Information Processing Letters 35 (1990) 313-316
North-Holland

15 September 1990

EFFICIENT PARALLEL k SELECTION ALGORITHM

Jang-Ping SHEU

Department of Electrical Engineering, National Central University, Chungli, 32054, Taiwan

Jyh-Shyan TANG

Institute of Information Engineering, Tatung Institute of Technology, Taipei, Taiwan

Communicated by K. Ikeda
Received 25 December 1989
Revised 25 April 1990

In this paper, a parallel algorithm to select the first k largest numbered processes in an n-cube network is proposed. The time complexity of the algorithm is max(O(k), O(n^2)) and it is optimal when $k \geqslant n^2$.

Keywords: Parallel algorithm, k selection, n-cube networks

1. Introduction

Election is the problem of choosing a unique processor as the leader of a network of processors. The election problem was first discussed by Lelann [7] in a ring connection network to elect a new leader as responsible for regenerating a new control token after the previous token is lost in the ring. He first poses the election problem and gives a solution with time complexity $O(N^2)$ in a unidirectional ring, where N is the number of processors in the ring network.

Peterson [9] presents an algorithm with $O(N \log N)$ messages in the worst case and he also disproves Hirschberg and Sinclair's [4] conjecture that any unidirectional solution must be $\Omega(N^2)$. The upper bound for the unidirectional ring with the smallest leading constant is obtained in [3]; it needs $(1.356\ N \log N) + O(N)$ messages in the worst case. Burns [1] has a bidirectional $O(N \log N)$ algorithm and gives an $\Omega(N \log N)$ lower bound for the bidirection case. Korach et al. [6] and Loui et al. [8] study the election problem in a complete network.

Rotem, Korach, and Santoro [10] present a k selection algorithm in a ring. The average number of messages transmitted for a k selection algorithm under the unidirectional ring is very closely approximated by $\ln N$ for large N. The objective of the k selection problem is that the largest process is the leader and the other processes are used as standby. If the leader fails, the next largest can immediately take control without restarting the election algorithm. Sheu and Wu [11] present a parallel algorithm to select the first k largest numbered processes in n-cube networks. The time complexity of their algorithm is $O(k(n - \log k) + k)$. It is optimal when $k = 1$ or $k = N$ $(= 2^n)$. In this paper, we shall propose a parallel algorithm to determine the first k largest numbered processes in n-cube networks. The time complexity of the algorithm is max(O(k), O(n^2)) and it is optimal when $k \geqslant n^2$.

2. Parallel k selection algorithm

In this section, we propose a parallel algorithm for the k selection problem in an n-cube network.

The n-cube network is a hypercube of dimension n. It consists of $N = 2^n$ identical nodes so that every node is a general-purpose processor with its own local memory. Every node is numbered from 0 to $N - 1$ by an n-bit binary number $(a_n a_{n-1} \dots a_1)$. If two nodes have node numbers different in a_j only, then they are called opposite ones in the jth direction. Every node has a direct link to the opposite node. So every node has n-neighbor nodes with direct links.

In order to study the efficiency of different algorithms, we not only use the computation time but also use the communication time as the measure in any execution of these algorithms. We measure the communication steps because it reflects the communication overhead (which may dominate the computation time) on the communication system of the n-cube networks. The following assumptions are used to analyze the algorithm complexity:

(1) Moving a fixed-sized data packet from one node to a neighbor one figures as a message complexity and takes a unit time;

(2) the data communication is bidirectional;

(3) each node has memory of size $O(k)$.

In the beginning, each process running at a node in the n-cube is uniquely identified by an integer number. All processes are identical except that each one has its own number. The k selection problem is to find the first k largest numbered processes and let all processes know where these k processes reside. Sheu and Wu [11] had proposed an algorithm to solve this problem. The time complexity of their algorithm is comparable to the lower bound when k is a small constant or approaches to N. However, the performance of their algorithm is poor, as k approaches to n or to a polynomial of n. Here, we propose a k selection algorithm that is optimal when $k \geqslant n^2$.

The basic idea of our algorithm is described as follows. First, the N process numbers in an n-cube are sorted in descending order [5]. The Johnsson's parallel sorting algorithm is based on Batcher's bitonic sort. First, Johnsson's algorithm takes $\frac{1}{2}n(n+1)$ steps to obtain $\frac{1}{2}N$ elements with ascending order in the half cube with highest address bit 0, and the other $\frac{1}{2}N$ elements with descending order in the half cube with highest address bit 1. Then two sorted subcubes of equal length can be merged in n steps.

