Network Performance Model for TCP/IP Based Cluster Computing

Akihiro Nomura #1, Hiroya Matsuba *2, Yutaka Ishikawa #*3

#*Dept. of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo*
7-3-1 Hongo, Bunkyo-ku, Tokyo, JAPAN
1nomura@il.is.s.u-tokyo.ac.jp
3ishikawa@is.s.u-tokyo.ac.jp

**Information Technology Center, The University of Tokyo*
2-11-16 Yayoi, Bunkyo-ku, Tokyo, JAPAN
2matsuba@cc.u-tokyo.ac.jp

Abstract— **A new communication model, called the PlogPT model, is proposed to predict communication performance in a commodity cluster where computing nodes communicate using TCP/IP. This model extends the PlogP model in order to consider the variation of bandwidth brought about by bottleneck links in network switches and the delay of packet retransmission by TCP/IP handling. Network switches are modeled as binary tree connections. To demonstrate the PlogPT model's modeling capability, the execution time of two all-to-all communication algorithms are estimated and compared with the actual execution time and that of the PlogP model. The PlogPT model predicts the execution time of those two algorithms more precisely than the existing model.**

I. Introduction

In parallel computing, improving the performance of computation is very important issue. Collective communication functions are frequently called and their execution time accounts certain amount of whole execution time in parallel applications. Predicting the execution time of these functions is important to improve their performance. In order to formalize the behavior of communication, many communication models [1]–[5] have been introduced.

Many cluster systems have been built using a commodity network, such as Ethernet, because this configuration achieves good cost-performance if the application does not require high network bandwidth. In such a commodity network environment, users often assume that the network is a full bisection bandwidth network if all nodes are connected by a single network switch. However, this assumption sometimes fails if the switch consists of several small switches and does not provide full bisection bandwidth for all ports of the switch. In fact, large switches often consist of small internal switches and do not provide full bisection bandwidth [6]. It is difficult for the user to realize this fact because the information of the internal switch configuration is not often disclosed.

The communication performance varies if bottleneck links exist, because a packet loss resulting from congestion causes a delay of the packet delivery. Such behavior makes performance prediction hard. The conventional approaches assume two things. First, the communication path ensures at the hardware level that the messages are transmitted without loss. Next, each communication is unaffected by communications in other nodes. These assumptions are not true in a commodity network. Therefore, the estimation of communication time cannot be predicted accurately.

In this paper, we propose a new communication model, called PlogPT (PlogP based on Tree). The PlogPT model extends the PlogP [4] model to express the impact of bottleneck links and the delay caused by retransmission timeout (RTO), neither of which are considered in the PlogP model and other models [1]–[3], [7].

In PlogPT model, a binary tree model is introduced in order to express the bottleneck links in network switches. The bottleneck link is expressed by the network gap, representing the time of message transfer, based on the throughput, which varies with the communication pattern. The condition for a loss of the last packet in communication, which causes RTO (retransmission time out), is formalized.

By measuring not only unidirectional, but also bidirectional communication, the estimated parameters of this model get close to the actual value. Two types of communication measurements make it possible to estimate the communication time more precisely.

The execution time of two all-to-all communication algorithms will be estimated by the PlogPT model and PlogP. By comparison between the actual execution time and the estimations, the PlogPT model estimates the execution time more accurate than the PlogP estimation.

II. Background

In this section, some of traditional communication models are summarized, and issues arising from applying those models to a commodity network are revealed. Communication models related to the proposed model will be shown in Section V.

A. The logP model

The logP model [1] expresses the characteristics of communication between each node by the following four parameters, the latency of each message (l), the overhead a processor spends for each transmission and reception (o), the minimum interval gap between each message (g), and the number of

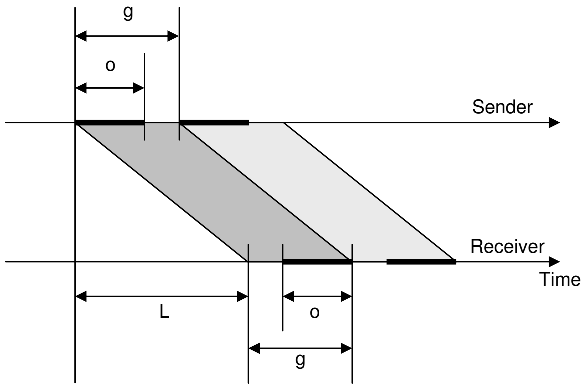


Fig. 1. Parameters in the logP model

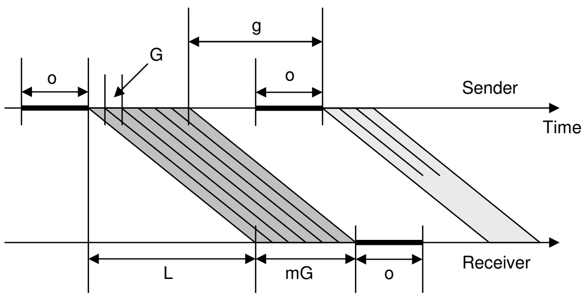


Fig. 2. Parameters in the logGP model

processors (P). Figure 1 illustrates the meaning of these parameters. In some variants, the parameter o is divided into o_s and o_r. o_s represents the overhead of sending a message, o_r represents that of receiving. This model expresses the characteristics of communications using only short messages. Therefore, this model is not suitable for communications whose message length varies or is too long.

