社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ARC－177 (32)
2008－HPC－114 (32)
2008／3／6

Pipelined Round-Robin Broadcast Algorithm in Homogeneous Clusters of SMP

Shan Axida † Ta Quoc Viet ‡ Tsutomu Yoshinaga †

† University of Electro-Communications
Email:axida@sowa.is.uec.ac.jp, yosinaga@is.uec.ac.jp
‡ GLOBAL CYBERSOFT Inc.
Email:Ta.Quocviet@jp.yokogawa.com

This study proposes a novel broadcast algorithm for large-sized data over symmetric multiprocessor (SMP) clusters. The algorithm is based on round-robin scheduling, and a pipelined data scattering pattern. It can salvage all available communication resources of systems at every point in time and is thereby capable of achieving approximately the theoretical limit of performance. This implies that for a large data size on a network with any even number of nodes, the broadcast execution time is approximately the time required for a node to send data to another node. We compare the performance of the algorithm with that of broadcast algorithms that are widely used in high-performance computing systems.

1. Introduction

Homogeneous symmetric multiprocessor (SMP) clusters of workstations are widely used to perform high-performance computing. For such clusters to be more effective, communication must be carried out as efficiently as possible.

Data broadcast is one of the most common collective communication operations. It is also the most essential task in parallel and distributed processing. In distributed matrix manipulations such as different algorithms for distributed matrix multiplication [1, 2, 3] or a distributed solution of linear equation systems [4], data broadcast over processor rows and columns occupies almost all of its communication time. Therefore, developing an effective broadcasting algorithm can clearly improve the overall performance.

The broadcast operation requires a message from the root node (sender) to reach all other nodes in the system at the end of the operation.

In this work, we define *execution time* as the duration between the time when the root starts sending and the time when the last machine receives the entire message. The effectiveness of broadcast algorithms depends on *execution time*. In our consideration, a large amount D of data is to be broadcast. If t is the time that a pair of nodes spends on sending and receiving D, this should be the theoretical limit of the broadcast operation. By splitting the data into several fragments and then distributing to the destinations them by a cyclic schema, the root can complete its task in that limited time t. At the same time, the destination nodes exchange the fragments of data that they have already received from the root by round-robin scheduling; they can also finish their tasks at almost the same point in time as the root. The aim is to ensure that all nodes are busy at every point in time and make use of all existing communication links during the entire process. As a result, our algorithm can reach the theoretical limit of execution time t.

The rest of the paper is organized as follows. Section 2 briefly introduces related works. Section 3 defines the network model we analyze for our algorithm. Section 4 introduces the round-robin scheduling algorithm and its usage in global data exchange. Section 5 explains the pipelined round-robin broadcast algorithm in detail. Section 6 presents our experimental results. Finally, section 7 provides with our conclusions and future work.

2. Related Work

We first consider the broadcast algorithm among *Nnodes* nodes. The root sends large data of size D to all other *Nnodes-1* nodes.

One of the simplest broadcast algorithms is based on a linear tree topology. In this method, node 0 sends data to node 1, node 1 sends data to node 2, and so on. Since the send and receive operations repeat *Nnodes 1* times and each time the required time is t, the total cost of this algorithm, *execution time*, is *(Nnodes 1)* $\times t$, which is higher than that of other proposed algorithms [14]. However, each node (except for the root and last node) spends only

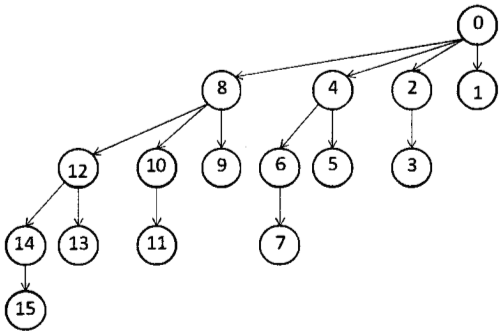

Figure 1. Binomial tree of order 4.

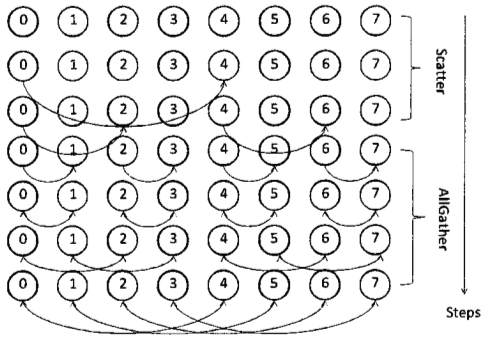

Figure 2. van de Geijn Algorithm.

