Selection on k-Dimensional Meshes with Multiple Broadcasting

YI PAN, MOUNIR HAMDI[1], GURDIP SINGH[2]

Department of Computer Science, University of Dayton, Dayton, OH 45469-2160, USA,
[1]Department of Computer Science, Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong and [2]Department of Computing and Information Sciences,
Kansas State University, Manhatton, KS 66506, USA
email: pan@hype.cps.udayton.edu

Randomized selection algorithms on k-dimensional mesh-connected computers with multiple broadcasting are proposed in this paper. We first show that a leader can be elected in $O(\log N)$ time on any k-dimensional mesh-connected computers with multiple broadcasting of size N. We then show that we can find the p-th smallest element among a data set of size N in $O((\log N + k + N^{1/(k(k+1))})\log N)$ expected time using a regular $N^{1/k} \times \ldots \times N^{1/k}$ k-dimensional mesh and in $O((\log N + k^2 N^{1/(k2^k)})\log N)$ expected time using an irregular $N^{(2^{k-1}k+1)/(k2^k)} \times N^{(2^{k-2}k+1)/(k2^k)} \times \ldots \times N^{(k+1)/(k2^k)}$ k-dimensional mesh. This leads to a selection algorithm which runs in $O((\log N)^2)$ expected time on a regular $((\log N/\log\log N)^{1/2})$-dimensional mesh or on an irregular $(\log\log N)$-dimensional mesh each with N processors. To our best knowledge, this is the first polylogarithmic selection algorithm on meshes with multiple broadcasting.

Received May 25, 1995; revised January 18, 1996

1. INTRODUCTION

The problem of selection has a number of applications in computer science, computational geometry and statistics. In particular, computing the median element of a set of data is a standard procedure in statistics analysis. Selection also has application in image analysis. In a database context, selection amounts to answering a query on the collection X of records. Many algorithms such as parallel merging, sorting and convex hull computation, use selection as a procedure [4]. Selection can be stated formally as follows. Given a list X of N elements whose elements are in random order and an integer satisfying $1 \leqslant p \leqslant n$, find the p-th smallest element in X.

Many parallel selection algorithms have been designed on different models to speed up the computation for selection. Parallel selection algorithms on shared memory models were discussed in [3, 9]. A number of algorithms exist for selection on a tree-connected computer [1, 16]. Parallel selection algorithms on reconfigurable linear arrays and meshes are discussed in [10, 11, 13, 14]. The selection problem has also been tackled on variants of basic 2-dimensional mesh-connected models with a broadcast capability [5, 6, 8, 12, 15].

In this paper, a new and efficient algorithm design methodology for solving the selection problem on higher-dimensional meshes with multiple broadcasting is proposed. The purpose of this work is to show that just like semi-group computations, selection computations can also be performed much faster on meshes with higher dimensions. We discuss selection computations on both regular and irregular meshes with multiple broadcasting. Here, regular meshes are meshes with equal size along each dimension and irregular meshes are those which have different sizes along each dimension. We propose a selection algorithm which runs in $O((\log N + k + N^{1/(k(k+1))})\log N)$ expected time using a regular $N^{1/k} \times \ldots \times N^{1/k}$ k-dimensional mesh and in $O((\log N + k^2 N^{1/(k2^k)})\log N)$ expected time using an irregular $N^{(2^{k-1}k+1)/(k2^k)} \times N^{(2^{k-2}k+1)/(k2^k)} \times \ldots \times N^{(k+1)/(k2^k)}$ k-dimensional mesh. This leads to a selection algorithm that runs in $O((\log N)^2)$ expected time on a regular $((\log N/\log\log N)^{1/2})$-dimensional mesh or on an irregular $(\log\log N)$-dimensional mesh. To our best knowledge, this is the first polylogarithmic selection algorithm on meshes with multiple broadcasting.

