An All-Reduce Operation in Star Networks Using All-to-All Broadcast Communication Pattern

Eunseuk Oh, Hongsik Choi, and David Primeaux

Department of Computer Science, School of Engineering,
Virginia Commonwealth University,
Richmond, VA 23284-3068, USA
{eoh, hchoi, dprimeau}@vcu.edu

Abstract. Most parallel computations require the exchange of data between processing elements. One of important basic communication operations is all-reduce, a variation of the reduction operation. This paper presents an all-reduce communication operation scheme using all-to-all broadcast communication pattern. All-to-all broadcast is the operation in which each processor sends its message to all other processors, and receives messages from all other processors in the system. In this paper, we develop an efficient all-reduce operation scheme in a star network topology with the single-port communication capability. Communication time is compared against known broadcasting schemes to verify the efficiency of the suggested scheme.

Keywords: all-reduce, all-to-all broadcast, distributed memory parallel computing systems, inter-processor communication, star network.

1 Introduction

Due to rapid progress in hardware technology, designing a distributed memory parallel computing system connecting autonomous microprocessors has become feasible. In such a system, high-performance microprocessors communicate by message passing and have no shared memory or global clock. Proper implementation of basic communication operations such as broadcast, reduction, and all-reduce on various parallel computing systems is key to the design of efficient parallel algorithms for distributed memory parallel systems.

One-to-all broadcast is an operation that disseminates information across processors in a multiprocessor system. It is not difficult to see that broadcasting stands as a foundation for many applications on parallel computing systems. To list a few applications that use broadcasting, we mention Fast Fourier Transformation (FFT), parallel matrix algorithms, parallel graph algorithms, and distributed algorithms. The all-reduce operation combines the arriving content in the input buffer of each processor using an associative operator (e.g. sum, maximum), and the result appears in the result buffer of all processors. All-reduce

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3514, pp. 419–426, 2005.

Table 1. Comparison of topological properties for parallel computer models of similar sizes: n-Cube (hypercube), MCT (mesh connected tree), D_n (de Bruijn network), and $HS_{n,m}$ (Cartesian product of hypercube and star)

Model	Size	Degree	Diameter	Model	Size	Degree	Diameter
S_5	120	4	6	S_6	720	5	7
7-Cube	128	7	7	10-Cube	1024	10	10
$MCT_4(3)$	81	12	16	$MCT_6(3)$	729	18	24
D_7	128	4	7	D_{10}	1024	4	10
$HS_{4,3}$	144	5	7	$HS_{5,3}$	720	6	9

is typically used for barrier synchronization on a distributed memory parallel computing system. Also, all-reduce is one of the most important MPI routines; a case study reveals that more than 40% of the execution time of MPI routines is spent in all-reduce or reduction operations [9].

In this paper, we study an all-reduce communication operation in star networks by using all-to-all broadcast communication pattern. The all-reduce operation is identical to performing an all-to-one reduction which is followed by a one-to-all broadcast of the result. Thus, we will compare the communication time of our scheme with all-reduce operation by using the best known broadcast scheme proposed in [10]. We first design a recursive all-to-all broadcast scheme that can be utilized to perform the all-reduce operation.

The star model has attracted considerable interest in the parallel processing research community $[1, 2, 3, 7, 8, 10, 11]$ due to its numerous desirable properties for building large parallel computer systems. Basic parameters such as size, degree, and diameter for the models whose size is similar to S_n are shown in Table 1.

Broadcasting schemes vary according to the communication capability of the channels or links. With single-port communication capability, every processor can simultaneously send and receive at most one message in one communication step. Also, a channel or link may be bidirectional or unidirectional. Cost measurements for the suggested scheme are provided under the single-port and bidirectional communication capability. Following the terminology used in [4], our scheme is "NODUP" in that there is no duplication of information on messages carried during the communication process.

The remainder of this paper is organized as follows. In Section 2 we introduce the communication model and the assumptions made about that model. In Section 3 we present an all-to-all broadcast scheme. In Section 4 we present an all-reduce operation based on our all-to-all broadcast scheme. In Section 5, we provide concluding remarks.

