第39卷 增刊(2) 中山大学学报(自然科学版) Vol.39 Suppl.(2)
2000年 12月 ACTA SCIENTIARUM NATURALIUM UNIVERSITATIS SUNYATSENI Dec. 2000

文章编号：0529-6579(2000)S2-0176-05

一种统一的双调排序标志方法*

张永民[1]，张永东[2]

(1. 中山大学计算机科学系，广州 510275；2. 中山大学科学计算与计算机应用系)

摘 要：提出一种双调排序标志方法和一般并行机上的实现算法. 该方法可简化双调排序算法在许多不同并行模型上的实现.

关键词：双调排序；并行算法；处理器阵列；标志方法；超立方

中图分类号：TP183 **文献标识码**：A

排序算法一直是计算机科学的重要研究领域[1-5]. 双调排序算法是由 Batcher[3]于1968年提出的. 该算法在通讯网络中也具有重要的应用价值[4]. 双调排序法出现之后对它的研究主要集中在将它移植到不同的并行模型上[5]. 但是，由于这些双调排序算法在不同的并行模型上实现时标志方法很不统一，在一定程度上复杂化了这一移植过程. 本文提出了一种标挚方法，它较大地简化了双调排序算法在并行模型上的移植过程，并使并行模型上的双调排序算法的实现更加简明.

本文第1节为双调排序的基本思想；第2节提出了本文的标志方法，并证明了它的正确性；第3节简略讨论了算法设计思想.

1 双调排序算法

排序算法可以把 n 个无序数排列成一列有序数 $A_0, A_1, \cdots, A_{n-1}$，使得 $A_i \leqslant A_{i+1}$(升序)或 $A_{i+1} \leqslant A_i$(降序)，$0 \leqslant i < n-1$. 双调排序法是基于 Batcher 定理实现的一种排序算法.

定义 1 n 个数的序列 $A_0, A_1, \cdots, A_{n-1}$ 称为双调序列，如果这个序列满足：

(1) 存在 j，$0 \leqslant j < n$，使得 $A_0 \leqslant A_1 \leqslant \cdots \leqslant A_j, A_{j+1} \geqslant A_{j+1} \geqslant \cdots \geqslant A_n$.

(2) 通过循环移位后，条件(1)满足.

例如，1,3,6,7,8,5,4,2 为双调序列，4,2,1,3,6,7,8,5 亦为双调序列.

Batcher 定理[3] 对于任何给定的 $n = 2^m$ 个元素的双调序列 $A_0, A_1, A\cdots, A_{n-1}$，如果 $B_i = \min(A_i, A_{\frac{n}{2}+i}), C_i = \max(A_i, A_{\frac{n}{2}+i}), 0 \leqslant i < \frac{n}{2}$. 那么

(1) $B_0, B_1, \cdots, B_{\frac{n}{2}-1}$ 和 $C_0, C_1, \cdots, C_{\frac{n}{2}-1}$ 均为双调序列.

* 基金项目：国家自然科学基金资助项目(19871094)；广东省自然科学基金资助项目(990229)；中山大学高等学术研究中心基金资助项目(00M8)

收稿日期：2000-03-02； 作者简介：张永民(1964～)，男，讲师.

(2) $B_i \leqslant C_i, 0 \leqslant i,j, < \frac{n}{2}$.

反复使用 Batcher 定理，可以把 $n = 2^m$ 个数的双调序列归并为有序序列，这一过程称为双调归并.8 个数的双调归并如图 1 所示.横线表示数的存放处，箭号表示两数进行比较，箭头处存较小的数，箭尾处存放较大的数.比较从左往右进行.