B. The logGP model

The logGP model [2] extends the logP model to cover inaccuracy of prediction for long messages. In this model, the inter message gap g in the logP model is divided into a constant element and an element proportional to message size (m), and is represented as $g + mG$ form. Figure 2 shows the meaning of the logGP parameters. This model assumes that the inter message gap is linear to the message size. This assumption is true for large messages. In some networks, such as an IP network, the behavior of the message gap changes contingent on the message size reaching the media level MTU. Figure 3 demonstrates that the linearity of the inter message gap is broken in short messages in the experimental environment shown in Table I. Thus, this model is not able to express the behavior of communication well if the message size is small.

TABLE I
EVALUATION ENVIRONMENT

Number of nodes	32
CPU	AMD Opteron 2GHz Dual
Memory	3.5GB
OS	Linux 2.6.11 SMP
Network adapter	Broadcom Tigon 3 1Gbps
MPI Library	YAMPI 1.0 [8]
Ethernet Switch	BayStack5510-48T

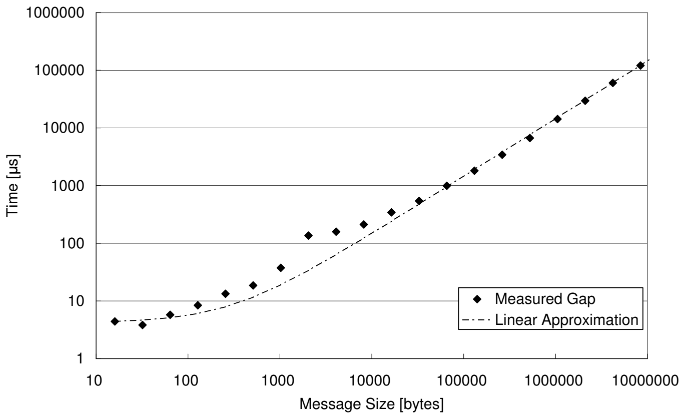


Fig. 3. The relationship between message size m and the inter message gap $g(m)$

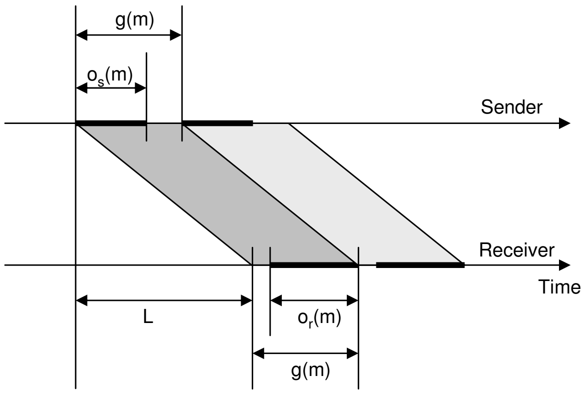


Fig. 4. Parameters in the PlogP model

C. PlogP model

The parameterized logP (PlogP) model [4] is another extension of the logP [1] model. In the PlogP model, the overhead of the sender and the receiver, o_s, o_r, and the gap g are parameterized by the message size m and written as $o_s(m)$, $o_r(m)$ and $g(m)$. The relationship between these parameters and the timing of communication are shown in Figure 4.

This model avoids the difficulty involved in describing the inter message gap in accordance with the message size. The gap depends on the minimum/maximum packet size of the physical network layer. It cannot be formulated simply by linear equations. Thus, the PlogP parameters are measured independently from the message size so that this model is able to express the correlation between gaps and message size for both large and small messages.

The measurement method used for the message gaps is important in the PlogP model. In the paper [9], a fast measuring method for a large number of parameters in the PlogP model

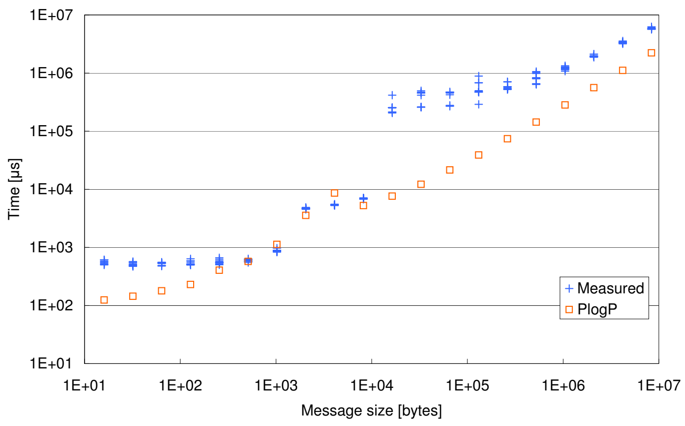


Fig. 5. Difference between the actual execution time and one estimated by the PlogP model