2. SELECTION ON REGULAR k-DIMENSIONAL MESHES

In this section, we describe an algorithm for selecting the p-th smallest element in X from among N given unsorted data items on a regular k-dimensional mesh with each processor containing one data item. In a regular k-dimensional mesh of N processors, the length of each dimension is equal and is $N^{1/k}$. Thus, in the case of a 2-dimensional mesh, it is a square mesh and processors in the same row or column are connected to a bus in addition to the local links. Figure 1 shows an example for a 2-dimensional 4×4 mesh with multiple broadcasting. Hence, the mesh has the broadcasting capability in each row and column. There are two types of data transfers executed by each processor: routing data into one of its four nearest neighbors via a local link and broadcasting

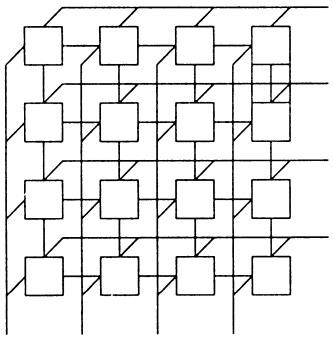

FIGURE 1. A 2-dimensional 4 × 4 mesh with multiple broadcasting.

data to the other processors in the same row (or column) via the row bus (or column bus). Similarly, in a k-dimensional mesh, broadcasting can be performed on a bus along any dimension. As in [8, 12] it is assumed that a broadcast takes constant time and only one processor is allowed to execute a successful broadcast on a bus at any one time. When two or more processors try to broadcast data on the same bus at the same time, a collision is detected by all the processors on the bus and no data are successfully transferred. The collision detection mechanism of a bus system is a reasonable assumption and has actually been implemented in many bus systems.

Our selection algorithm proceeds along lines similar to those [2]. The divide-and-conquer strategy is applied to solve the selection problem efficiently. Every iteration of the algorithm involves a current set C of candidates, that is, elements of the original input set A that have a chance of being selected as the p-th smallest element of A. In every iteration, we partition the set C into three disjoint subsets

$$C_1 = \{c \in C | c < m\},$$
$$C_2 = \{c \in C | c = m\},$$
$$C_3 = \{c \in C | c > m\},$$

where m is a distinguished element of C. The procedure to choose the partition element m will be described later.

In case $|C_1| \geqslant p$, we eliminate C_2 and C_3 and proceed recursively to solve the problem of selecting the p-th smallest element in C_1. In case $|C_1| < p$ and $|C_1 \cup C_2| \geqslant p$, the desired element is m; finally, if $|C_1 \cup C_2| \geqslant p$ then we proceed recursively to select the $(p|C_1 \cup C_2|)$-th smallest element in C_3. In this manner we can replace the given problem by a smaller problem. This process is continued until the second condition above is satisfied and we find the p-th smallest element in the set A.

In the algorithm, a current set of candidates is maintained through a set of variables, IN, one at each processor. For any IN whose value is 1, the corresponding data item D in the same processor is in the current set and will participate in future selection processes. Initially, all data items are in the current set. We equate active processors to those processors that contain a data item in the current set. In the following presentation, we assume that the N elements are unique. We can assume the numbers are unique without loss of generality since if we are given arbitrary numbers $x_0, x_1, \ldots x_{N-1}$, we can replace x_i by (x_i, i) and define an order of the tuples by $(x_i, i) < (x_j, j)$ if $x_i < x_j$ or if $x_i = x_j$ and $i < j$.

Before we describe the selection algorithm, we present two simple procedures. The first one is a broadcast operation and the second one is a leader election algorithm which elect a leader among the active processors. The broadcast operation can be performed as follows.

Assume that $PE(i_{k-1}, \ldots, i_1, i_0)$ is the source processor in a broadcast operation. $PE(i_{k-1}, \ldots, i_1, i_0)$ broadcasts its data item through its bus on dimension 0 to all processors in $A(i_{k-1}, \ldots, i_1, *)$. All the processors in the subarray then broadcast it through the buses on dimension 1 to all processors in $A(i_{k-1}, \ldots, i_2, *, *)$. In general, processors then broadcast it through the buses on dimension e to all processors in $A(i_{k-1}, \ldots, i_{e+1}, *, \ldots, *)$. After k such broadcasts, all processors receive the data item from $PE(i_{k-1}, \ldots, i_1, i_0)$. This procedure requires $O(k)$ time.

