Available online at www.sciencedirect.com
ScienceDirect
J. Parallel Distrib. Comput. 68 (2008) 809–824
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

Techniques for pipelined broadcast on ethernet switched clusters☆

Pitch Patarasuk[a], Xin Yuan[a,*], Ahmad Faraj[b]

[a] *Department of Computer Science, Florida State University, Tallahassee, FL 32306, United States*
[b] *Blue Gene Software Development, IBM Corporation, Rochester, MN 55901, United States*

Received 24 February 2007; received in revised form 31 October 2007; accepted 27 November 2007
Available online 25 February 2008

Abstract

By splitting a large broadcast message into segments and broadcasting the segments in a pipelined fashion, pipelined broadcast can achieve high performance in many systems. In this paper, we investigate techniques for efficient pipelined broadcast on clusters connected by multiple Ethernet switches. Specifically, we develop algorithms for computing various contention-free broadcast trees that are suitable for pipelined broadcast on Ethernet switched clusters, extend the parametrized LogP model for predicting appropriate segment sizes for pipelined broadcast, show that the segment sizes computed based on the model yield high performance, and evaluate various pipelined broadcast schemes through experimentation on Ethernet switched clusters with various topologies. The results demonstrate that our techniques are practical and efficient for contemporary fast Ethernet and Giga-bit Ethernet clusters.

Keywords: Broadcast; Ethernet; Collective communication

1. Introduction

Switched Ethernet is the most widely used local-area-network (LAN) technology. Many Ethernet switched clusters of workstations are used for high performance computing. For such clusters to be effective, communications must be carried out as efficiently as possible.

Broadcast is one of the most common collective communication operations. In this operation, the *root* process (the sender) sends a message to all other processes in the system. The Message Passing Interface routine that realizes this operation is *MPI_Bcast* [18]. Broadcast algorithms are classified as either *atomic broadcast* algorithms or *pipelined broadcast* algorithms. Atomic broadcast algorithms distribute the broadcast message as a whole through the network. Such algorithms apply to the cases when there is only one broadcast operation and the broadcast message cannot be split. When there are multiple broadcast operations or when the broadcast message can be split into message segments, a pipelined broadcast algorithm, which distributes message segments in a pipelined fashion, can usually achieve higher performance than atomic broadcast algorithms.

Two main issues must be addressed in order for pipelined broadcast to achieve high performance in a cluster. First, a broadcast tree that allows efficient pipelined broadcast must be determined. In pipelined broadcast, communications on different branches of the broadcast tree can be active at the same time. To maximize the performance, communications that can potentially happen simultaneously should not share the same physical channel and cause network contention. Hence, the broadcast trees for pipelined broadcast should be contention-free. Second, appropriate segment sizes must be selected since the segment sizes directly affect the total broadcast time. A small segment size may introduce excessive communication start-up overheads while a large segment size may decrease pipeline efficiency.

We investigate efficient pipelined broadcast for realizing *MPI_Bcast* with large message sizes on clusters connected by multiple Ethernet switches. In this case, broadcasting a large message is carried out by a sequence of pipelined broadcasts with smaller message segments. We develop algorithms for computing various contention-free broadcast trees that are suitable for pipelined broadcast on Ethernet switched clusters,

☆ A preliminary version of this paper is published in IEEE IPDPS 2006.
* Corresponding author.
E-mail addresses: patarasu@cs.fsu.edu (P. Patarasuk), xyuan@cs.fsu.edu (X. Yuan), faraja@us.ibm.com (A. Faraj).

0743-7315/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2007.11.003

and extend the parametrized LogP model [14] for predicting appropriate segment sizes. We evaluate the proposed techniques on fast Ethernet (100 Mbps) and Giga-bit Ethernet (1000 Mbps) clusters. The results show that the proposed techniques are practical and efficient on such clusters. We will refer to fast Ethernet as 100 Mbps Ethernet and Giga-bit Ethernet as 1000 Mbps Ethernet in the rest of this paper. The main conclusions are as follows:

- Pipelined broadcast is more effective than other broadcast schemes, such as those used in MPICH 2-1.0.1 [19] and LAM/MPI 7.1.1 [15] on both 100 Mbps and 1000 Mbps Ethernet switched clusters in many situations.
- Contention-free broadcast trees are essential for pipelined broadcast to achieve high performance on clusters with multiple switches. Pipelined broadcast using topology-unaware broadcast trees may result in poor performance in such an environment.
- Pipelined broadcast is relatively insensitive to the segment size in that the range of segment sizes that can yield high performance for a given operation is large. Our extended parameterized LogP model is sufficiently accurate for finding the appropriate segment size for a given pipelined broadcast on a platform.