is introduced. Measurement tools for an MPI environment are currently available. In this method, first, the parameter $g(0)$ is estimated from the transmission frequency of zero sized messages. Then, the latency parameter L is given as follows, using the RTT (round trip time) of a zero sized message:

$$RTT = 2(L + g(0)) \tag{1}$$

Finally, the parameter $g(m)$ for each message size m is calculated from $g(0)$, L, and the round trip time of a size m message. $o_s(m)$ and $o_r(m)$ are measured at the same time.

Figure 5 shows the estimated execution time in the PlogP model and the actual execution time in an implementation of MPI_Alltoall communication using TCP/IP. The environment used in this experiment is shown in Table I.

The estimated time is much shorter than the actual time due to the following two reasons. One reason is that the MPI_Alltoall implementation uses bidirectional communication with bandwidth that is usually less than the double bandwidth of unidirectional communication. However, the gap parameter is measured using the unidirectional communication bandwidth in this method. Thus, the estimation of bidirectional communication bandwidth is larger than the actual bandwidth.

Another reason is that the PlogP model does not consider the retransmission timeout of TCP/IP. The difference between the estimated time and the measured time in a larger message is approximately an integral multiple of 200 ms. The 200 ms time is the minimum retransmission timeout of the TCP/IP implementation in Linux. This fact implies that the retransmission mechanism of TCP/IP when performed in some condition holds. This mechanism is shown in the next paragraph.

D. Retransmission Control

Unlike a dedicated high performance network such as Infiniband [10] or Myrinet [11], a commodity network, i.e., Ethernet, does not guarantee reliability in the data link layer, however the upper protocol layer, in the form of transmission protocol such as TCP/IP supports reliability. In TCP/IP, the receiver reports the completion of transmission by sending an acknowledge packet to the sender. If the acknowledge packet does not arrive at the sender within a certain time, called the retransmission timeout (RTO), the sender assumes that the data packet sent by the sender has been lost and retransmits the lost data packet again. The RTO is significantly longer than the round trip time.

The RTO may frequently happen in MPI communication [12]. If the last data packet of an MPI message is lost, the receiver can not detect the loss until a packet of another MPI message from the same sender is received. In this case, thus, the sender waits for an RTO to retransmit the lost packet.

E. Bottleneck Links in Network Switches

Inexpensive Ethernet switches with many ports have small internal sub-switches of the same size. Those sub-switches may not be fully connected. That is, the communication performance between sub-switches is lower than that inside of a sub-switch. For example, the article [6] says that some 24 or 48 port switches consist of six groups of ports and each group is connected by a 1Gbps internal network. That is, the bisection bandwidth across two groups is limited to 1Gbps. In the environment mentioned in Table I, the bisection bandwidth, shown in Table II as $b(h)$, differs depending on the division.

TABLE II
BISECTION BANDWIDTH AND LATENCY

h	**unidirectional**		**bidirectional**	
	$d(h)$	$b(h)/2^{h-1}$	$2d(h)$	$b(h)/2^{h-1}$
1	500 [μs]	936.4 [Mbps]	502 [μs]	580.2 [Mbps]
2	517 [μs]	935.6 [Mbps]	902 [μs]	580.1 [Mbps]
3	535 [μs]	936.1 [Mbps]	913 [μs]	576.0 [Mbps]
4	527 [μs]	934.1 [Mbps]	910 [μs]	579.2 [Mbps]
5	543 [μs]	625.0 [Mbps]	909 [μs]	485.0 [Mbps]

$b(h)$ denotes the bisection bandwidth of a group of 2^h nodes and $d(h)$ denotes the additional round trip time from the leftmost node to the rightmost node in the group

III. Design

A. Binary Tree Network Model

First, we propose a network topology model in which the network is modeled as a perfect binary tree, as shown in Figure 6. A perfect binary tree has $n = 2^D$ leaves at the same depth D from the root node. All computing nodes are located as the leaves of the tree. The edges of the tree represent communication links, and each edge has its own bandwidth. Each intermediate node represents a switch with two links.