2 Model and Assumptions

The network model considered here is the star graph model. An n-star graph, S_n, consists of $n!$ nodes labeled with the $n!$ permutations on the symbols $\{1, 2, \ldots, n\}$.

There is a communication link between two processors p_i and p_j in S_n if and only if the permutation label of p_j can be obtained from the permutation label of p_i by exchanging the symbol in the first position in p_i with the symbol in some other position in p_i. If the label of p_j is obtained from the label of p_i by exchanging the first symbol of p_i with the symbol in kth position of p_i, then p_i and p_j are said to be connected along the communication link k.

The pattern of interconnected processors in the star network can be viewed recursively as follows. S_1 is a trivial network with one processor. Suppose that S_{n-1} is defined inductively, then S_n is composed of n graphs, S^i_{n-1}, $i = 1 \ldots n$, where each S^i_{n-1} is an isomorphic copy of S_{n-1} with symbols $\{1, \ldots, n\} - i$, and with symbol i appearing as the nth symbol in each processor in S^i_{n-1}. Connecting the nodes in different copies S^i_{n-1} and S^j_{n-1} is done with respect to the above definition. A node u in S^i_{n-1} is connected to a node v in S^j_{n-1}, $i \neq j$ when the label of v can be obtained from the label of u by exchanging the first symbol with the nth symbol.

The communication model for parallel computers varies depending on the communication hardware and the memory bus bandwidth. Most commercial systems support the single-port model. In the single-port communication model, a processor can send a message on only one of its communication links at a time. The sending and the receiving ports are not necessarily the same. The system model we consider is as follows; (1) The system is completely connected with synchronous communication. (2) A processor sends a message to a connected processor in one communication step. (3) Single-port communication and the communication links are bidirectional.

3 All-to-All Broadcast

All-to-all broadcast is performed recursively. After performing all-to-all broadcast in each S^i_{n-1}, $i = 1, \ldots, n$, all-to-all broadcast in S_n is performed. To avoid sending a message more than once to the same processor in the network, the private memory of each processor will be divided into two parts: the result buffer, and the outgoing message buffer. The partial sum will be stored in the result buffer.

All-to-All broadcast in S_4: The algorithm first calls itself recursively to perform broadcast in each of S^1_3, S^2_3, S^3_3, and S^4_3. The base of the recursion is when the network is an S_2. The algorithm performs broadcast in S_2 by a simple exchange of messages between the two processors. Then each processor in S_2 sends its message along communication link 3, and saves it in its result buffer. After that each processor sends its received message along communication link 2, and broadcast in S_3 is terminated. Now each S_3 in S_4 performs all-to-all broadcast in parallel fashion. At this point every processor computes its message by concatenating the message in its outgoing buffer with the message in its result buffer. Each processor stores the concatenated message in the result buffer, and writes a copy of the concatenated message over the current content of the outgoing buffer. We call this concatenated message the *meta message*. Then, every processor sends its meta message along communication link 4, and saves its message in its result buffer. Once a processor receives the message along communication link 4, it only needs to broadcast within each S_3. It starts this process by sending its message along communication link 3, then 2, meanwhile storing received messages in the result buffer.

All-to-All broadcast in S_n: In general, suppose inductively that all-to-all broadcast has been completed within each S^i_{n-1}, $i = 1 \ldots n$, and let us see how this can be extended to all-to-all broadcast in S_n. Since all-to-all broadcast has been completed in each S^i_{n-1}, each processor in S^i_{n-1} has received the messages from all other processors in S^i_{n-1}, and hence all processors in S^i_{n-1} share the same information. Denote the meta message in S^i_{n-1} by Δ^i_{n-1}. Let $\Delta_n = \bigcup_{i=1}^n \Delta^i_{n-1}$, then all-to-all broadcast in S_n is achieved once every processor in S_n holds Δ_n. All-to-all broadcast is performed as follows. In the first stage every processor p in S^i_{n-1}, $i = 1 \ldots n$, broadcasts its meta message Δ^i_{n-1} along communication link n and saves the meta message in its result buffer. After this stage, each S^i_{n-1} contains all messages in Δ_n among its processors. Thus, the only thing left to be done in the second stage is to propagate the information within each S^i_{n-1}. This step needs to be done with some care so that to avoid sending a message more than once to the same processor. Once p receives the meta message Δ^j_{n-1}, $j \neq i$, along communication link n, it propagates Δ^j_{n-1} across S^i_{n-1} by sending it along communication links $n-1, n-2, \ldots, 2$, respectively. Also, the received message is stored in the result buffer.