Leader election is also very useful in situations where a unique processor is to be identified to perform certain functions. In our algorithm, an elected leader may not hold the largest data item in the current set. The important thing is that the elected processor is unique and active. The simple leader election procedure is described as follows.

First, a leader is elected among the active processors for each subarray $A(i_{k-1}, \ldots, i_1, *)$ in the mesh concurrently. We divide a subarray into two sections of equal length and randomly select a section. The selection of a section can be performed by a designated processor; e.g. $A(i_{k-1}, \ldots, i_1, 0)$ in subarray $A(i_{k-1}, \ldots, i_1, *)$. All the active processors in the chosen section transmit their addresses to their row buses. In the case that no active processor is in the section, we can select the other section. If a conflict is detected, we further divide the section into two subsections and repeat the transmission and detection process. The above process is repeated until no conflict is detected in all the subarrays. Now, each subarray consists of only one active processor which also has a data item in the current set. Processors selected above each send their data items to processors $A(i_{k-1}, \ldots, i_1, 0)$ through the buses on dimension 0. Clearly, the whole process takes $O(\log N^{1/k})$ time.

We repeat the above process for subarrays $A(i_{k-1}, \ldots, i_2, *, 0)$ and put the selected data item in $PE(i_k - 1, \ldots, i_2, 0, 0)$. In general, we randomly select an active processor in each subarray $A(i_{k-1}, \ldots, i_e, *, 0, \ldots, 0)$ and put the selected data item in $PE(i_{k-1}, \ldots, i_e, 0, \ldots, 0)$. After k such selections, we obtain an active data item in the current set in $PE(0, 0, \ldots, 0)$. The active processor holding the data item is a leader. If the data set has duplicates, we can send a selected data item along with the ID of the processor which holds it. After the above election process,

$PE(0, 0, \ldots, 0)$ can inform all processors of the leader elected in the mesh by a broadcast.

Since each selection along a dimension takes $O(\log N^{1/k})$ time and we have k dimensions, we need $O(\log N)$ time to elect a leader. The time to broadcast the leader ID is $O(k)$. Hence, the time taken in the leader election algorithm is $O(\log N + k)$.

The selection algorithm on a regular k-dimensional mesh with multiple broadcasting is described formally in REGULAR-SELECT. In the algorithm, a mesh with k dimensions is denoted as $A(*, *, \ldots, *)$ and a processor with indices $i_k, \ldots, i_1, i_0$ is represented as $PE(I_k, \ldots, i_1, i_0)$. Thus, $A(*, *)$ is a 2-dimensional mesh and $A(*, k)$ is a subarray (its k-th column of processors).

2.1. Algorithm REGULAR-SELECT (*D, IN, B, p*)

Input: A data vector D of length N and an integer p are distributed in an k-dimensional $N^{1/k} \times N^{1/k} \ldots \times N^{1/k}$ mesh with multiple broadcasting. Each processor contains one data item. Also assume that initially, all processors participate in the selection process; i.e. all INs contain a value of 1.

Output. The p-th smallest element of the data vector D is found and stored in memory cell m of all processors.

Step 1. We first elect a leader and select a unique data item using the algorithm described before. This step takes $O(\log N + k)$ time.

Step 2. The leader elected in step 1 broadcasts its local data item. The data item is to be used as the partition number m in the following steps. This step requires $O(k)$ time.

Step 3. All processors holding an active data item compare the received value m with its local data items. If m is larger than the data item, set its local B to 0, indicating that the data item is in C_1; otherwise, set B to 1, meaning that the data item is in C_2 or C_3. The time taken is $O(1)$ in this step.

Step 4. Perform a binary summation over B and IN cross the whole mesh and put the two sums in s and t, respectively. Clearly, s is the size of C_3 or the number of data elements, in the current set of candidates, which are larger than the distinguished number m; and t is the size of the current set of candidates. This step basically performs two semigroup operations and can be performed in $O(N^{1/(k(k+1))})$ time according to the result in [12].

