DECREMENTAL SCATTERING FOR DATA TRANSPORT BETWEEN HOST AND HYPERCUBE NODES

Mukesh Sharma and Meghanad D. Wagh
Department of Electrical Engineering and Computer Science,
Lehigh University, Bethlehem, PA 18015.
MS0G@Lehigh.Edu and MDW0@Lehigh.Edu

Abstract – *There are many parallel signal processing algorithms that are I/O bound. For such algorithms, fast data and program distribution is very important. In this paper we study a new strategy for data distribution from host to MIMD hypercube nodes. This strategy, based on scattering on hypercubes of decremental dimensions, performs much better than other strategies known today. It optimizes both the host and node utilization and also exploits the possible overlap in data sent to different nodes.*

I. INTRODUCTION

Recent years have seen dramatic improvements in the computational speeds of individual processors in a parallel architecture. Unfortunately the communication between processors has not improved to the same degree. Consequently, problems related with data communication in parallel architectures have assumed a greater importance.

In this paper we concentrate on applications such as image processing, pattern recognition and digital filtering. In all of these applications, there is a large amount of data used, a large number of output points computed and each output point computation is independent of the other. It is this last attribute that makes parallel architectures attractive for these applications [1]. In these I/O bound applications the total execution time is dominated by the data distribution and result collection delays [2, 3]. However the problem of data communication in this context is complicated because the data sets going to different processors overlap to some degree [4, 5, 6, 7] and the host to node communication has different characteristics from the node to node communication.

Some earlier strategies to reduce the time to distribute data from host to the hypercube nodes consisted of getting data into a single node and then broadcasting it from there [8]. This implies that while the broadcast is going on within the hypercube, the host is idle. Recently Prasad and Murty have given several new strategies to keep host busy concurrently with the node broadcast through scattering and multiple window broadcast [9]. However their work does not exploit the communication concurrency fully. In addition, their algorithms ignore the data overlap found in most common applications.

In this paper we derive a new strategy for host to hypercube node communication based on multiple communications from host to decreasing dimension hypercubes. We allow arbitrary overlaps between data sets going to different processors as well as differing characteristics of host-to-node and node-to-node communication. Our solution suggests that host of a dimension d hypercube should send messages sequentially to the roots subcubes of dimensions $d-1$, $d-2$, $\ldots$, $x+1$, x, x, where the value of x is determined by the expressions we have derived. Each root then scatters the data within the subcube. Analysis shows that this *decremental scattering* strategy has a superior performance.

II. PROBLEM AND ARCHITECTURE DESCRIPTION

The architecture model employed in these studies is a Multiple Instruction Multiple Data (MIMD) hypercube system with identical processors working asynchronously. A degree d hypercube has $p = 2^d$ nodes each labeled with a length d binary string. Two nodes whose labels differ in exactly one bit can communicate with each other and a message of length N takes $\beta + N\tau$ time to go across between them. The constants β and τ are the set-up and incremental times for node to node communication. We assume that the host is a processor separate from the 2^d hypercube nodes. In most real systems, such a host is the system interface to the outside world facilitating data transfer between the system and the storage devices, printers and the user. Earlier researchers have assumed that the host can communicate with any node of the architecture [9]. Our scheme needs communication with at most d properly chosen processors. Since host has a greater load, length N message transmission between the host and a node requires $\beta_h + N\tau_h$ time where β_h and τ_h are the set-up and incremental times for host to node communication. To simplify the analysis, we assume

Proceedings of the International Conference on Parallel Processing (ICPP'93)
0-8493-8983-6/0-7 $20.00 © 1993 IEEE

that the different communication parameters are related by

$$(\beta_h/\beta) = \sigma \quad \text{and} \quad \tau_h = \tau. \qquad (1)$$

Note that generally $\sigma > 1$ and $\tau_h = \tau$ because each depends on the transmission baud rate.

We consider applications in which the tasks in each processor are essentially independent. Examples of such applications arise in diverse applications. In digital filtering of a long sequence, each processor is required to compute one (or more) output points by taking inner product of a part of the input sequence with the filter sequence. The inner products being executed in distinct processors are independent of each other. Image processing through various two dimentional filters clearly fall in the same category. Pattern recognition problems can be solved by distributing the image within the processors and letting each search in its domain for the pattern independently. Note that even though the region along the border between two processors needs to be multiply assigned, there needs be no interaction between the neighboring processors.