After sorting, the ith largest process number is located in the ith node whose address number is $i-1$. Then the ith node, for $1 \leqslant i \leqslant k$, sends its process number to its opposite node in the nth direction. After this step, there are $2k$ nodes each with one of the first k largest process numbers. These $2k$ nodes also send their process numbers to the opposite node in the $(n-1)$th direction again. We repeat the above operations $n - \lceil \log k \rceil$ times by letting every node of the n-cube get one of the first k largest process numbers. Note that there are some $(2^n - k2^{n-\lceil \log k \rceil})$ nodes having no such process number when $k \neq 2^i$. After the above procedure, we merge the data of each node along the first direction of its opposite node to the $\lceil \log k \rceil$th direction. In each merge step, we insert data received from the opposite node in the jth direction into the rear (front) of the original data list as the jth bit is 0 (1). The formal algorithm of the k selection problem is described as follows.

Algorithm PKS (Parallel k selection).

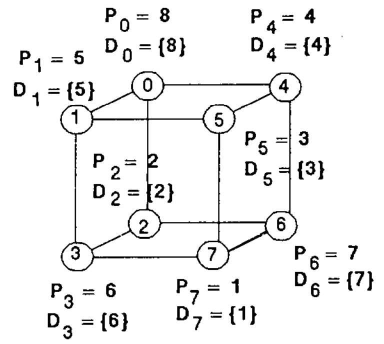
```
Step 1: Sort the N process numbers of an n-cube in descending order. After sorting, the ith largest number is located in the ith node.
Step 2:
  (2.1) Each of the first k nodes sends its process number to its opposite nodes from the nth direction to the (⌈log k⌉ + 1)th direction.
  (2.2) Besides, each node which receives process number from its opposite node in the ith direction also sends the process number to its opposite nodes from the (i − 1)th direction to the (⌈log k⌉ + 1)th direction.
Step 3: For i = 1 to ⌈log k⌉ do
      /* Data exchange phase */
  (3.1) Each node exchanges its accumulated process numbers with its opposite node in the ith direction.
      /* Data accumulated phase */
  (3.2) After data exchanging, if the ith bit of a node is 0 (1), it will insert the data received from the opposite node in the ith direction into the rear (front) of the original data list.
```

Example 2.1. In the following, we show how to get the first two largest process numbers in a 3-cube.

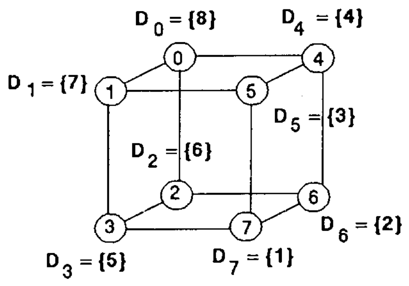
P_i: Represents the process number residing in node i.

D_i: Represents a temporary buffer which will be used to store the first k largest process numbers at node i.

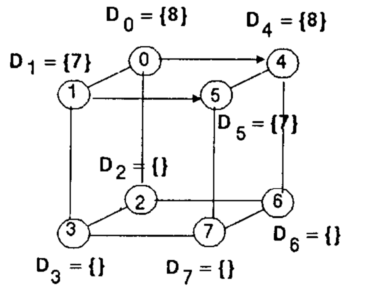
Initially, P_i and D_i of node i are shown in Fig. 1(a). After sorting the 8 process numbers, the ith largest number is located in the ith node, which is shown in Fig. 1(b). The arrows in Fig. 1(c) show the actions in the third direction (bit 3). The node 0 sends the largest process number (8) to node 4 and the node 1 sends the second largest process number (7) to node 5. The result is shown in Fig. 1(c). After sending in the second direction (bit 2), the result is shown in Fig. 1(d). Finally, all nodes exchange their data in the first direction (bit 1)



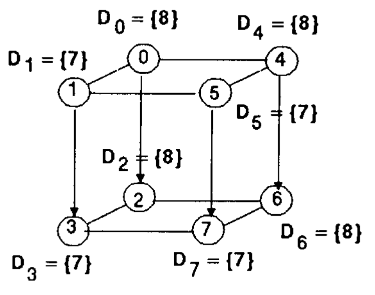
(a) Initial



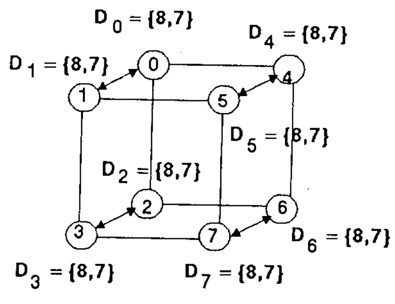
(b) After completion of sorting



(c) After sending of the third direction



(d) After sending of the second direction



(e) After data exchange phase of the first direction

Fig. 1. Data exchanges and ordered data lists for $k = 2$ selection in a 3-cube.

