International Journal of Parallel Programming, Vol. 18, No. 5, 1989

Pipelined Search on Coarse Grained Networks[1]

Selim G. Akl[2] and Frank Dehne[3]

Received September 1988; Revised March 1990

The time complexity of searching a sorted list of n elements in parallel on a coarse grained network of diameter D and consisting of N processors (where n may be much larger than N) is studied. The worst case period and latency of a sequence of pipelined search operations are easily seen to be $\Omega(\log n - \log N)$ and $\Omega(D + \log n - \log N)$, respectively. Since for $n = N^{1+\Omega(1)}$ the worst-case period is $\Omega(\log n)$ (which can be achieved by a single processor), coarse-grained networks appear to be unsuitable for the search problem. By contrast, it is demonstrated using standard queuing theory techniques that a constant expected period can be achieved provided that $n = O(N2^N)$.

KEY WORDS: Average-case analysis; coarse grained processor network; parallel algorithms; pipelining; searching.

1. INTRODUCTION

The parallel complexity of searching a sorted sequence $S = (s_1, \ldots, s_n)$ of size n for a given element s was studied by Snir[1] for an N-processor shared memory model of computation. For a variant of the model, where an item of information can be accessed by one processor only at a time, he showed that searching for one element requires at least $\Omega(\log n - \log N)$ steps. (Henceforth, all logarithms are with respect to the base 2.) In this short communication, we study the parallel complexity of searching for the more

[1] This research was supported by the Natural Sciences and Engineering Research Council of Canada under Grants A3336 and A9173.

[2] Department of Computing and Information Science, Queen's University, Kingston, Ontario K7L 3N6, Canada.

[3] Center for Parallel and Distributed Computing, School of Computer Science, Carleton University, Ottawa, Ontario K1S 5B6, Canada.

realistic model of a *processor network* of size N storing a sorted list of n elements.

Section 2 reviews some results on the worst case time complexity of parallel search in processor networks. For coarse grained networks, i.e., $n = N^{1+\Omega(1)}$, the worst case period of a sequence of pipelined search operations is $\Omega(\log n)$ which can also be achieved by a single processor. This seems to suggest that coarse grained networks are not particularly well suited, *in the worst case*, for the search problem. By contrast, it is shown in Section 3 that a constant expected period can be achieved on coarse grained networks provided that $n = O(N2^N)$. Therefore, coarse grained networks are good, *on the average*, for the search problem.

2. WORST CASE COMPLEXITY OF PIPELINED SEARCH ON COARSE GRAINED NETWORKS

In a processor network, a set of N processors $P_1, \ldots, P_N$ are connected by bidirectional unit-time communication links between pairs of processors (e.g., mesh-of-processors, tree, mesh-of-trees, pyramid, or hypercube architectures; see Akl[2]). By contrast with the model studied in Snir,[1] there exists no shared memory. Instead, each processor P_i has its own local finite memory M_i (which can solely be accessed by P_i); processors can communicate only by sending messages via the communication links. The distance between two processors P_i and P_j is the minimum number of direct communications links that a message has to traverse in order to travel from P_i to P_j. The diameter D of the network is the maximum distance between processors. The time complexity of a parallel algorithm executed on a network consists of the local computation time for the processors and the time for the messages sent via the communication links. As usual we count, for each processor, an arithmetic or comparison operation as well as a transmission of a word of length $O(\log n)$ bits to an adjacent processor as a constant time operation.

For the sequential model of computation, the worst case complexity of searching a sorted sequence of size n is $O(\log n)$ (this is achieved by binary search; see Aho *et. al.*[3]). When solving the search problem on a processor network, the sorted sequence S of n elements is distributed among the M_i's such that each receives a sorted subsequence of size n/N. A designated processor receives as input the element to be searched for and, in the worst case, must propagate it through the network to another processor P_i at a distance D away. In addition, P_i has to search its subsequence sequentially. Consequently, the worst case lower bound on the time required to search is $\Omega(D + \log(n/N))$, i.e. $\Omega(D + \log n)$ when $n = N^{1+\Omega(1)}$. For example, on the hypercube architecture where D equals $\log N$ the lower bound is $\Omega(\log n)$, which can be achieved sequentially by a single processor.