In each of these applications, neighboring nodes share certain amount of data. We abstract this notion of data sharing as follows. Let S denote the set of input points and $S_i, 0 \leq i < p$ the set of points to be transported to processor i. Clearly, $S = \cup_{i=0}^{p-1} S_i$. We assume that all sets S_i are of equal size M, i.e., $|S_i| = M, \quad 0 \leq i < p$, and that the number of elements common to any pair of sets S_i and to S_{i+j} is independent of i, i.e.,

$$|S_i \cap S_{i+j}| = k_j, \quad 0 \leq i < p \quad \text{and} \quad j = 1, 2, \ldots$$

For efficient data transport, the important parameters are the set size M, and the number of *new* elements in set S_i beyond the collection of all earlier sets. We denote this number of extra elements by Δ and define it as:

$$\Delta = |\cup_{j=0}^{i} S_j| - |\cup_{j=0}^{i-1} S_j|.$$

The value of Δ is decided by the application and the algorithm. In a large number of Digital Signal Processing applications it is possible to predetermine the structure and details of the computations, allowing the user to detemine the data overlap. Many of the typical bilinear algorithms have as much as 70 to 80% redundancy in data sets required for various computations. If a data transport algorithm can exploit this redundancy, then the overall efficiency of the parallel computations will improve a great deal.

We calculate Δ in two sample cases. The first case, typical of digital filtering, is characterized by

$$|S_i \cap S_{i+j}| \supset |S_i \cap S_{i+j+1}|. \qquad (2)$$

Here, $M > k_1 > k_2 \ldots$ and

$$\Delta = |\cup_{j=0}^{i} S_j| - |\cup_{j=0}^{i-1} S_j|$$

$$= |\cup_{j=0}^{i-1} S_j| + |S_i| - |[\cup_{j=0}^{i-1} S_j] \cap S_i| - |\cup_{j=0}^{i-1} S_j| \quad (3)$$

The last step is obtained by using the Principle of Inclusion and Exclusion [10]. Now distributing the Intersection over Union in the above equation and using (2), we get from (3),

$$\Delta = M - k_1.$$

In the second case, we assume $k_3 = 0$. Thus,

$$|[\cup_{j=0}^{i-1} S_j] \cap S_i| = |\cup_{j=0}^{i-1} [S_j \cap S_i]|$$

$$= |[S_{i-1} \cap S_i] \cup [S_{i-2} \cap S_i]| \quad (4)$$

Equation (4) used $k_3 = 0$. Applying the principle of Inclusion and Exclusion [10] again, we get

$$|[\cup_{j=0}^{i-1} S_j] \cap S_i| = k_1 + k_2 - k_2', \qquad (5)$$

where k_2' denotes the cardinality of set $S_i \cap S_{i-1} \cap S_{i-2}$. Finally, combining the results of (3) and (5) we get:

$$\Delta = M - k_1 - (k_2 - k_2').$$

III. DATA DISTRIBUTION ALGORITHMS

Three methods of distributing data from host to all hypercube processors are available in the literature. The simplest method of required data transfer is Sequential Loading. In this method the host sends M elements to each of the p processors sequentially. This therefore entails a cost of

$$T_1 = p(\beta_h + M\tau_h) \qquad (6)$$

Clearly this method of *sequential loading* leaves much to be desired since its use of the available links is very sparse and consequently the communication cost very high.

In order to exploit the large number of links in the hypercube and the fact that communication between nodes is cheaper than that between host and nodes, Saad and Schultz have proposed [8] that the data be first downloaded from the host into a specific node P_0 of the hypercube and then scattered from there using the standard data scatter algorithms [8]. The time required to complete the data distribution by this *data scattering* method is given by

$$T_2 = \beta_h + Mp\tau_h + d\beta + M(p-1)\tau \qquad (7)$$

The first two terms of the equation represent the time for host to node transfer of N elements while the remaining terms represent the time for scattering the data in the hypercube. This is a good method because during scatter process, the available links of the hypercube are used fairly efficiently.