$2 \times t$ for communication. By effective communication computation scheduling, it can perform computation tasks while others are in communication phases. By this way, in spite of an under-optimistic performance, linear tree broadcast algorithm is still acceptable in these cases. Moreover, the linear tree broadcast algorithm can be easily improved by the pipeline approach.

Another simple and well-known broadcast algorithm is based on binomial tree topology [11]. Figure 1 shows the structure of a binomial tree of order 4 as an example. In the first step, the *root* sends the complete data to node *Nnodes/2*. Next, the root and node *Nnodes/2* send the data to node *Nnodes/4* and node $(3/4) \times Nnodes$, respectively. Then, the algorithm is continued recursively. The total cost of the binomial tree broadcast algorithm is $log_2(Nnodes) \times t$, which depends on the height of the tree.

There is one more interesting and well-known broadcast algorithm known as the van de Geijn algorithm [11]. This algorithm consists of two phases (Figure 2).

(1) Scatter phase: The message is divided and scattered among all nodes by using a binomial tree.

(2) Allgather phase: The divided messages are collected by the recursive doubling technique. Then, the entire message becomes available in every node.

The total cost of the van de Geijn algorithm is $2\log_{Nnodes} \cdot s + 2(Nnodes-1)/Nnodes \cdot D/b$. The cost of the scatter phase is $(s+D/2b) + (s+D/4b) + (s+D/4b) + (s+D/8b) + \cdots = \log_{Nnodes} \cdot s + (Nnodes-1)/Nnodes \cdot D/b$. The cost of the Allgather phase is the same as that of the scatter phase.

Table 1. Total costs of broadcast algorithms

Algorithms	The total cost
Linear tree	$(P-1) \cdot (s+D/b)$
Linear tree (pipelined)	$(P-1) \cdot s+D/b$
Binomial tree	$\text{Log}_p \cdot (s+D/b)$
Van de Geijn	$2\log_p \cdot s+2(P-1)/P \cdot D/b$
RoundRobin (pipelined)	$(p-1) \cdot s+D/b$

Table 1 shows the comparison of total cost of various broadcast algorithms. P is the number of nodes; s is the communication delay; b is the network bandwidth and D is the data size.

A large number of broadcast algorithms have been proposed for different network topologies and platforms [7, 8, 9, 10, 11]. Unfortunately, they are not applicable for our network model.

3. Network Model

Here we list the definitions of our network model.

1. The interconnection network is static, consisting of point-to-point communication links among nodes.

2. The network is homogeneous and fully connected. All the nodes can be treated equally in terms of the local performance, and the communication rates between any pair of nodes are equal. Furthermore, the communication rate between a pair of nodes does not depend on the number of currently active communications of other pairs of nodes. In other words, the backbone bandwidth is sufficiently wide for all the nodes of the network to communicate at their individual full rates.

3. A node can send data to and/or receive data

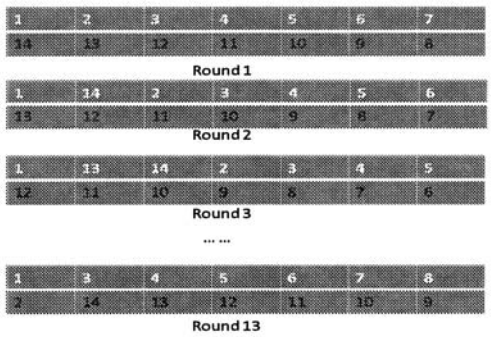

Figure 3. Rotating algorithm for round-robin scheduling with 14 participants

from only another single node at a time. This assumption is termed one-port communication model.

4. The network is full-duplex. The time required for two nodes to exchange messages of an equal length is equal to the time that they require while one of them sends and the other receives. In other words, a node can send and receive data to/from another node with the same rate as that of sending or receiving only. A majority of the modern switches mostly satisfy this assumption.

5. The time spent to communicate (send and receive, or exchange) a message between two nodes depends on and is in proportion to the message size only. By this assumption, we omit the preparation time and latency, which in some cases occupy a noticeable percentage of the overall communication time. However, by employing sufficiently large messages during communication, we can reduce this percentage such that it can be omitted.

4. Round-Robin Scheduling

4.1 Round-Robin Scheduling Algorithm

Our proposal is based on round-robin scheduling, which is widely employed in sport tournaments. In a round-robin tournament, each participant plays every other one and only one match. With *Nnodes* participants, in each round, there are *Nnodes/2* matches that can be played simultaneously. The number of rounds is *Nnodes-1*. If *Nnodes* is odd, we should add a virtual participant to the existing participants. In this case, there will be *Nnodes* rounds with *(Nnodes-1)/2* matches each. In each round, there is one competitor that does nothing if its partner by the schedule is the virtual participant.