One justification for advocating parallel search is to improve throughput in the case of a stream of queries presented to the network in a pipelined fashion. The searching problem for processor networks within this setting is referred to as *pipelined* search:

- An infinite input sequence of search queries is sent, as an *input stream*, to a designated processor of the network, the *input processor*.
- The search queries are processed in a pipelined fashion and the answers are reported back via another processor, the *output processor* (which may coincide with the input processor).

The performance of a pipelined search algorithm is measured by

- the *latency*, i.e., the time between an arrival of a query at the input processor and the reporting of the result by the output processor, and
- the *period*, i.e., the time elapsed between the moments where the processing of any two consecutive queries begin.

The obvious lower bounds for latency and period are $\Omega(D)$ and $\Omega(1)$, respectively. Indeed, for N close to n, Dehne and Santoro(4,5) have presented algorithms for mesh and hypercube networks, with $O(D)$ latency and $O(1)$ period. This leads to m queries being processed in time $O(m+D)$. However, many current attempts at implementing databases on parallel computers involve *coarse-grained* networks. In these networks, each node is a relatively powerful processor with a significant amount of memory. An example of such a system is the Intel iPSC hypercube which consists of a relatively small number (between 8 and 512) of processors with up to 16 MBytes of memory each; for the iPSC/3, currently under development by Intel, a hard disk can even be attached to every subhypercube of two or four processors. Under these conditions, the search problem has to be solved for n much larger than N. It is easy to see that the worst case lower bounds on the period and latency are $\Omega(\log n - \log N)$ and $\Omega(D + \log n - \log N)$, respectively. When $n = N^{1+\Omega(1)}$, the worst case period is $\Omega(\log n)$. Consequently, the total time to perform pipelined search for m queries is $\Omega(D + m \log n)$ in the worst case. However, the same problem can be solved on a single processor storing n elements in time $O(m \log n)$ in the worst case. In other words, a single processor is at least as fast (perhaps even faster) than an N-processor network.

3. AVERAGE BEHAVIOR OF PIPELINED SEARCH ON COARSE GRAINED NETWORKS

The worst case lower bounds presented in Section 2 are based on the fact that *all* queries may refer to elements in the *same* subsequence S_i stored in processor P_i. Obviously, the likelihood of this happening is very low. Hence, from a practical point of view, it is also important to study the average case complexity of pipelined search on coarse-grained networks. In this section we present and analyse a simple pipelined search algorithm for coarse-grained networks which has a constant period if $n = O(N2^N)$.

For a given processor network, the standard single-source shortest path algorithm (see Aho *et. al.*[2], p. 207) determines the shortest path from the input processor to every one of the other processors. These shortest paths define a spanning tree of the network which will be referred to as the *routing tree*. Likewise, the shortest paths from all processors to the output processor define the *report tree*. The elements are stored such that every processor contains n/N elements in sorted order and the concatenation of these sorted subsequences, defined by the inorder traversal of the processors with respect to the routing tree, is again a sorted sequence. In addition to the data structure for searching S_i sequentially, every processor P_i maintains two queues for storing pending queries and results which will be referred to as the *search queue* and the *report queue* of P_i, respectively. The pipelined search algorithm consists of three processes which are executed simultaneously on all processors:

1) The *routing process*: Every query s arriving at the input processor is sent via the routing tree to the processor P_i with s in the interval $[\min(S_i), \max(S_i)]$. There, s is inserted at the end of the search queue of P_i.
2) The *searching process*: Every processor P_i continuously removes the first query s from its search queue, if one exists, and tests whether $s \in S_i$. The result is inserted at the end of the report queue of P_i.
3) The *reporting process*: Every processor P_i continuously removes the first result from its report queue, if one exists, and sends it to its successor on the shortest path from P_i to the output processor (as defined by the report tree). Every processor receives the results from its predecessors in round robin fashion and inserts them at the end of its report queue.

