(o+1)$，再加 1 即为其上的操作数个数. □

定理 5. 对任意处理机个数 P，令 $f_{T-1} < P \leqslant f_T$，则在 T 个时间周期内基于最优求和树 $S^*(P;L,o,g)$ 的求和算法最多能处理的操作数个数为：$N_S = \sum_{i=0}^{P-1} T_i - oP + o + 1$.

证明：对处理机 (P_i,T_i)，设其孩子个数为 K_i，由引理 1，知其 N_i 为 $T_i-K_i(o+1)+1$. 又 $\sum_{i=0}^{P-1} K_i = P-1$，故 $N_S = \sum_{i=0}^{P-1} N_i = \sum_{i=0}^{P-1} T_i - (o+1)\sum_{i=0}^{P-1} K_i + P = \sum_{i=0}^{P-1} T_i - oP + o + 1$. □

算法 2. 计算 N 个操作数之和，结果由处理机 P_0 输出

输入:$L,o,g,P,P_0,N,a[0..N-1]$及递推关系 $f_n=f_{n-g}+f_{n-(L+1)-2o}$.

输出:$\sum_{j=0}^{N-1}a[j]$

```
Summation(L, o, g, P, P0, N, a[0..N-1], fn)
(1) Find T which satisfy f_{T-1}<P≤f_T;                    /* 计算 P0 的有效时间 T */
(2) Pi=GetMyProcId();                                      /* 获取进程所在处理机的处理机编号 */
(3) Compute local Ti according to T*(Pi;L+1,o,g);          /* 计算本处理机的有效时间 */
(4) Ki=⌊(Ti-(L+2o+1)/g)⌋+1;                                /* Ki 为本处理机的孩子个数 */
(5) Ai=Ti-(o+1)Ki+1;                                       /* Ai 为本处理机特有的工作量 */
(6) Bi=⌊(n-Σ_{j=0}^{P-1} Aj)/P⌋;                           /* Bi 为各处理机平均分配的工作量 */
(7) Ni=Ai+Bi;                                              /* Ni 为本处理机上总的工作量 */
(8) Copy Ni operands from input array a to local array b;  /* 将 Ni 个操作数存放于数组 b */
(9) for (loc_sum=0,j=0;j<Ni;j++) {                         /* 循环处理 Ni 个局部操作数 */
    (9.1) loc_sum +=b[j];                                  /* 计算一次局部和 */
    (9.2) if (message arrived) {                           /* 判断消息是否到达 */
        Get message chd_sum from buffer;                   /* 接收消息 */
        loc_sum+=chd_sum;                                  /* 将消息加到局部和上 */
        }
    }
(10) if (Pi! =P0!) {                                       /* 判断是否根处理机 */
        parent=ComputeParent();                            /* 计算父结点的处理机号 */
        Send message loc_sum to processor parent;          /* 发送局部和给父处理机 */
    }
    else {Print loc_sum;}                                  /* 根处理机输出结果 */
```

计算结点(P_i-T_i)的父结点算法为:

```
Function ComputeParent(Pi,Ti,T)
(1) Pi=(Pi-P0) mod P;                                      /* 结点编号归一化 */
(2) parent=0;                                              /* 赋初值 */
(3) while (1) {                                            /* 循环直至找到父结点 */
    (3.1) for (k=0; k<⌊(T-(L+1+2o))/g⌋+1; k++) {           /* 循环处理各孩子结点 */
          temp=parent+f_T-f_{T-kg}+1;
          if (temp==Pi)
            return (parent+P0) mod P;                      /* 找到父结点则返回 */
          if (temp<Pi && Pi<temp+f_{T-(L+1+2o)-kg})
            goto nextlevel;                                /* 到求和树的下一级寻找 */
          }
    (3.2) nextlevel;
    (3.3) parent=temp;                                     /* 搜索求和树的下一层 */
    (3.4) T=T-(L+1)-2o-kg;
}
```

由定理 5,很容易得到算法 2 的时间复杂度如下:

定理 6. 算法 2 的时间复杂度为 $T(N)=T_B+(N-\sum_{i=0}^{P-1}A_i)/P=T_B+(N-N_S)/P$.

图 2 为一个采用最优求和算法在 7 个处理机上对 82 个操作数进行求和的例子.其中左图圆圈内的数字为各结点特有的操作数个数 A_i,右图圆圈内的数字为平均分配给各结点的操作数个数 B_i.

4　结束语

在 LogP 模型上如何设计通信与计算效率都最好的并行算法是本文讨论的重点.LogP 作为一种新提出的模型,虽有文献[4,6]对其上的算法设计进行了讨论,但都没有给出完整

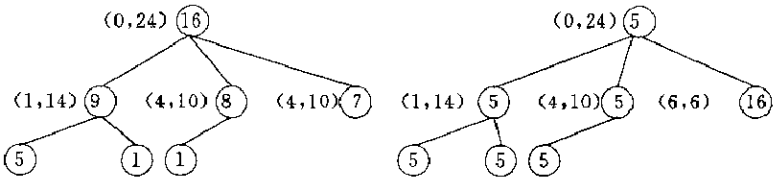


图2 $S^*(7;5,2,4)$各结点上的操作数个数分配

的算法.本文主要考虑播送与求和这2类问题,给出了Log$P$模型上的最优单项播送算法与最优求和算法,并分析了算法的时间复杂度.目前在Log$P$模型上实现的算法还很有限,今后将在Log$P$模型上对更多的算法进行设计与分析.

致谢 作者非常感谢郑世荣教授对本文写作的鼓励与支持.

参考文献

1 Valiant L G. A bridging model for parallel computation. Communications of the ACM, 1990,**33**(**8**):103～111.

2 Culler D E, Karp R M, Patterson D *et al*. Log$P$: towards a realistic model of parallel computation. In: Proc. of 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 1993. 1～12.

3 Bonniger T, Esser R, Krekel D. CM-5E, KSR2, Paragon XP/S: a comparative description of massively parallel computers. Parallel Computing, 1995,**21**:199～232.

4 Karp R M, Sahay A, Santos E *et al*. Optimal broadcast and summation in the Log$P$ model. In: Proc. of 5th *Symposium* on Parallel Algorithms and Architectures, June 1993.

5 Eicken T, Culler D E, Goldstein S C *et al*. Active message: a mechanism for integrated communication and computation. In: Proc. of 19th International Symposium on Computer Architecture, May 1992.

6 Sahay A. Hiding communication costs in bandwidth limited parallel FFT computation. TR UCB/CSD 92/722, UC Berkeley, 1992.

THE IMPLEMENTATION OF OPTIMAL BROADCAST AND SUMMATION ALGORITHMS IN THE Log*P* MODEL

SHOU Biao CHEN Guoliang

(*Department of Computer Science University of Science and Technology of China Hefei* 230027)

Abstract Compared with other parallel computation models, the Log$P$ model reflects the characteristics of MPC(massively parallel computers) more reasonably. In this paper the authors try to use the Log$P$ model to write portable algorithms for MPC. They first propose a general recurrence formula for the defining of optimal broadcast tree, then design and analysis the complete optimal single item broadcast algorithm and optimal summation algorithm in the Log$P$ model whose parameters may have any given values.

Key words massively parallel computers, parallel computation model, parallel algorithm, optimal broadcast algorithm, optimal summation algorithm.