In the following, we often focus on subtrees of the network tree, and the distance from leaves is more important than that from the root node for each edge. We introduce the concept of the height of edges and intermediate nodes for descriptive purposes. We define the height of node h as the length of the path from the leaves, which equals $D - d$, in which d is the depth of the node. In other words, the height h node is the root node of the height h subtree. We also define the height of the edge as the distance between the edge and its descendant leaves, including the edge itself, the same value as the edge's parent node's height. The edge at height h has 2^{h-1} descendant leaves, and its bandwidth should be 2^{h-1} times

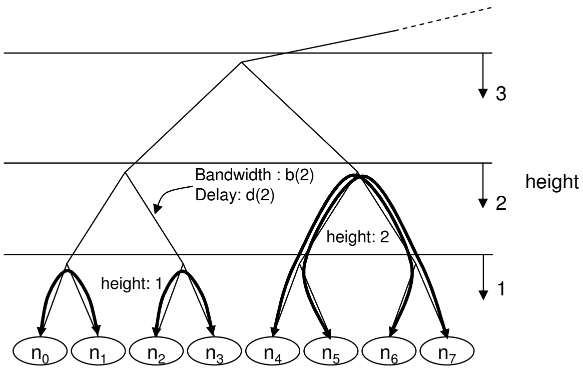


Fig. 6. Binary Tree Network Model

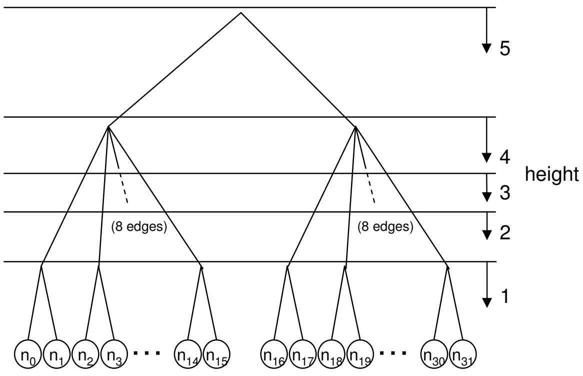


Fig. 7. Image of the Actual Tree Network in the Testing Environment

larger than the edge at height 1, if the network guarantees full bisection bandwidth. We define the height of communication as the maximum height of the edges which are involved in the communication.

In this model, the network is symmetric about each intermediate node, including the root node. All edges at the same height h have the same properties, such as bandwidth and delay. We introduce two parameters $b(h)$ and $d(h)$ for each height h. $b(h)$ denotes the bandwidth of each height h edge, which equals the bisection bandwidth of a height h tree. Even if $2b(h) < b(h+1)$ is satisfied in the configuration of an actual network switch, the flow of the edge at height $h+1$ never exceeds twice the flow at height h. So we treat $b(h+1)$ as the same as $2b(h)$ in this case. Thus $b(h+1) \leq 2b(h)$ is always satisfied. In a full bisection bandwidth network, $\frac{b(h)}{2^{h-1}}$ for all $1 \leq h \leq D$ is the same.

A buffer delay $d(h)$ is the latency caused by the message buffer in a switch and host. It can occur from the difference in one-way latency in the situation where the network is filled by burst communication and the situation where the network is not filled by messages. In this $d(h)$ calculation, the latency is measured between a left descendant leaf and a right descendant leaf from some height h intermediate node.

These parameters are measured by the communication between the node n_k and $n_{k \oplus 2^{h-1}}$ for all nodes, where the $\oplus$ operator is the exclusive-or operation. In this communication pattern, edges with a height smaller than or equal to h are involved in the communication while other edges not meeting this criterion are not involved. The $d(h)$ parameter denotes the time which the packet spends in the buffer of an intermediate node when the contention occurs in the height h subtree.

The proposed model can express an arbitrary switch configuration. For example, suppose that an actual switch is configured as shown in Figure 7. In this example, there are no switches at height 2 and 3 intermediate nodes, but there are two switches, each of which has eight links, in a height 4 intermediate node. In that case, the model expresses the switch as follows:

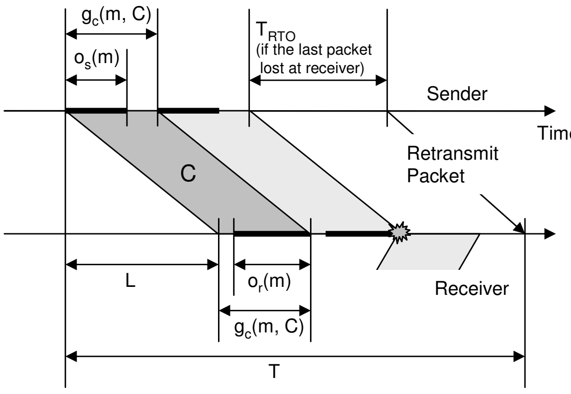


Fig. 8. Parameters in the PlogPT model

$$\frac{b(2)}{2^1} = \frac{b(3)}{2^2} = \frac{b(4)}{2^3} \tag{2}$$
$$d(2) = d(3) = d(4) \tag{3}$$

We define the predicate $P_S(h, h')$ which denotes whether the switches between height h and h' exist as follows:

$$P_S(h, h') \iff \neg((2^{h'-h}b(h) = b(h')) \wedge (d(h) = d(h'))) \tag{4}$$

In practice, the measured $b(h)$ and $d(h)$ contain errors, and thus $P_S(h, h')$ should absorb them. The practical definition of $P_S(h, h')$ is given using the error factor ε_S as follows:

$$P_S(h, h') \iff \left(\log\left(2^{h'-h}\frac{b(h)}{b(h')}\right)\right)^2 + \left(\log\frac{d(h)}{d(h')}\right)^2 > \varepsilon_S \tag{5}$$

As noted in section II-E, inexpensive ethernet switches have small internal sub-switches and therefore communication performance may differ between intra sub-switch communication and inter sub-switch communication. Such a switch configuration is also well expressed in this binary tree model.