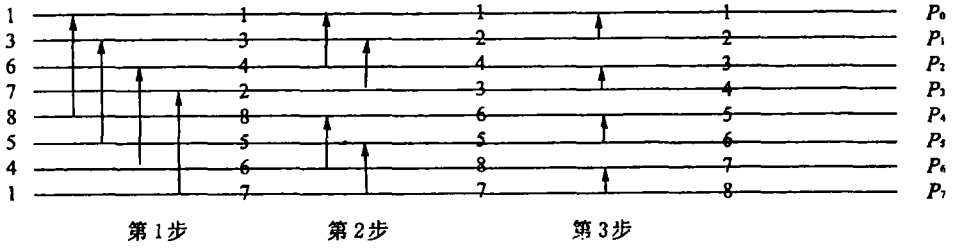

图 1　$n = 2^3$ 个数的双调归并

我们可以得到 2 个数的有序序列，这样组合成 4 个数的双调序列，归并这些双调序列，形成 8 个数的双调序列，…，归并 2^m 个数的双调序列，这样实现的排序称为双调排序，8 个数的双调排序如图 2 所示.

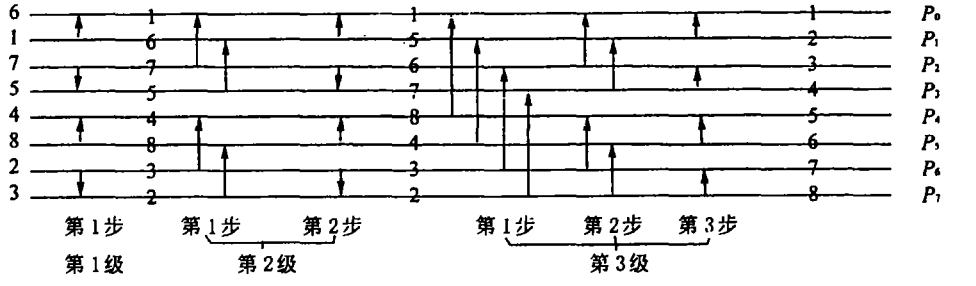

图 2　$n = 2^m$ 个数的双调排序

对于 $n = 2^m$ 个数的双调归并需要 m 步并行比较.对于 2^m 个数的双调排序需要 m 级的双调归并，第 $i(1 \leqslant i \leqslant m)$ 级双调归并需要 i 步并行比较，因此，$n = 2^m$ 个数双调排序共需要 $1 + 2 + \cdots + m = m(m+1)/2$ 步并行比较.

2　一种统一的标志方法

如果把图 1,2 中的每一条水平线上的数用一个相应的处理器来存放，每一个箭号表示箭头和箭尾处的处理器所存数之间的比较.易看出，只要任何箭号处的 2 个处理器可以通信，每一步中的比较可以同时进行，双调排序法很容易用多处理器结构来实现.

在 $n = 2^m$ 个数的双调排序算法中，每一步都有 $n/2$ 对处理器参与比较.如果给这些处理器用 0 到 $2^m - 1$ 进行编，并用 P_i 表示第 i 号处理器$(0 \leqslant i < 2^m)$，则这些比较是否有规律可循?与所有级、步处理器号之间有没有关系?Stone 进行观察之后总结出下面的规律.

引理 1[5]　$n = 2^m$ 个数分别放于 n 个处理器 $P_0, P_1, \cdots, P_{n-1}$ 之中.并行双调排序的第 j 第 k 步$(1 \leqslant j \leqslant m, 1 \leqslant k \leqslant j)$ 的比较方法为处理器号的二进制表示 $i_{m-1}i_{m-2}\cdots i_0$ 的第 $j-k$ 位不同的处理器对之间进行比较.

这一比较方法可用图3表示. b_i 表示在所有标号的二进制表示第 i 位不同的处理器对之间进行比较.

$n = 2^m$ 个数的双调排序结果处理器 P_i 存放了一个数 $A_i(0 \leqslant i < 2^m)$，并且可以是升序和降序2种情况. 若为降序，则 $A_i \leqslant A_j, 0 \leqslant j < i < 2^m$；若为升序，则 $A_i \leqslant A_j, 0 \leqslant i < j < 2^m$. 前面只考虑了升序.

双调排序不仅要确定处理器之间的比较方法，而且要确定比较的方向，即2个处理器内的数进行比较之后较小的数和较大的数如何分别存在这2个处理器之中.