In our network model, a participant corresponds to a computation node; a match between two participants is the process of sending and receiving data between the corresponding nodes. All of these pairs can communicate at the same time by sending and /or receiving data by employing their own communication links. The amount of data that each pair of nodes communicates is set to be equal, such that these communication tasks require the same period of time.

In our experiment, we apply the rotating algorithm with 14 participants shown in Figure 3. In each round, the two numbers in a column are the indices of the two players of a match. For example, in round 1, player 1 plays player 14, 2 plays 13, 3 plays 12, and so forth. In the next round, we fix the competitor number 1 and rotate the others clockwise and so on. We continue this action for *Nnodes-1* rounds.

4.2 Data Exchange

Suppose that the nodes of the network initially store different pieces of same size of the complete data. We term the process in which all the nodes mutually broadcast their own pieces of data as global data exchange. The round-robin scheduling can be applied directly in this case. We consider an even value of *Nnodes*. In each round, the pairs of nodes exchange data simultaneously. After *Nnodes-1* rounds, all the nodes possess the whole data. If the size of the complete data is *D*, the size of each part is *D/Nnodes*. Because the time spent on sending and receiving *D* is *t*, the time required for each round is *t/Nnodes*. In addition, the number of rounds is *Nnodes-1*. As a result, the time required for the entire data exchange operation is $((Nnodes\ 1)/Nnodes) \times t$. This value is also the theoretical limit of the execution time. In fact, it is the time required by a node needs to receive its missing data. The corresponding value for an odd number of nodes can be found similarly. The simplicity of the algorithm is due to the symmetric feature of the operation. Each node stores the same amount of data and plays the same role. However, the broadcast operation is not as simple. It comprises only a single data source for delivering

```
for (i=0;i<members-1;i++){
 if my partner is the root then receive my
message in segment (K+1);
 if my partner is the virtual node then do
nothing;
 if my partner is another destination then
exchange message of segment K;}
```

Figure 4. Pseudo code for loop iteration for destination nodes.

```
for (i=0;i<members-1;i++){
  if my partner is the virtual node then do
nothing;
  if my partner is a destination then send
message of segment K+1 to him;}
```

Figure 5. Pseudo code for loop iteration for the root

data to all the remaining nodes. In order to reach the theoretical limit, we improve the algorithm by adding a pipelined scattering phase. The next section discusses the algorithm in further detail.

5. Pipelined Round-Robin Broadcast

5.1 Terminologies

In this section, we explain terminologies that are used in succeeding sections.

1. *Message*. Within this work, a message is an atomic unit of data during the communication. A message should be sufficiently large so that its communication costs are in proportion to its size.

2. *Segment*. A set of *Nnodes-1* messages forms a segment. It is the object of the data exchange operation in an iteration of our algorithm. Segments are indexed from 0 to *Nsegs-1*, and the number of segments *Nsegs* can be found by *Nsegs=datasize/fragmentsize*.

5.2 Proposed Algorithm

1. Initial Phase

In the initial phase, the root scatters segment 0 (containing *Nnodes -1* messages) into the remaining *Nnodes -1* nodes. Due to the small size of a segment, we can ignore the costs of this phase.

2. Main Loop

The main loop contains *Nsegs-1* iterations. In an iteration, the nodes participate in a round-robin

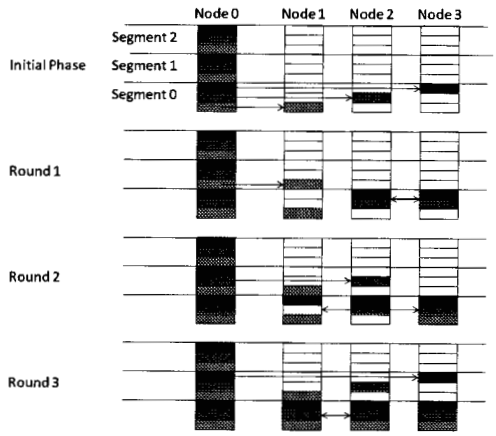

Figure 6. Initial phase and iteration 0.

tournament. In the K^{th} iteration, the root scatters the messages of segment *K+1* to the destinations, while the destinations exchange the data of segment *K*. After the completion of the iteration, all the destinations should receive the entire segment *K* and a message of segment *K+1*. The pseudo codes for the K^{th} iteration for the destination and root node are shown in Figure 4 and 5, respectively. The number of players (*members*) of the tournament can be found by adding a virtual player to *Nnodes* if *Nnodes* is odd:

members=(Nnodes%2==0)? Nnodes : (Nnodes +1)

3. Final Phase

In the final phase, since the root has already scattered all of its data, it does nothing. The destinations exchange the messages of the last segment. Similar to the initial phase, we can ignore the costs of the final phase. As a result, the performance of the algorithm can be evaluated by determining the costs of just the main loop.