However, in the *data scattering* method described in [8], the host is idle during the data scatter in hypercube. Prasad and Murthy have therefore suggested a strategy which calls selecting a suitable size subcube of dimension x [9]. The host downloads the data for this subcube into a single node from where it is scattered using standard scatter algorithms. Concurrent with the scattering however, the host sequentially sends data to the remaining nodes of the hypercube. This *sequential/scattering* strategy completes the required data transmission in time given by

$$T_3 = \min_x\{\beta_h + M2^x\tau_h + \max\{(p-2^x)\beta_h + M(p-2^x)\tau_h, x\beta + M(2^x-1)\tau\}\} \quad (8)$$

The first two terms of the equation represent the time to transfer elements from the host to the root of the designated size x subccube. The maximum gives the larger of the two times: scattering time and sequential host loading time. The minimum function indicates that one should choose x appropriately to minimize the cost T_3. One may notice that when $x = d$, the degree of the hypercube, one gets the cost of the *data scatter* procedure. Thus, at all times, $T_3 \leq T_2$.

There are two problems with the *sequential/scattering* strategy described above. Firstly, since x is an integer, the terms within the max function of (8) cannot be balanced very well resulting in idle nodes or host. Secondly, this method cannot exploit any data overlap because of the host loading individual data sets into many of the nodes.

Our strategy for host to hypercube node communication is based on partitioning the hypercube into progressively smaller degree $d-1, d-2, \ldots, x+1, x, x$ subcubes. We appropriately choose x, $(0 \leq x < d)$ to minimize the overall communication cost. The host sends data sequentially to the designated roots of each of these subcubes. Each root then scatters the data within the subcube using a small modification of the standard algorithm [1] as follows. At the i-th step, $i = 0, 1, \ldots, t-1$, each node of the degree t subcube whose label q ends in $t-i$ zeros transfers appropriate amount of data to node $q + 2^{t-i-1}$. For example, when $t = 3$, at 0th step, node 0 sends data to node 4; at 1st step, nodes 0 and 4 send data to 2 and 6 respectively; and at the 2nd step, nodes 0, 2, 4, and 6 send data to 1, 3, 5 and 7 respectively. Assuming that the data sets being sent to consecutively labeled nodes have an overlap, this scattering scheme allows us to exploit that overlap to the maximum extent. Note that because of the overlap, the amount of data communicated at step i (along each of the participating link) is $M + (2^{t-i-1} - 1)\Delta$.

The total data distribution time using this scheme is obtained by noting that when $\beta_h > \beta$, the last subcube will finish last. Thus the total time expended in this data propagation is the sum of the times spent by the host on all its sequential communications with the subcubes added with the time required to scatter the data in degree x hypercube. This gives the total complexity as

$$T_4 = \min_x\{(d-x+1)\beta_h + (M-\Delta)(d-x+1)\tau_h + \Delta 2^d\tau_h + x\beta + x(M-\Delta)\tau + (2^x-1)\Delta\tau\} \quad (9)$$

By comparing (9) with (6)-(8) one can see the merit of the *Diminishing Scattering* strategy. Note that this strategy requires host to communicate with at most d hypercube nodes, where d is the hypercube degree. This is a feature lacking earlier strategies [9]. A small number of host connections is desirable for realistic implementation.

When the communication parameters are related as in (1), expression for T_4 takes the form

$$T_4 = \min_x\{((d-x+1)\sigma + x)\beta + (M-\Delta)(d+1)\tau + (2^d + 2^x - 1)\Delta\tau\} \quad (10)$$

Fig.s 1 and 2 show a comparison of the computational times of the *Diminishing Scattering*(DS) strategy with those obtained from earlier algorithms. In Fig. 1, the number of data points sent to each processor, $M = 100$ and $\beta = 800\mu$ and in Fig. 2, $M = 500$ and $\beta = 6500\mu$. In both simulations, $\sigma = 1.5$ and $\tau = 8\mu$. The overlap (defined as overlap $= M - \Delta$) between data sets going to consecutive hypercube nodes is varied between 0 and $M-1$.

These figures show that the *Diminishing Scattering* strategy is substantially better than the earlier ones. It can exploit the data overlap to lower distribution time. Note that in digital filtering, where each output point is being evaluated by an independent processor, the data overlap is indeed $M-1$, allowing a substantially lower transport time through this new strategy.