B. PlogPT communication model

We propose the PlogPT (PlogP based on Tree) model by extending the PlogP model. Figure 8 shows the roles of each parameter.

1) Improvement in PlogP Parameter Measurement: In section II, we pointed out the fact that the method proposed in the paper [9] does not consider the behavior of bidirectional communication. Most collective operations, including all to all communication, are implemented using bidirectional communications instead of unidirectional communications. The PlogP parameters should be measured not only unidirectional but also bidirectional communications. In the PlogPT model, all the PlogP parameters are measured for bidirectional concurrent communication.

2) Bottleneck bandwidth: To apply the effect of bottleneck links represented in the binary tree model, the PlogPT model introduces a gap parameter $g_c(m, C)$ for each communication C which starts at t_s. Let the throughput of the communication C at time t be $b_c(C, t)$. The value of $g_c(m, C)$ is derived as the only solution for the following equation.

$$b(1)g(m) = \int_{t_s}^{t_s+g_c(m,C)} b_c(C,t) \tag{6}$$

Note that $g(m)$ is the gap at height 1 in the PlogPT model. These values do not depend on the pair of communicating nodes because the same bandwidth for all switches in height 1 is assumed in this model.

The left part of this equation expresses the amount of data to be sent, including overheads. It is equal to the temporal integration of the actual throughput. Let $\mathbf{E}(C, t)$ be the set of edges which the communication C uses at time t. We also define that $c(E)$ be the number of communications used in the edge E at time t. Then, the throughput $b_c(C, t)$ is calculated by the following equation.

$$b_c(C,t) = \min_{E \in \mathbf{E}(C,t)} \left(\frac{b(h)}{c(E)} \right) \tag{7}$$

We assume that the bandwidth of bottleneck edges are split among each communications fairly.

3) Penalty of Retransmission Timeouts: The PlogPT model assumes that a packet is lost when the buffer in a switch is full. Such a case may happen when the number of communications participated in by a switch increases. We consider the situation that communication C ends and communication C' starts at the same time, t_C, and the destinations of these communications are the same. Let $h(C)$ and $m(C)$ be the height and the message size of communication C, respectively. We define $B(C)$ as the total size of buffers in switches which are involved in the communication C that finishes at time t_f.

$$B(C) = d(h(C))b_c(C, t_f) \tag{8}$$

Let $P_B(C)$ be a predicate denoting whether the buffer of the switch is filled by the communication C.

$$P_B(C) \iff B(C) < m(C) \tag{9}$$

RTO may occur at message C if the following conditions are satisfied:

- Existence of C' whose destination and timing are mentioned above,
- $P_B(C)$ is true, and
- $P_S(h(C), h(C'))$ is true.

If RTO occurs, the communication C additionally takes T_{RTO}.

The constant T_{RTO} denotes the RTO time. The value of T_{RTO} is 200 ms in Linux TCP environment.

C. Parameters of the PlogPT model

In addition to the PlogP parameters, $b(h), d(h)$ and $g(m)$ for bidirectional communication are introduced in PlogPT. The values of parameter h in $b(h)$ and $d(h)$ is proportional to D, the logarithm of the number of nodes, and the number of parameter m in $g(m)$ is the same as that in unidirectional $g(m)$ in the PlogP model. Therefore, the additional measurement doesn't break the scalability of parameter measurement.

The bidirectional PlogP $g(m)$, are measured the same way as the fast measuring method of PlogP [9], except for that communication nodes in the measurement acts the roles of sender and receiver simultaneously.

In order to measure $b(h)$ and $d(h)$, we split the computation nodes $c_0, \ldots, c_{2^n-1}$ in height h tree into two groups, the left descendants $c_0, \ldots, c_{2^{n-1}-1}$ and the right descendants $c_{2^{n-1}}, \ldots, c_{2^n-1}$, and measure the communication between the two groups. The $b(h)$ is obtained as the sum of the bandwidth of burst communications from c_i to $c_{2^{n-1}+i}$. The ping-pong latency between the leftmost node c_0 and rightmost node c_{2^n-1} are measured under this workload. The ping-pong latency of the same nodes are also measured without this workload. The $d(h)$ is obtained as the difference between these ping-pong latencies.

The parameters $b(h)$ and $d(h)$ are also measured for bidirectional communications in the same way as the bidirectional $g(m)$. The only one difference from the unidirectional communication is that the buffer latency parameters $d(h)$ is calculated as the half of the difference of the ping-pong latencies.