Figure 6 illustrates the initial phase and the first iteration of the main loop with *Nnodes* = 4 and node 0 as the data source. In this figure, a cell represents a memory space reserved for a message, and the first three data segments are shown. The colored and blank cells represent memory spaces with and without corresponding data, respectively. Initially, the three destination nodes have no data, and their cells are blank. The arrow denotes a sending and receiving operation. The bidirectional arrow denotes a mutual data exchange operation. Since *Nnodes* = 4, there are 3 rounds in each

same time (we term this type of communication as data transfer). However, we can send and receive a message at the same time between two nodes, and this feature is suitable for the pipelined round-robin broadcast algorithm (we term this type of communication as data exchange). As a result, the execution time of data transfer is approximately twice that of data exchange in communication experiments on our cluster.

Our network model cannot achieve the ideal performance discussed earlier. The time it requires to broadcast over a large number of nodes is approximately twice that in the case of two nodes. This is incident which is explained by the network model. Not all the assumptions for the fully connected network are satisfied by our experimental cluster. The links between nodes are switched and not static. When the number of nodes is large, the costs of network switching may become noticeable. Besides, the number of working links may also affect the communication rate. Due to these reasons, we are unable to reach the time limit in the actual experiment.

7. Conclusion and future work

In an ideal homogeneous, fully connected, and full-duplex network, the pipelined round-robin broadcast algorithm can reach the theoretical minimum limit of the execution time with a large data size. With any large number of nodes of a cluster, the time it spends for the entire broadcast operation is exactly equal to the time required by a pair of needs to transfer data from one node to another. The round-robin scheduling algorithm allows the nodes to communicate simultaneously and thereby utilize all the available links at any point in time. The look-ahead scattering technique makes data available for exchanging between the destinations nodes all the time.

There is scope for improvement in the performance of the actual broadcast operation. For example, we may run the experiment using different message sizes and obtain the best value for it. Another option is to improve the broadcast schedule in order to minimize the number of switching operations, thereby reducing the preparation overheads. We intend to address above mentioned issues in our research in the near future.

Acknowledgments

This research is supported in part by the Grants-in-Aid for Scientific Research of Japan Society for Promotion of Science (JSPS) No. 19500040.

References

[1] R. A. van de Geijn and J. Watts. SUMMA: Scalable Universal Matrix Multiplication Algorithm. *LAPACK Working Note 99*, technical report, University of Tennessee, April 1995.

[2] J. Choi. A Fast Scalable Universal Multiplication Algorithm on Distributed-Memory Concurrent Computers. *Proc. IPPS'97*, no.1063-7133, pp. 310-314, April 1997.

[3] J. Choi J. J. Dongarra, and D. W. Walker. PUMMA: Parallel Universal Matrix Multiplication Algorithms on Distributed Memory Concurrent Computers. Concurrency: Practice and Experience, pp. 543-570, August 1993.

[4] T. Q. Viet, T. Yoshinaga, B. A. Abderazek, and M. Sowa. Construction of Hybrid MPI-OpenMP Solutions for SMP Clusters. IPSJ Transactions on Advanced Computing Systems, no. SIG_3(ACS_8), pp. 25-37, January 2005.

[5] MPICH Team. MPICH, a Portable MPI Implementation. http://www-unix.mcs.anl.gov/mpi/mpich1

[6] W. Gropp, E. Lusk, and A. Skjellum. Using MPI Portable Parallel Programming with the Message-Passing Interface, The MIT Press Cambridge, Massachusetts London, England, 1994.

[7] H. Ko, S. Latifi, and S. Srimani. Near-Optimal Broadcast on All-Port Wormhole-Routed Hypercubes Using Error Correcting Codes. IEEE TPDS, no. ISSN: 1045-9219, pp. 247-260, March 2000.

[8] O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. Pipelined Broadcast on Heterogeneous Platforms. 19th international parallel & distributed processing symposium, Denver, Colorado, April 2005.

[9] O. Beaumont, L. Marchal, and Y. Robert. Broadcast Tree for Heterogeneous Platforms. IEEE Transactions on Parallel and Distributed Systems, pp. 300-313, April 2005.