	第1步	第2步	第3步	…	第 m 步
第1级	b_0				
第2级	b_1	b_0			
第3级	b_2	b_2	b_0		
⋮					
第 m 级	b_{m-1}	b_{m-2}	b_{m-3}	…	b_0

图3 $n = 2^m$ 个数的双调排序的比较方法

图2和图3中较小数放在箭头处的处理器中，较大数放于箭尾处的处理器之中. 双调排序算法要在每步比较之前先确定每个处理器是存放在参与比较的2数之中的较小数还是较大数. 若存放较小数该处理器比较之前应置标志为0，否则，置标志为1. 这称为双调排序的标志方法. n 个数的双调排序的标志方法并非唯一. 这是因为每一级比较（最高级比较除外）只要形成许多双调序列便可以了，并不具体规定每一比较的方向. 下面的定理给出了一种双调排序的标志方法.

引理2 $n = 2^m$ 个数分别放于 2^m 个处理器之中，如果这 $n = 2^m$ 个数为双调序列，并且按比较模式 $b_{m-1}, b_{m-2}, \cdots, b_0$ 进行 m 步并行比较. 处理器 $P_i(0 \leqslant i < 2^m, i$ 的二进制表示为 $i_{m-1}i_{m-2}\cdots i_0$ 的第 k 步的标志为 i_{m-k}，则 m 步比较之后的结果为升序；若上述标志为 $\bar{i}_{m-k}$，则 m 步比较之后的结果为降序.

证明 当 $m = 1$ 时，引理显然成立. 假设当 $m = t$ 时的引理成立. 即 2^t 个数的第 t 步并行模式为 $b_{t-1}, b_{t-2}, \cdots b_0, P_i$ 的第 t 步比较标志分别为 $i_{t-1}, i_{t-2}, \cdots, i_0$（$i$ 的二进制表示 $i_{t-1}i_{t-2}\cdots i_0$）时，结果为升序. P_i 的比较标志分别为 $\bar{i}_{t-1}\bar{i}_{t-2}\cdots\bar{i}_0$ 时，结果排成降序.

当 $m = t + 1$ 时，2^{t+1} 个数的双调序列首先进行一次并行比较 b_t, P_i 的并行比较标志为 i_t（i 的二进制表示 $i_ti_{t-1}\cdots i_0$），即 P_i 和 $P_{2^t+i}(0 \leqslant i < 2^t)$ 进行比较，且较小数放在 P_i 中，较大数放在 P_{2^t+i} 中. 依 Batcher 定理，可得到2个双调序列，分别存于 $P_0, P_1, \cdots, P_{2^t-1}$ 和 $P_{2^t}, P_{2^t+1}, \cdots, P_{2^{t+1}-1}$ 之中，且 $P_0, \cdots, P_{2^t+1}$ 中的数小于 $P_{2^t}, \cdots, P_{2^{t+1}-1}$ 中的数. 因为此后的 t 步比较模式为 $b_{t-1}, b_{t-2}, \cdots, b_0, P_i$ 的比较标志为 $i_{t-1}, i_{t-2}, \cdots, i_0(0 \leqslant i < 2^t, i$ 的二进制表示为 $i_ti_{t-1}\cdots i_0$），依归纳假设，$P_0, P_1, \cdots, P_{2^t-1}$ 可排升序. 又因为 $P_{2^t}, P_{2^t+1}, \cdots, P_{2^{t+1}-1}$ 的标号的二进制表示除最高位均为1之外，其余与 $P_0, P_1, \cdots, P_{2^t-1}$ 相同，而最高位在后面的 t 步比较模式和标志中并未出现，故亦可排成升序. 这样就实现了 2^{t+1} 个数的排序，且排为升序. 同样可证，当 P_i 的并行比较标志分别为 $\bar{i}_t, \bar{i}_{t-1}, \cdots, \bar{i}_0$ 时（$0 \leqslant i < 2^{t+1}, i$ 的二进制表示 $i_ti_{t-1}\cdots i_0$），可排为降序.