Theorem 1. *At the termination of all-to-all broadcast, each processor in S_n holds the meta message Δ_n.*

4 All-Reduce Operation

We perform all-reduce by using the communication pattern of all-to-all broadcast. Throughout this discussion, without loss of generality, we assume that addition is the associative operation performed in the all-reduce. An illustration of the all-reduce operation on S_3 is given in Figure 1. At each node, the final sum is obtained by adding the content in the result buffer and the outgoing buffer.

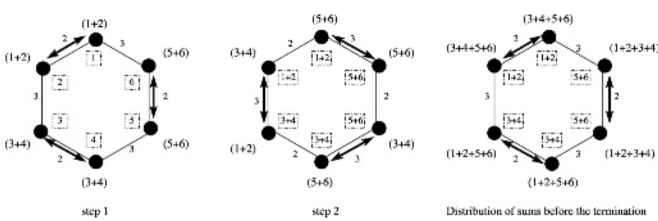

Fig. 1. The all-reduce operation on S_3. At each node, parentheses show the local sum in the outgoing buffer and the contents in the box is the local sum accumulated in the result buffer

All-Reduce

1. each S_{n-1}^i performs All-Reduce recursively;
   /* At this point, all processors in S_{n-1}^i have the sum of corresponding numbers to be added. */
2. each processor in S_{n-1}^i, $i = 1 \dots n$, sends the local sum along the communication link n and saves the local sum in its result buffer;
3. $d = n - 1$;
4. once each processor in S_{n-1}^i receives the local sum from a processor in S_{n-1}^j, where $j \neq i$, it performs the following
   while $d \geq 2$ **do**
      send the local sum to a neighbor of S_{n-1}^i along the communication link d;
      add the number received from the processor and the content of the result buffer in S_{n-1}^i;
      $d = d - 1$;
5. every processor adds the contents in its result buffer and the outgoing message buffer;

Fig. 2. All-reduce communication operation scheme

Assume that each number in the box, initially in the result buffer, is a number to be added.

An all-reduce operation follows the communication steps of all-to-all broadcast, but adds two numbers instead of concatenating messages. Thus, each message transferred in the all-reduce operation has only one word, where each word hold the partial sum of numbers. At the termination of the all-reduce operation, each node holds the sum $(1 + 2 + \dots + n!)$. Figure 3 shows all-reduce performed in S_4. The all-reduce scheme **All-Reduce** is shown in Figure 2.

Theorem 2. *At the termination of the all-reduce operation, each processor in S_n holds the sum $\sum_{i=1}^{n!} i$.*

Proof. The statement is vacuously true when $n = 1$. Let $n > 1$, and assume inductively that when **All-reduce** on S_{n-1} terminates, each processor in S_{n-1} contains the sum of corresponding numbers. When **All-reduce** is called on S_n, **All-reduce** calls itself recursively on each S_{n-1}^i, $i = 1 \dots n$. Since each S_{n-1}^i is a copy of S_{n-1}, by the inductive hypothesis, when each of these recursive calls terminates, each processor in S_{n-1}^i, $i = 1 \dots n$, holds the sum of corresponding numbers.

From the recursive definition of S_n given in Section 2, each S_{n-1}^i is linked to the other S_{n-1}^j, $j \neq i$ by exactly $(n-2)!$ links along communication link n. Thus, after the execution of step 2 of **All-reduce**, exactly $(n-2)!$ processors in