D. Estimation of Execution time

We propose a method for estimating execution times of MPI communications based on the PlogPT model.

First, we replace MPI point to point communication functions with simple operations, Isend, Irecv, and Wait where both Isend and Irecv take three arguments source, destination, and message size and Wait takes one argument to specify the corresponding operation. That is, MPI_Send and MPI_Recv are replaced with the combinations of Isend, Irecv, and Wait. MPI collective operations are defined by a sequence of those simple primitives.

Let the operation $O[n, t]$ be introduced to express the tth operation issued in the process of the MPI rank n. Each MPI process is supposed to be assigned in each compute node. The following variables are associated with each operation $O[n, t]$: We refer to these variables as $O.e$ and so on.

- e: the time when the process enters the operation.
- l: the time when the process leaves the operation.
- f: the time when the process finishes the communication operation (Only for Isend and Irecv).

- f_{noRTO}: the time when the process finishes the communication if an RTO does not occur (Only for Isend).

The following constraints are held:

$$O[n, 0].e = 0 \tag{10}$$

$$O[n, i + 1].e = O[n, i].l \tag{11}$$

We will write simply O instead of $O[n, i]$ if the values of n and i are not important. The execution time of the MPI communication function described in operations $\{O[n, i]\}$ is $max(\{O[n, k_n].l\})$ for all processes n. $O.l$ is defined as follows:

$$O.l = \begin{cases} O.e & \text{if } O \text{ is Irecv or Isend} \\ max(O.e, O'.f) & \text{if } O \text{ is Wait for } O' \end{cases} \tag{12}$$

In this formula, the execution cost of both Irecv and Isend is ignored, but all costs are charged in the Wait operation.

Now we define an $O.f$ that is only available for the Isend and Irecv operations. First, $O.f_{noRTO}$ is defined as follows.

$$O.f_{noRTO} = O.l + g_c(m, O) \tag{13}$$

The value of the gap parameter $g_c(m, O)$ for operation O is derived from equation (6) in which the variable $b_c(C, t)$ is defined in expression (7). In order to obtain the $b_c(C, t)$ variable, all communications at time t must be counted.

Then, $O.f$ for Isend is defined as follows:

$$O.f = \begin{cases} O.f_{noRTO} + T_{RTO} & \text{if } \exists O' : \text{Isend s.t.} \\ & |O'.e - O.f| < \varepsilon_T, \\ & P_S(h(O), h(O')), \\ & P_B(O) \\ O.f_{noRTO} & \text{otherwise} \end{cases} \tag{14}$$

$O.f$ for Irecv is defined as follows:

$$O.f = O'.f + L \tag{15}$$

where O' is the corresponding Isend operation of O.

In equation (14), the last two factors of conditional clause indicates the condition of RTO delay described in Section III-B.3. The threshold value ε_T is used to absorb the jitters of packet arrival timing, which is ordinarily larger than the time consumed to process one packet. The value of ε_T should be a few times larger than that time.

E. An Example

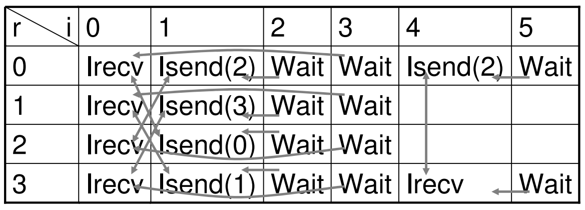
For example, we consider a sample sequence of operations shown in Figure 9 on a cluster consisting of four nodes. Let the PlogPT parameters be as follows: $b(1) = 3, b(2) = 4, g(m) = 8, L = 3$. That is, the height 2 edges are the bottleneck links.

The sequence of primitive operations in this program, $O[r, i]$ for all rank r, is illustrated in Figure 10. We will determine the time parameters for these operations as follows.

Trivially, the variables $O[i, 0].e, O[i, 0].l, O[i, 1].e, O[i, 1].l$ and $O[i, 2].e$ should be 0. The Wait operation at $O[i, 2]$ of MPI process rank i waits for Isend operations at $O[i, 1]$. It's leave time $O[i, 2].l$ equals the finish time of Isend $O[i, 1].f$.

```
Sample(int r, int m) {// r: Rank , m: message size
  Irecv(r, r xor 2, m);
  Isend(r, r xor 2, m);
  Wait(Isend);
  Wait(Irecv);
  if (r = 0) {
    Isend(0, 3, m);
    Wait(Isend);
  if (r = 3) {
    Irecv(3, 0, m);
    Wait(Irecv);
  }
}
```

Fig. 9. Sample Program



r \ i	0	1	2	3	4	5
0	Irecv	Isend(2)	Wait	Wait	Isend(2)	Wait
1	Irecv	Isend(3)	Wait	Wait		
2	Irecv	Isend(0)	Wait	Wait		
3	Irecv	Isend(1)	Wait	Wait	Irecv	Wait

The argument of an Isend shows its destination node.
An arrow shows a relationship between operations.