定理1 如果① 有 $n = 2^m$ 个处理器 $P_0, P_1, \cdots, P_{2^m-1}$. 初始时每个处理器存放1个数；② 处理器之间的比较方法按引理2进行；③ $\mathrm{Mark}(i, j, k) = i_j \oplus i_{j-k}, 0 \leqslant i < 2^m, 1 \leqslant j \leqslant m, 1 \leqslant k \leqslant j, i$ 的二进制表示为 $i_{m-1}i_{m-2}\cdots i_0$. $\mathrm{Mark}(i, j, k)$ 为处理器 P_i 在第 j 级第 k 步的标志.

那么，当 $i_m = 0$ 时，经过 m 级比较之后，$n = 2^m$ 个数在 n 个处理器中排成升序；当 $i_m = 1$ 时，排成降序.

证明　当 $m = 1$ 时，定理显然成立. 假设当 $m = t$ 时定理成立. 即如果 $n = 2^t$ 个处理器开始时分别存放了 1 个数，比较模式遵守引理 1，即为 $b_0, b_1, b_{t-1}, b_{t-2}, \cdots, b_0$. $\mathrm{Mark}(i, j, k) = i_j \oplus i_{j-k}, 0 \leqslant i < 2^t, 1 \leqslant j \leqslant t, 1 \leqslant k \leqslant j$. i 的二进制表示为 $i_{t-1}, i_{t-2}, \cdots, i_0 A! \# i_t = 0$ 时，这 2^t 个数在 n 个处理器中排成升序，$i_t = 1$ 时，这 2^t 个数在 n 个处理器中排成降序.

当 $m = t + 1$ 时，2^{t+1} 个处理器 $P_0, P_1, \cdots, P_{2^{t+1}-1}$ 可分成 2 部分看待. $P_0, P_1, \cdots, P_{2^t-1}$ 的下标二进制表示第 t 位均为 0，$P_{2^t}, P_{2^t+1}, \cdots, P_{2^{t+1}-1}$ 的下标的二进制表示第 t 位均为 1.

依归纳假设，在前 t 级比较结束之后，$P_0, P_1, \cdots, P_{2^t-1}$ 中的数应排成升序，$P_{2^t}, P_{2^t+1}, \cdots, P_{2^{t+1}-1}$ 中的数应排成降序. 即 $P_0, P_1, \cdots, P_{2^{t+1}-1}$ 中的数在 t 级并行比较之后构成一个 2^{t+1} 个数的双调排序.

在 $t + 1$ 级比较时，$\mathrm{Mark}(i, t + 1, k) = i_{t+1} \oplus i_{t-k+1}, 1 \leqslant k \leqslant t + 1, 0 \leqslant i < 2^{t+1}$，$i$ 的二进制表示为 $i_t i_{t-1} \cdots i_0$. 当 $i_{t+1} = 0$ 时，$\mathrm{Mark}(i, t + 1, k) = i_{t+1-k}$，即 P_i 在 $t + 1$ 级的 $t + 1$ 步并行比较的标志分别为 $i_t, i_{t-1}, \cdots, i_0$，又因为 $t + 1$ 级的比较模式为 $b_t, b_{t-1}, \cdots, b_0$，根据引理 1，第 $t + 1$ 级比较之后 2^{t+1} 个数的双调序列可排成升序. 同样，当 $i_{t+1} = 1$ 时，$\mathrm{Mark}(i, t + 1, k) = \bar{i}_{t+1-k}$，根据引理 1，第 $t + 1$ 级比较之后可排成降序. 证毕.

3　统一的双调排序算法

利用引理 1 和定理 1，本节给出了统一的双调排序算法，它可以方便地移植到 SIMD 和 MIMD 并行模型上.

实现该算法使用了 Receive 语句和 Send 语句. 它们均有 3 种形式：

(1) P_i Receive $A[i]$from $B[i']$，处理器 P_i 接收处理器 $P_{i'}$ 的寄存器 $B[i']$ 的内容且存放于寄存器 $A[i]$ 中.