IV. CONCLUSION

This paper describes a communication strategy to distribute data and program from host to hypercube

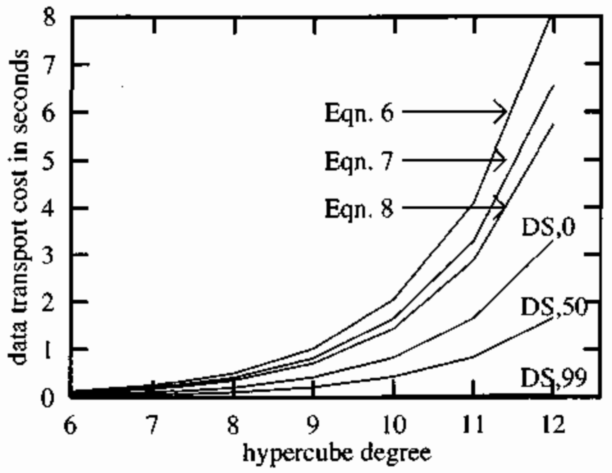

Fig. 1. Comparison of the *Diminishing Scattering* (DS) strategy with others when $M = 100$ and data overlap = 0, 50 and 99

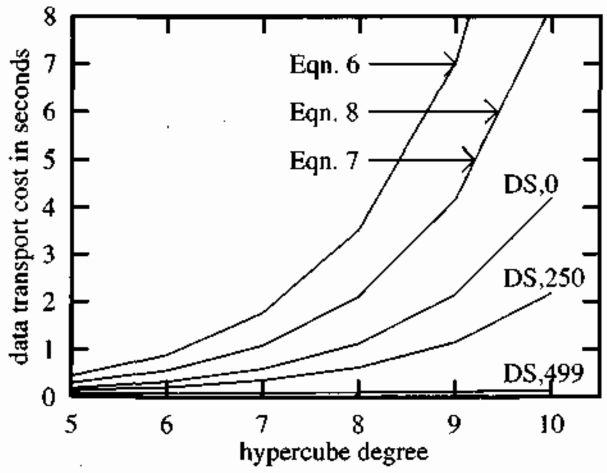

Fig. 2. Comparison of the *Diminishing Scattering* (DS) strategy with others when $M = 500$ and data overlap = 0, 250 and 499

nodes. The new strategy proposed is based on partitioning the hypercube into diminishing degree subcubes. The host sends data sequentially to the selected roots of these subcubes. Using a modified scattering technique described here, the roots then distribute the data into respective subcube nodes.

This strategy, called *Decremental Scattering*, uses subcubes of degree $d-1, d-2, \ldots, x+1, x, x$, where the integer x is chosen appropriately to minimize the total communication time. The choice of x takes into account the communication parameters in both host to node and node to node communication, the data set sizes as well as any possible overlap in the data sets.

Simulation results indicate that the *Decremental Scattering* strategy is highly superior to other strategies presented in the literature. This success of the strategy can be attributed to the fact that it maximizes the usage of all nodes and the host, as well as it meaningfully exploits the data set overlap. Thus, for widely varing communication conditions, this strategy consistently performs better. In applications such as digital filtering where the overlap is very large, this communication technique is highly recommended.

References

[1] S. Ranka and S. Sahni, *Hypercube Algorithms with applications to image processing and Pattern recognition.* New York, NY: Springer-Verlag, 1990.

[2] E. Jensen, "The honywell experimental distributed processor -an overview," *Computer*, vol. 11, pp. 28–38, Jan. 1978.

[3] T. Agarwala, "Communication, computation and computer architecture," in *1977 Int. Commun. Conf. Rec.*, June 1977.

[4] R. E. Blahut, *Fast Algorithms for Digital Signal Processing.* Reading, MA.: Addision-Wesley, 1985.

[5] M. D. Wagh and H. Ganesh, "A new algorithm for the discrete cosine transform of arbitrary number of points," *IEEE Trans. on Computers*, vol. C-29, pp. 269–277, 1980.

[6] K. A. Doshi and P. Varman, "Optimal graph algorithms on a fixed-size linear array," *IEEE Trans. on Computers*, vol. C-36, pp. 460–470, Apr. 1987.

[7] M. Sharma, *A new algorithm for calculation of the discrete Hartley transform.* M.S. dissertation, Lehigh University, 1989.

[8] Y. Saad and M. H. Schultz, "Data communication in hypercubes," *J. Parallel and Distributed Comput.*, no. 6, pp. 115–135, 1989.

[9] V. V. R. Prasad and C. S. R. Murthy, "Downloading node programs/data into hypercubes," *Parallel Computing*, no. 17, pp. 633–642, 1991.

[10] K. P. Bogart, *Discrete Mathematics.* Lexington, MA.: D. C. Heath and Company, 1988.

Proceedings of the International Conference on Parallel Processing (ICPP'93)
0-8493-8983-6/0-7 $20.00 © 1993 IEEE