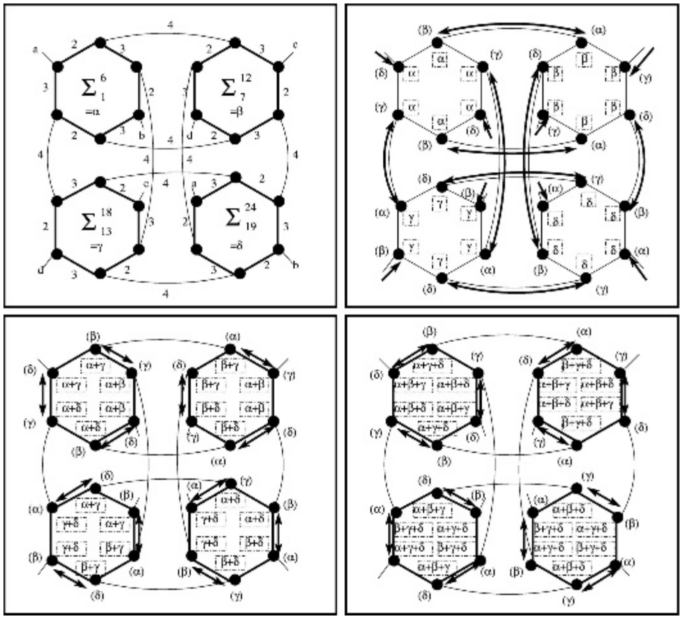

Fig. 3. The all-reduce operation on S_4

S^i_{n-1} holds the sum received from S^j_{n-1}. Also, each processor in S^i_{n-1} has the local sum stored in its result buffer. Now, each S^i_{n-1} holds the partial sums such that adding all partial sums results in $\sum_{i=1}^{n!} i$. Let p_1 be a processor in S^i_{n-1}, and let $\sum_j$ the partial sum received from S^j_{n-1}. From the above discussion, we know that there are exactly $(n-2)!$ processors in S^i_{n-1} that hold the partial sum $\sum_j$. If p_1 is one of these processors then we are done. Suppose this is not the case. Now from step 4 in **All-reduce**, we know that each of these processors will send $\sum_j$ along communication links $n-1, \ldots, 2$, respectively. We first claim that no processor p_1 in S^i_{n-1} is the neighbor of two distinct processors p_2 and p_3 that hold $\sum_j$ at the beginning of step 4. Suppose, for the sake of contradiction, that this were not the case. Let $p_1 = \sigma_1 \ldots \sigma_n$, and suppose that p_1 is the neighbor of p_2 and p_3 along communication links x and $y \in \{1, \ldots, n-1\}$, respectively. Notice first that $x \neq y$ since a processor is connected to exactly one processor along any given communication link. Since p_1 is connected to p_2 along communication link x, $p_2 = \sigma_x, \ldots, \sigma_1, \ldots \sigma_n$. Similarly $p_3 = \sigma_y, \ldots, \sigma_1, \ldots \sigma_n$. Since both p_2 and p_3 have $\sum_j$, both p_2 and p_3 are connected to S^j_{n-1} along communication link n. Now all processors in S^j_{n-1} have the same nth symbol. Since p_2 and p_3 are the neighbors of two processors in S^j_{n-1} along communication link n, it follows that both p_2 and p_3 have the same first symbol and $\sigma_x = \sigma_y$, a contradiction, since σ_x and σ_y are two symbols in the representation of processor p_1, and hence must be distinct.

Table 2. Comparison of all-reduce operations on communication time

Dimension of S_n	Size of S_n	Tseng [11]	Sheu [10]	Ours
3	6	12	6	3
4	24	27	12	6
5	120	48	18	10
6	720	75	26	15
7	5040	108	34	21
8	40320	127	42	28
9	362880	192	50	36
10	3628800	243	60	45

It follows from the above claim that the neighbors of the processors possessing the sum $\sum_j$ at the beginning of step 4 of **All-reduce** are distinct. Now each processor that holds $\sum_j$ broadcasts it to exactly $n-2$ neighbors along communication links $n-1,\ldots,2$. Since all these neighbors are distinct, the number of processors in S^i_{n-1} that receive $\sum_j$ from a processor in S^i_{n-1} at the beginning of step 4 is $(n-2)(n-2)!$. Thus, the total number of processors in S^i_{n-1} that hold $\sum_j$ at the end of **All-reduce** is $(n-2)!+(n-2)(n-2)! = (n-1)!$. It follows that all processors in S^i_{n-1} hold $\sum_j$ and in particular p_1. Since p_1 and j were arbitrarily chosen, every processor in S^i_{n-1} possesses every $\sum_j$ at the end of **All-reduce**, and hence holds the total sum $\sum_{j=1}^{n!} j$. □

Theorem 3. **All-reduce** *performs an all-reduce operation on* S_n *in time* $n(n-1)/2$.