Step 5. Calculate $|C_1| = t - s - 1$, $|C_2| = 1$ and $|C_3| = s|$. Here, $|C_i|$ is the size of the set C_i for $i = 1, 2, 3$. If $|C_i| \geqslant p$, then the p-th smallest element is in C_1. Hence, we set $IN = 0$ for all those processors whose $B = 1$ to eliminate C_2 and C_3. Also change their local B to 0. In this way, these processors become passive and will not participate in future computation. Then, we recursively call REGULAR-SELECT(D, IN, B, p). If $|C_1| < p$ and $|C_1| + |C_2| \geqslant p$, then the p-th smallest is the distinguished number m, stop. If $|C_1| + |C_2| < p$, then the p-th smallest element is in C_3. Hence, we set $IN = 0$ for all those data items whose corresponding $B = 0$ or the data item whose value is m to eliminate data elements in C_1 and C_2. Then, we recursively call REGULAR-SELECT(D, IN, B, $p - (|C_1| + |C_2|)$). Obviously, this step takes a constant time.

Since the above algorithm runs recursively, we need to determine the total time for each iteration first. According to the above description, it is clear that each iteration requires $O(\log N + k + N^{1/(k(k+1))})$ time.

We now calculate the average number of iterations for this algorithm. Before we can talk about the average number of iterations of an algorithm, we must agree on what the probability distribution of the inputs is. For selection, a natural assumption, and the one we shall make, is that every permutation of the set of numbers to be selected is equally likely to appear as an input. For simplicity in the timing analysis assume that all elements of the set to be selected are distinct. This assumption will maximize the size of C_1 and C_3 constructed, and therefore maximize the average time spent in the recursive calls. Let $R(N)$ be the expected number of iterations required by REGULAR-SELECT to select the p-th smallest element in a set of N elements. Clearly, $R(0) = R(1) = 1$. In the best case, m is the p-th smallest element and we do not need to continue calling REGULAR-SELECT recursively. On the other hand, in the worst case only one element is eliminated after each iteration. In general, the selection problem of size N is reduced to a subproblem of size i, $0 \leqslant i < N$, after each iteration. Since i is equally likely to take on any value between 0 and $N - 1$, we have the following relationship:

$$R(N) \leqslant \frac{1}{N}\sum_{i=0}^{N-1}(R(i) + 1) \leqslant \frac{1}{N}\sum_{i=0}^{N-1} R(i) + 1. \quad (1)$$

We shall show that for $N \geqslant 2$, $R(N) \leqslant 4 \log_e N$. For the basis $N = 2$, $R(2) \leqslant \frac{1}{2}\Sigma_{i=0}^{1} R(i) + 1 = 2$ from (1), which is smaller than $4 \log_e 2$. For the induction step, write (1) as

$$R(N) \leqslant \frac{R(0) + R(1)}{N} + \frac{1}{N}\sum_{i=2}^{N-1} R(i) + 1$$

$$\leqslant \frac{2}{N} + \frac{4}{N}\sum_{i=2}^{N-1} \log_e i + 1. \quad (2)$$

Since $\log_e i$ is concave upwards, it is easy to show that

$$\sum_{i=2}^{N-1} \log_e N \leqslant \int_2^{N-1} \log_e x dx$$

$$\leqslant N \log_e N - N - 2 \log_e 2 + 2. \quad (3)$$

Substituting (3) in (2) yields

$$R(N) \leqslant \frac{2}{N} + \frac{4}{N}(N \log_e N - N - 2 \log_e 2 + 2) + 1$$

$$\leqslant 4\log_e N - \frac{(3N + 8\log_e 2 - 8 - 2)}{N}. \quad (4)$$

Since $N \geqslant 2$, it follows that $3N + 8\log_e N - 10 \geqslant 0$. Thus, $R(N) \geqslant 4\log_e N$ follows from (4); i.e. the total average number of iterations of the selection algorithm for a set of size N is at most $4\log_e N$. The total expected time of the selection algorithm is the product of the average number of iterations and the time spent in each iteration. Therefore, we have the following theorem.