(2) P_i Receive $A[i]$from $P_{i'}$，处理器 P_i 从处理器 $P_{i'}$ 的寄存器接收一个数据且存放于寄存器 $A[i]$ 中.

(3) P_i Receive $A[i]$from X-edge，处理器 P_i 从其输入边 X-edge 收一个数据且存放于寄存器 $A[i]$ 中.

(4) P_i Send $A[i]$to $B[i']$，处理器 P_i 将其寄存器 $A[i]$ 的内容送至处理器 $P_{i'}$ 的寄存器 $B[i']$ 中.

(5) P_i Send $A[i]$to X-edge，处理器 P_i 将其寄存器 $A[i]$ 的内容送至其输入边 X-edge.

(6) P_i Send $A[i]$to $P_{i'}$，处理器 P_i 将其寄存器 $A[i]$ 的内容送至处理器 $P_{i'}$ 之中.

在 SIMD 模型中，Send 语句和 Receive 语句由控制单元统一形成一系列传送操作来实现. 除被屏蔽的处理器外，所有处理器同时执行该操作；在 SIMD 模型中，Send 语句 Receive 语句起同步作用. Receive 语句使处理器在未收到数之前一直等待，收到数时发出回答信号. Send 语句在发出数据后在处理器未得到回答信号之前等待回答信号. 下面给出算法：

Bitonic Sorting (n)

/* $n = 2^m$，P_i 具有至少 2 个寄存器 $A[i]$ 和 $B[i], 0 \leqslant i < n$. 排序开始时 n 个数分别存放于 $A[0], A[1], \cdots, A[2^m - 1]$ 之中. 排序结束时，$A[0] \leqslant A[1] \leqslant \cdots \leqslant A[2^m - 1]$ */

Begin

```
for j←1 to m step 1 do
  for k←j to 1 step -1 do
    for i←0 to n-1 step -1 do in parallel
      if Mark(i,j,k)=1
      then P_i Send A[i] to P_i'
      else P_i Receive B[i] from P_i'
      if Mark(i,j,k)=1
      then P_i Receive A[i] from P_i'
      else if A[i]<B[i]
        then P_i Send B[i] to P_i'
        else begin
        P_i Send A[i] to P_i'
        Move B[i] to A[i]
        end
      endfor
    endfor
  endfor
End
```

上述算法中第3句中 in parallel 指出该步为一个并行步，即所有处理器同时执行该循环体内的语句. 因此整个算法的执行时间为 $O(m^2) = O(\log n)$.

替换上述算法中的 Receive 语句和 Send 语句可以将其移植到许多并行模型，如 EREW，CCC，网孔，洗牌交换网等.

参考文献：

[1] LEE T H, CHOU J J. Some topological properties of bitonic sorters. IEEE T C, 1998, 47 (9): 983 ~ 997.

[2] LIN Y C. Perfectly overlapped merging and sorting on a two-way linear array. Information Processing Lettersk, 1996, 60: 183 ~ 187.

[3] BATCHER K E. Sorting Networks and their applications. Proc AFIPS. 1968 Spring Joint Comput. Conf, Atlanbic City, New Jersey, 1968. 307 ~ 314.

[4] TANENBAUM A S. 计算机网络（第三版）. 北京：清华大学出版社，1996. 151 ~ 155.

[5] STONE H S. Parallel processing with the perfect shuffle. IEEE T C, 1981, 28 (12): 907 ~ 917.

A Uniform Marking Method for Bitonic Sorting

ZHANG Yong-ming *, *ZHANG Yong-dong*

Abstract: A Uniform marking method for bitonic sorting is presented to simplify the designing of paralled bitonic algorithms on different paralled models.

Keywords: bitonic sorting; parallel algorithm; proccessor arrays; marking method; hypercube

* Department of Computer Sciences, Zhongshan University, Guangzhou 510275, China