Proof. The above theorem proves that when the algorithm **All-reduce** terminates, each processor holds the sum $\sum_{i=1}^{n!}$ in the system. Let $T(n)$ be the number of communication steps performed by **All-reduce** on S_n. Each S^i_{n-1} performs an all-reduce operation within itself and then sends a single message along communication link n, and then along communication links $n-1,\ldots,2$. Thus, the number of communication steps performed by each S^i_{n-1} is $T(n-1)+n-1$. Since all the S^i_{n-1}'s do this in parallel, the number of communication steps for S_n is the same as the number of communication steps performed by each S^i_{n-1}. Thus, the total number of communication steps performed by each S^i_{n-1}, and hence, by the whole network is given by the recurrence $T(n)=T(n-1)+n-1$. It gives $T(n)=n(n-1)/2$. □

5 Concluding Remarks

In this paper we presented an efficient all-reduce communication operation scheme by using the all-to-all broadcast communication pattern. Our scheme performs an all-reduce operation on an n-star network with the single-port capability in $n(n-1)/2$ time steps. If we use an all-to-one reduction followed by a one-to-all broadcast, an all-reduce can be performed in time $2\sum_{i=2}^{n}(\lceil log(i-1)\rceil+1)$ by the broadcast scheme proposed in [10] and in time $3(n-1)^2$ by the scheme proposed in [11]. In terms of the communication time shown in Table 2, our algorithm provides an improvement over the algorithms in [10, 11].

References

1. S. B. Akers, D. Harel, and B. Krishnamurthy, "The Star Graph: An Attractive Alternative to The n-cube," *Proc. Int'l. Conf. of Parallel Processing*, 1987, pp. 393-400.
2. S. B. Akers and B. Krishnamurthy, "The Fault Tolerance of Star Graphs," *Proc. 2nd Int'l. Conf. on Supercomputing*, 1987, pp. 270-276.
3. S. G. Akl, K. Qiu, and I. Stojmenovic, "Fundamental Algorithms for The Star and Pancake Interconnection Networks With Applications to Computational Geometry," *Networks*, 1993, vol.23, no. 4, pp. 215-225.
4. M. -S. Chen, P. S. Yu, and K. -L. Wu, "Optimal NODUP All-to-All Broadcast Schemes in Distributed Computing Systems," *IEEE Trans. Parallel and Distributed Systems*, 1994, pp. 1275-1284.
5. K. Efe and A. Fernandez, "Computational Properties of Mesh Connected Trees: Versatile Architecture for Parallel Computation," *Int'l Conference on Parallel Processing*, 1994, pp. 72-76.
6. Intel Corp. iPSC/2 User's Guide, Intel Corp., Mar, 1988.
7. I. M. Mkwawa and D. D. Kouvatsos, "An Optimal Neighbourhood Broadcasting Scheme for Star Interconnection Networks," *J. Interconnection Networks*, 2004, vol. 4, pp. 103-112.
8. E. Oh and J. Chen, "Strong Fault-Tolerance: Parallel Routing in Star Networks with Faults," *J. Interconnection Networks*, 2003, vol.4, pp. 113-126.
9. R. Rabenseifner and P. Adamidis, "Collective Reduction Operation on Cray X1 and Other Platforms," *Proc. the Cray User Group*, 2004
10. J. -P. Sheu, C. -T. Wu, and T. -S. Chen, "An Optimal Broadcasting Algorithm Without Message Redundancy in Star Graphs," *IEEE Trans. Parallel and Distributed Systems* 1995, vol.6, no. 6, pp. 653-658.
11. Y. -C. Tseng and J. -P. Sheu, "Toward Optimal Broadcast in A Star Graph Using Multiple Spanning Trees," *IEEE Trans. Computers* 1997, vol. 46, pp. 593-599.