THEOREM 1

The REGULAR-SELECT algorithm selects the p-th smallest element in a data set of N elements on an $N^{1/k} \times \ldots \times N^{1/k}$ k-dimensional mesh in $O((\log N + k + N^{1/(k(k+1))})\log N)$ expected time.

It is interesting to analyse the time complexity in Theorem 1. Assume that the size of the selection problem is N, different k will result in different time complexity for the algorithm. Let $N^{1/(k(k+1))} \leqslant O(\log N)$, we get $k \geqslant (\log N/\log\log N)^{1/2}$. Thus, if $(\log N/\log\log N)^{1/2} \leqslant k \leqslant \log N$, the expected time complexity of algorithm REGULAR-SELECT is $O((\log N)^2)$.

3. SELECTION ON AN IRREGULAR k-DIMENSIONAL MESHES

In this section, we show that a better result can be obtained if an irregular k-dimensional mesh with multiple broadcasting is used. Here, the sizes of the k dimensions in the mesh are different. In the case of two dimensions, the mesh is called rectangular. Let us consider an $N^{(2^{k-1}k+1)/(k2^k)} \times N^{(2^{k-2}k+1)/(k2^k)} \times \ldots \times N^{(k+1)/(k2^k)}$ k-dimensional mesh. The previous algorithm can still be applied here. Step 1 of the algorithm uses $((2^{k-1}k+1)/(k2^k) + (2^{k-2}k+1)/(k2^k) + \ldots + (k+1)/(k2^k))\log N = ((1/2 + 1/4 + \ldots + 1/2^k) + 1/2^k)\log N = \log N$ time using the previous transmission and detection algorithm. Step 2 remains $O(k)$ time. Steps 3 and 5 use $O(1)$ time as before. Step 4 performs two semigroup operations on the irregular k-dimensional mesh and requires $O(k^2 N^{1/(k2^k)})$ time according to [7]. Thus, we have the following theorem.

THEOREM 2

We can select the p-th smallest element in a data set of N elements on $N^{(2^{k-1}k+1)/(k2^k)} \times N^{(2^{k-2}k+1)/(k2^k)} \times \ldots \times N^{(k+1)/(k2^k)}$ k-dimensional mesh in $O((\log N + k^2 N^{1/(k2^k)})\log N)$ expected time.

Clearly, the above time complexity is lower bounded by $O((\log N)^2)$. Now let us calculate when it will reach the lower bound. Assume that

$$N^{1/(k2^k)} \leqslant O(1), \quad (5)$$

we have

$$k2^k \geqslant \log N. \quad (6)$$

Obviously, when

$$2^k \geqslant \log N, \quad (7)$$

inequality (6) holds. Thus, when $k \geqslant \log\log N$ and $k^2 \leqslant \log N$, inequality (5) is satisfied and the expected time complexity of the selection algorithm is $O((\log N)^2)$.

CONCLUSIONS

In this paper, the selection problem is discussed on k-dimensional meshes with multiple broadcasting. It is clear from the discussion that we can reduce the time complexity of the selection problem by increasing the number of dimensions of the meshes. Two types of meshes are discussed in this paper. One is a regular mesh where all dimensions have the same size. The other is an irregular mesh where different different dimensions have different sizes. It is shown that an $O((\log N)^2)$ expected time selection algorithm can be achieved on a $(\log N/\log\log N)^{1/2}$-dimensional regular mesh of size N and on a $(\log\log N)$-dimensional irregular mesh. We conclude that better time complexity for the selection problem can be achieved on meshes with higher dimensions, and fewer dimensions are needed on an irregular mesh than on a regular mesh to achieve the same $O((\log N)^2)$ expected time complexity.

It is clear that the dominating factor of our algorithm is the leader election algorithm for higher dimensional meshes. If we could solve it in $o(\log N)$ time, then we can reduce the total time complexity of the selection algorithms presented in this paper. Since it is possible to elect a leader on a bus system with an average time complexity of $O(\log\log N)$ [17], it would be of interest to know whether $O(\log\log N)$ time algorithms are also achievable for electing a leader on k-dimensional meshes with multiple broadcasting. This promises to be an exciting area for further research.