[10] T. Chiba, T. Endo, and S. Matsuoka. High-Performance MPI Broadcast Algorithm for Grid Environments Utilizing Multi-lane NICs. the Seventh IEEE International Symposium on Cluster Computing and the Grid, no. ISBN: 0-7695-2833-3, pp. 487-494, May 2007.

[11] P. Patarasuk, A. Faraj, and X. Yuan, Pipelined Broadcast on Ethernet Switched Cluster. the Twentieth IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rhodes Island, Greece, April 2006.

[12] R. A. DeVenezia, Round Robin Tournament Scheduling. http://www.devenezia.com/downloads/round-robin/index.html.

[13] C. E. Leiserson. Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. IEEE Transactions on Computers, no. ISSN:0018-9340, pp. 892-901, October 1985.

[14] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary. HPL- A Portable Implementation of the High-Performance Linpack Benchmark for Distributed Memory Computers. http://www.netlib.org/benchmark/hpl/

Table 2. Theoretical broadcast time with Some odd values of *Nnodes*

Nnodes	Execution time ($\times t$)
3	1.5
5	1.25
7	1.167

iteration. Node 0 in turn sends a message to the destination. In a round, the destinations receive a message from the source or exchange an existing message with another destination depending on whether or not its partner is the root. As shown in the figure, after the first iteration, every destination receives the entire data segment 0 and a piece of data segment 1. Continuing with the next iterations, it can fill the complete data without idling.

5.3 Performance

e can determine the overall broadcast time by determining *execution time* of any node. Let us consider the root with different values of *Nnodes*. Its *execution time* includes active and idle times. Active time is the period of time when it actually sends data to others. Idle time is the period of time when it does nothing during the matches in which its partner is the virtual player (if any).

1. *Nnodes == 2*

Obviously, this is the simple case of sending and receiving data. The execution time is t. No broadcast algorithm is required.

2. *(Nnodes > 2) && (Nnodes%2 ==0)*

There is no virtual player and the root always has a partner to send its data. As a result, the execution time is t again, the theoretical limit.

3. *(Nnodes > 2) && (Nnodes%2==1)*

The active time of the root is t. In an iteration, the root is idle in one match and active in *Nnodes 1* matches. Therefore, its idle time can be found by *t/(Nnodes 1)* and the execution time can be found by :

$t + (t/(Nnodes\ 1)) = (Nnodes/(Nnodes\ 1)) \times t.$

It is interesting to note that the execution time reduces with increasing number of nodes. Table 2 lists the broadcast time corresponding to some odd values of *Nnodes*.

6. Experimental Results

We implement the algorithm on a PC cluster

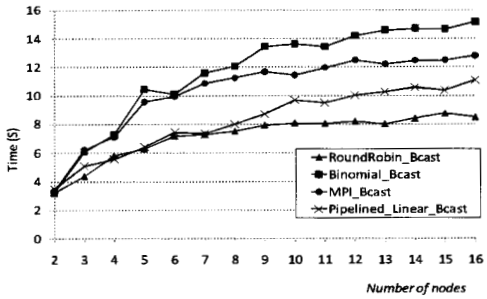

Figure 7. Broadcast execution time with increasing number of nodes

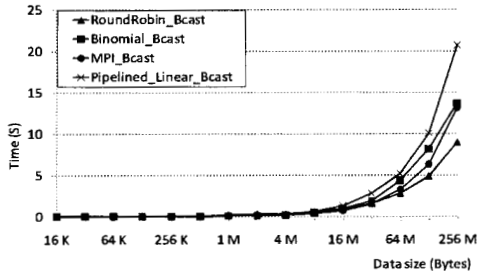

Figure 8. Broadcast execution time with increasing data size

connected via a gigabit switch by MPI programming [13]. Our cluster includes 16 computation nodes with Intel Xeon 2.8 GHz processors. We use Fedora Core and MPICH 1.2.7 in our experiment. As compiler, we use ICC (Intel C compiler 9.0). The data size is 30 million double precision numbers, which is equivalent to 1.92 Gbits. The message size is 500 Kbits. Figure 7 and 8 illustrate the execution time of the broadcast algorithms we mentioned in this paper. The performance of the linear broadcast that we omit from the figure is the worst one.

The pipelined round-robin broadcast shows the best performance in both the cases the constant number of nodes but increasing size of data and the case of constant size of data but increasing number of nodes.

The pipelined linear broadcast can reach the theoretical limit of the execution time with a large amount of data [11]. Unfortunately, in our network model (assumption 3), it is not possible to realize a node that receives a message from the previous node and sends a message to the next node at the