The rest of the paper is organized as follows. Related work is discussed in Section 2. The network model and commonly used broadcast algorithms are described in Section 3. Section 4 details the algorithms for computing various contention-free broadcast trees on Ethernet switched clusters and presents the extended parameterized LogP model. Section 5 reports the results of our experimental evaluation. Finally, Section 6 concludes the paper.

2. Related work

The broadcast operation has been extensively studied and a very large number of broadcast algorithms have been proposed. More closely related to this work are various pipelined broadcast schemes. Various binomial tree based pipelined broadcast algorithms have been developed [11,13,23–25]. In these schemes, each node sends successive segments to its children in a round-robin fashion. One example is the k-binomial tree algorithm [13], which is shown to have a better performance than traditional binomial trees. Although these schemes can achieve theoretical optimal or nearly optimal performance, the shapes of the broadcast trees are fixed. Such trees require high network connectivity to be contention-free. For networks with lower connectivity, such as Ethernet that has a tree topology, such broadcast trees do not have a contention-free embedding and thus, the techniques in [11,13,23–25] cannot be extended to clusters connected by multiple Ethernet switches.

Pipelined broadcast has also been investigated in other environments. In [1,2], heuristics for pipelined communication on heterogeneous clusters were devised. These heuristics focus on the heterogeneity of links and nodes, but not the network contention issue. In [27], a pipelined broadcast technique is proposed for the mesh topology. The effectiveness of pipelined broadcast in cluster environments was demonstrated in [10,21,26]. It was shown that pipelined broadcast using topology-unaware trees can be very efficient for clusters connected by a single switch. In [20], a scheme was proposed where the broadcast tree changes smoothly from a binary tree to a linear tree as the message size increases.

In this paper, we do not propose new pipelined broadcast schemes. Instead, we develop practical techniques to facilitate the deployment of pipelined broadcast on clusters connected by multiple Ethernet switches. Similar to other architecture specific collective communication algorithms [7,9,16], the techniques developed in this paper can be used in advanced communication libraries [6,8,12,28]. Our research extends the work in [10,21,26] by considering multiple switches. As shown in the performance study, pipelined broadcast using topology-unaware trees in such an environment may yield extremely poor performance. To the best of our knowledge, methods for building fully contention-free trees for pipelined broadcast over a physical tree topology have not been developed. Moreover, although various models that can be used to determine appropriate segment sizes for pipelined broadcast have been proposed [3–5,14,20,24,25], these schemes cannot directly apply to Ethernet switched clusters either because the model assumptions do not hold or because the model parameters cannot be measured with sufficient accuracy. We extend the parameterized LogP model in [14] for determining the appropriate segment sizes in pipelined broadcasts. Notice that we could have extended the $\mathrm{Log}_n P$ and $\mathrm{Log}_3 P$ models [4] to have more powerful models (e.g. having the ability to deal with non-contiguous data types). However, these models mainly focus on dealing with non-contiguous memory accesses, which is not the emphasis in our paper. As such, we extended the simpler parameterized LogP model that is sufficient for our purpose.

The pipelined broadcast approach can only be efficient for broadcasting large messages. For small messages, other broadcast algorithms are needed. There are techniques to develop adaptive MPI routines that use different algorithms according to the message sizes [6,19]. These adaptive techniques allow our algorithms and the complementary algorithms for broadcasting small messages to co-exist in one MPI routine.

3. Network model

We consider Ethernet switched clusters where each workstation is equipped with one Ethernet port and each Ethernet link operates in the duplex mode that supports simultaneous communications in both directions with full bandwidth. Communications in such a system follow the 1-port model [2], that is, a machine can send and receive one message at any one time. The switches may be connected in an arbitrary way. However, a spanning tree algorithm is used to determine forwarding paths that follow a tree structure [22]. As a result, the physical topology of the network is always a **tree** with switches being the internal nodes and machines being leaves.