ACKOWLEDGEMENTS

Y. Pan is supported in part by the NSF Research Opportunity Award CCR-9540901 and by the Research Council of the University of Dayton under Grant 94063-13. M. Hamdi is supported in part by the Hong Kong Research Grant Council under Grant RGC/HKUST 619/94E. G. Singh is supported in part by the NSF Research Initiation Award CCR-9211621 and the NSF CAREER Award CCR-9502506.

REFERENCES

[1] Aggarwal, A. (1984) A comparative study of X-tree, pyramid and related machines. *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*, Singer Island, FL, October, pp. 89–99.
[2] Aho, A., Hopcroft, J. and Ullman, J. (1974) *The design and analysis of computer algorithms*. Addison-Wesley Publishing Company, Reading, MA.
[3] Akl, S. G. (1984) An optimal algorithm for parallel selection. *Information Processing Lett.*, **19**, 47–50.

[4] Akl, S. G. (1989) *The Design and Analysis of Parallel Algorithms*. Prentice-Hall, Inc, Englewood Cliffs, NJ.
[5] Bhagavathi, D., Looges, P. J., Olariu, S., Schwing, J. L. and Zhang, J. (1993) Selection on rectangular meshes with multiple broadcasting. *BIT*, **33**, 7–14.
[6] Bhagavathi, D., Looges, P. J., Olariu, S., Schwing, J. L. and Zhang, J. (1994) A fast selection algorithm for meshes with multiple broadcasting, *IEEE Trans. Parallel Distributed Systems*, **5**, 772–778.
[7] Chen, Y. C., Chen, W. T., Chen, G. H. and Sheu, J. P. (1990) Design efficient parallel algorithms on mesh-connected computers with multiple broadcasting. *IEEE Trans. Parallel Distributed Systems*, **1**, 241–245.
[8] Chen, Y. C., Chen, W. T. and Chen, G. H. (1990) Two-variable linear programming on mesh-connected computers with multiple broadcasting. *International Conference on Parallel Processing*, St Charles, IL. Vol. III, CRC Press, Boca Raton, FL, pp. 270–273.
[9] Cole, R. and Vishkin, U. (1986) Deterministic coin tossing and accelerating cascades: Micro and macro techniques for designing parallel algorithms. *Proc 18th Annual ACM Symposium on Theory of Computing*, Berkeley, CA, pp. 206–219.
[10] ElGindy, H. and Wegrowicz, P. (1991) Selection on the reconfigurable mesh. *International Conference on Parallel Processing*, St Charles, IL, August 12–16. Vol. III, CRC Press, Boca Raton, FL, pp. 26–33.
[11] Hao, E., MacKenzie, P. D. and Stout, Q. F. (1992) Selection on the reconfigurable mesh. *The Fourth Symposium on the Frontiers of Massively Parallel Computation*, McLean, VA, October 19–21. IEEE CS Press, Los Alamitos, CA, pp. 38–45.
[12] Prasanna Kumar, V. and Raghavendra, C. S. (1987) Array processor with multiple broadcasting. *J. Parallel Distributed Comp.*, **4**, 173–190.
[13] Olariu, S., Schwing, J. L., Shen, W., Wilson, L. and Zhang, J. (1993) A simple selection algorithm for reconfigurable meshes. *J. Parallel Algorithms Appl.*, **1**, 29–41.
[14] Pan, Y. (1995) Order statistics on a linear array with a reconfigurable bus. *Future Generation Computer Systems*, **11**, 321–327.
[15] Stout, Q. F. (1983) Mesh Connected Computers with Broadcasting. *IEEE Trans. Comp.*, **32**, 826–830.
[16] Tanimoto, S. L. (1984) Sorting, histogramming, and other statistical operations on a pyramid machine. In Rosenfeld, A. (ed.), *Multiresolution Image Processing and Analysis*. Springer-Verlag, New York, pp. 136–145.
[17] Willard, D. E. (1986) Log-logarithmic selection resolution protocols in a multiple access channel. *SIAM J. Comp.*, **15**, 468–477.