and insert data received from the opposite node into the rear or front that depends on the bit being 0 or 1. For example, the node 0 inserts process number (7) into the rear of the original data list and the node 1 inserts process number (8) into the front of the original data list. The final result is shown in Fig. 1(e).

In the following, we shall analyze the time complexity of Algorithm PKS. First, the time complexity of sorting N data in an n-cube is $O(n^2)$ [5]. In Step 2, consisting of $n - \lceil \log k \rceil$ iterations, each node sends at most one data packet to its opposite node in each iteration. In Step 3, consisting of $\lceil \log k \rceil$ iterations, each node sends twice as many data packets to the previous iteration. Thus, the time complexity $T(n)$ for an n-cube is derived as follows:

$$\begin{aligned} T(n) &= O(n^2) + n - i \\ &\quad + (2^0 + 2^1 + \cdots + 2^{i-2} + 2^{i-1}) \\ &= O(n^2) + n - i + k - 1, \end{aligned}$$

where $i = \lceil \log k \rceil$.

Hence, the time complexity of this algorithm is equal to $\max(O(k), O(n^2))$. The lower bound of the k selection problem is

$$\left[\log N - \lceil \log k \rceil + 1 + k(1 - 2^{1-\lceil \log k \rceil})\right]$$

[11]. Without loss of generality, let $k = 2^i$, then the lower bound of the k selection problem is $n - i + k - 1$. Therefore, our algorithm is optimal when $k \geqslant n^2$.

3. Conclusions

The advantage of selecting the first k largest process numbers is that every node knows where the first k largest numbered processes are as well as that the first k largest process numbers can determine the ranks. Algorithm PKS is proposed for selecting the first k largest numbered processes in an n-cube network. The time complexity of Algorithm PKS is $\max(O(n^2), O(k))$ and it is optimal when $k \geqslant n^2$. On the theoretical EREW PRAM model, the sorting problem can be completed on $O(n)$ time using N processors [2]. Under the PRAM model, the k-selection problem can be solved in $O(n)$ time.

References

[1] J.E. Burns, A formal model for message passing systems, Tech. Rept. 91, Computer Science Department, Indiana University, Bloomington, IN (1980).

[2] R. Cole, Parallel merge sort, *SIAM J. Comput.* **17** (4) (1988) 770–785.

[3] D. Dolev, M. Klawe and M. Rodeh, An $O(n \log n)$ unidirectional distributed algorithm for extreme finding in a circle, *J. Algorithms* **3** (1982) 245–260.

[4] D.S. Hirschberg and J.B. Sinclair, Decentralized extrema-finding in circular configurations of processors, *Comm. ACM* **23** (1980) 627–628.

[5] S.L. Johnsson, Combining parallel and sequential sorting on a boolean n-cube, in: *International Conference on Parallel Processing* (1984) 444–448.

[6] E. Korach, S. Moran and S. Zaks, Tight lower and upper bounds for some distributed algorithms for a complete network of processors, in: *Proc. 3rd Annual ACM symposium on Principle of Distributed Computing* (1984) 199–207.

[7] G. Lelann, Distributed systems—towards a formal approach, *IFIP Inform. Process.* **77** (1979) 155–160.

[8] C.M. Loui, A.T. Matsushita and D.B. West, Electron in a complete network with a sense of direction, *Inform. Process. Lett.* **22** (1986) 185–187.

[9] G.L. Peterson, An $O(n \log n)$ unidirectional algorithm for the circular extrema problem, *ACM Trans. Program. Lang. Syst.* **4** (4) (1982) 758–762.

[10] D. Rotem, E. Korach and N. Santoro, Analysis of a distributed algorithm for extrema finding in a ring, *J. Parallel and Distributed Comput.* **4** (1987) 575–591.

[11] J.P. Sheu and C.L. Wu, Selection of the first k largest processes in hypercubes, *Parallel Comput.* **11** (3) (1989) 381–384.