Fig. 10. Sequence of MPI Operations $O[r, i]$ in the Sample MPI program

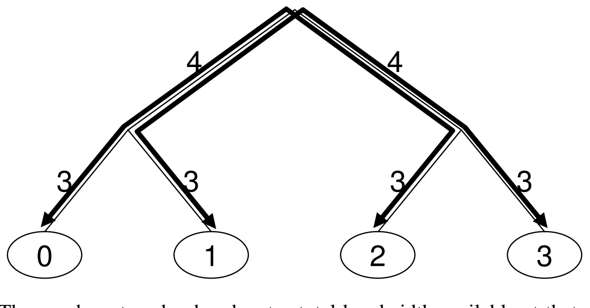
Then we have to determine the throughput of these Isends $b_c(O[i, 1], t)$ for all MPI processes in order to calculate their finish time. The communication pattern while these Isends running are is shown in Figure 11. The height 2 edges are the bottleneck link of these communications. The throughput $b_c(O[i, 1], t)$ is calculated as $\frac{1}{2}b(2) = 2$ from equation (7). Substituting this value in equation (6), the following simple equation for $O[i, 1].f_{noRTO}$ is derived.

$$3 \cdot 8 = \int_0^{O[i,1].f_{noRTO}} 2dt \tag{16}$$

$$O[i, 1].f_{noRTO} = 12 \tag{17}$$

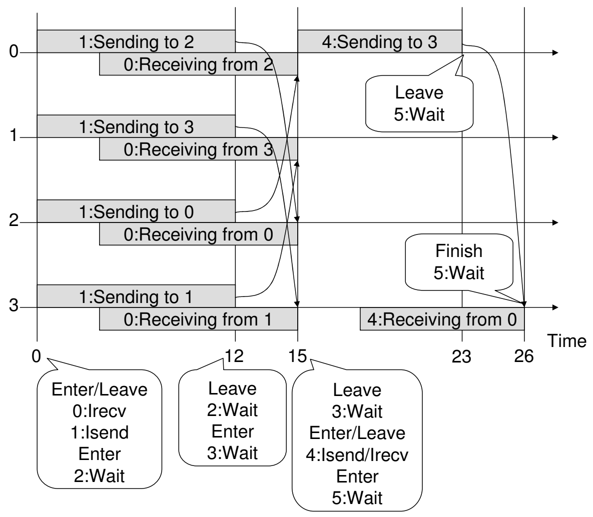
The condition for RTO delay at equation (14) requires the existence of an Isend starting at time 12. Since all processes wait for the completion of the Irecv issued at time 0, no such Isend exists. Therefore, $O[i, 1].f = O[i, 2].l = 12$ is derived.

Next, we will determine the leave time of Wait proce-



The number at each edge denotes total bandwidth available at that edge.

Fig. 11. Communication pattern during first Isends



The leading number j of operations show the index of the operation.
The rectangle shows the timing of actual communications.
The balloons show the timing of MPI operations.

Fig. 12. Timing of each operation in the sample procedure

dures for Irecvs, $O[i,3].l$. From the constraints, $O[i,3].l = O[i,0].f = O[i \oplus 2, 1].f + L = 15$ are derived.

The timing variables of operation $O[i,4], O[i,5]$ for $i = 0, 3$ is derived in the same way as shown above. Finally, the leave time of the last Wait $O[3,5]$ is derived as 26, which is the expected time of total communications in this sample program. The timing of enter and leave for all operations is described in Figure 12.

IV. Evaluation

We compared the actual execution time of the $P = 2^D$ process MPI_Alltoall operations shown in Figure 13 with the estimated time in both the PlogP model and the PlogPT model. Note that the interface of MPI functions in this figure is simplified. We omitted the execution time of local copy in the prediction because the overhead of local copy is hidden by the latency of communication.

A. Environment

The evaluation was obtained on the cluster shown in Table I. We use YAMPI [8] as the MPI environment. In this cluster, the flow control between computation nodes and the switch is not enabled. The bisection bandwidth per node is shown in Table II, which becomes narrower at height 5 of the tree. Each MPI process is assigned in each compute node.