In the remainder of this section we will study the average case behavior of this algorithm under the following assumptions:

A1) For each processor P_i, the time to search its subsequence S_i sequentially is $c \log n/N$, for some constant c.

A2) The arrival of queries at the input processor is a Poisson process with constant period T; i.e., $P\{n_t = k\} = e^{-\alpha t}(\alpha t)^k/k!$, where $\alpha = 1/T$ and n_t is the number of arriving queries within a time period of length t.

A3) The probability is $1/N$ that the element searched for by a given query resides in a particular processor.

A4) The period T of the query stream is larger than the time to send a query result from one processor to an adjacent one.

Theorem. (a) The expected length of the search queue of an arbitrary processor P_i is at most *a* constant L provided that $n \leqslant N 2^{TN(L+1-(L^2+1)^{1/2})/c}$

(b) The expected length of the report queue of an arbitrary processor P_i is constant.

Proof. (a) Consider the search queue at an arbitrary processor P_i. From assumptions $A2$ and $A3$ it follows that the period of the queries which have to be handled by P_i is NT. Hence, the arrivals of these queries at P_i's search queue are a Poisson process with $P\{n_t = k\} = e^{-\lambda t}(\lambda t)^k/k!$, where $\lambda = 1/NT$. On the other hand, it follows from assumption $A1$ that the time between two results departing from P_i's search queue is fixed (for given n and N). Hence, the queue at P_i is an $M/D/1$ queue; see Papoulis.[6] Let $\rho = (c \log(n/N))/NT$. The queue is *stationary* if and only if $\rho < 1$; i.e., if $\rho < 1$ then the expected length of the queue is a constant, otherwise it becomes arbitrarily large. The expected length of an $M/D/1$ queue, $\rho(2-\rho)/2(1-\rho)$, is equal to a constant L if and only if $\rho = L + 1 - (L^2+1)^{1/2}$, i.e., $n = N 2^{TN(L+1-(L^2+1)^{1/2})/c}$. Since for all smaller n the time between two outgoing results from P_i decreases, and therefore also the expected length of the queue, part (a) follows.

(b) For every report queue, the period of the input stream is at least T. From assumption $A4$ it follows that T is larger than the period of the output stream of the report queue. Hence, the report queue is stationary; i.e., its expected length is a constant. □

Let D_{in} and D_{out} denote the expected minimum distance from the input and output processors to an arbitrary processor P_i, respectively. If $n \leqslant N 2^{TN(L+1-(L^2+1)^{1/2})/c}$ and assumptions $A1$ to $A4$ hold, then it follows from this theorem that the pipelined search algorithm has optimal expected period, a constant T, and optimal expected latency $O(D_{\text{in}} + D_{\text{out}} + \log(n/N))$, and the expected length of the search queue is a constant L.

ACKNOWLEDGMENTS

The authors wish to thank the referees for their constructive comments on the first version of this paper.

REFERENCES

1. M. Snir, On Parallel Searching, *SIAM J. Comput.* **14**:3: 688–708 (1985).
2. S. G. Akl, *The Design and Analysis of Parallel Algorithms*, Prentice-Hall (1989).
3. A. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley (1976).
4. F. Dehne and N. Santoro, Optimal VLSI Dictionary Machines on Meshes, in *Proc. Int. Conference on Parallel Processing*, pp. 832–840, St. Charles, Ill. (1987).
5. F. Dehne and N. Santoro, An Optimal VLSI Dictionary Machine For Hypercube Architectures, to appear in *Proc. Workshop on Parallel and Distributed Computing*, Bonas (France) (1988).
6. A. Papoulis, *Probability Theory, Random Variables, and Stochastic Processes*, McGraw-Hill (1984).