B. Alltoall

First, we estimate the execution time of these implementations in the PlogP model. The communication pattern is independent from the rank of the node for both algorithms. Thus, the execution time in only one process is analyzed in this model. In Alltoall_A, as the message is sent after the arrival of the previous message and there are no overlaps, these messages are completely independent. So, the total time

```
Alltoall_A(int P, int r, int m) {
  // P: Number of Process, r: Rank m: size
  for (i = 0 to P - 1) {
    MPI_Irecv(r xor i,m);
    MPI_Send(r xor i, m);
    MPI_Wait();
  }
}

Alltoall_B(int P, int r) {
  // P: Number of Process, r: Rank of the node
  for (i = 0 to P - 1)
    if (i != r) MPI_Irecv(i, m);
  for (i = 1 to P - 1)
    MPI_Send(r xor i, m);
  MPI_Waitall();
}
```

Fig. 13. Alltoall algorithms

```
Alltoall_A(int P, int r, int m) {
  // P: Number of Processes, r: Rank, m: size
  for (i = 0 to P - 1) {
    Irecv(r, r xor i, m);
    Isend(r, r xor i, m);
    Wait(Irecv);
    Wait(Isend);
  }
}

Alltoall_B(int P, int r, int m) {
  // P: Number of Processes, r: Rank, m: size
  for (i = 0 to P - 1) {
    if (i != r) {
      Irecv(r, i, m);
    }
  }
  for (i = 1 to P - 1) {
    Isend(r, r xor i, m);
    Wait(Isend);
  }
  for (i = 0 to P -1) {
    Wait(Irecv);
  }
}
```

Fig. 14. Transformation of Alltoall algorithms

spent in the entire communication is estimated as the simple summation of time spent in each message $g(m) + L$. The estimated execution time in the PlogP model is as follows:

$$T^{A}_{\text{PlogP}} = (P-1)(g(m)+L) \qquad (18)$$

On the other hand, in Alltoall_B, the sender never waits for the arrival of previous messages. Therefore, the latency L is hidden by the next transmission. The estimated execution time is as follows:

$$T^{B}_{\text{PlogP}} = (P-1)g(m)+L \qquad (19)$$

Comparing these estimation times, the following inequality is brought forth:

$$T^{A}_{\text{PlogP}} > T^{B}_{\text{PlogP}} \qquad (20)$$

This means that Alltoall_B is estimated to be always faster than Alltoall_A in the PlogP model.

Then, we estimate the execution time of these implementations in the PlogPT model proposed in this paper. The sequence of operations is a repetition of Irecv, Isend, Wait(Isend), Wait(Irecv) as shown in Figure 14. Let $C_{r,k}$ be

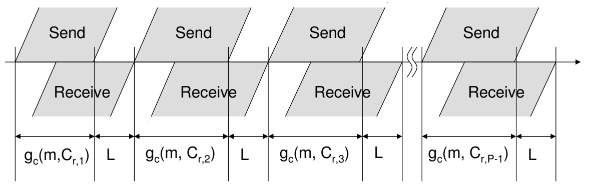


Fig. 15. Communications in Alltoall_A

the kth communication whose receiver node is r. The first Isend starts at time 0 at each process, and these Isends of all processes communicate simultaneously through the height 1 network. An RTO never occurs at this time because there is no process that starts an Isend based on this timing. Thus, the time spent in these Isends is $g_c(m, C_{r,1}) = g(m)$.

The leave time of Wait for Irecv is L after the finish time of Isend, and is equal to the next Isend's start time. After that, the next Isend starts at time $g(m) + L$ and height 2 communications occur in each process, simultaneously. The throughput and the execution time of these communications are shown as follows:

$$b_c(C_{r,2}, t) = \frac{1}{2} b(2) \tag{21}$$

$$g_c(m, C_{r,2}) = 2\frac{b(1)}{b(2)} g(m) \tag{22}$$

Continuing similar calculation, finally, we can obtain the following total execution time estimated by the PlogPT model:

$$T_{\mathrm{T}}^{A} = \sum_{h=1}^{D} \left(2^{h-1} \left(2^{h-1} \frac{b(1)}{b(h)} g(m) \right) \right) + (P-1)L \tag{23}$$

We consider the execution time of Alltoall_B similarly. As shown in Figure 14, the sequence of operations is as follows: $\{\{\text{Irecv}\}^{P-1}, \{\text{Isend, Wait(Isend)}\}^{P-1}, \{\text{Wait(Irecv)}\}^{P-1}\}$, where the superscript denotes repetition of the same primitives. We use $C_{r,k}$ with the same meaning as that given above.

The list of heights of Isends is as follows: $\{1, \{2\}^2, \ldots, \{k\}^{2^{k-1}}, \ldots, \{D\}^{2^{D-1}}\}$, where the superscript denotes repetition as well. The start time of an Isend is the finish time of the previous Isend. The latency L is added to the execution time only once by the Wait(Irecv) at the end of the communication. The total execution time will be $\sum_i g_c(m, C_{r,i}) + L$ if an RTO never happens. However, as there are Isends starting at the time another Isend ends its communication, the RTO condition in equation (14) can be satisfied.

From the equation $P_S(h, h')$ in the condition and the fact that $P_S(h, h) = false$, the RTO will occur at most $D - 1$ times when the height of communication changes. As in Figure 16, the finish time of the Isends will delay the T_{RTO} time. Considering the other condition in equation (14), the finish time of Isends gets delayed at most at the same frequency as

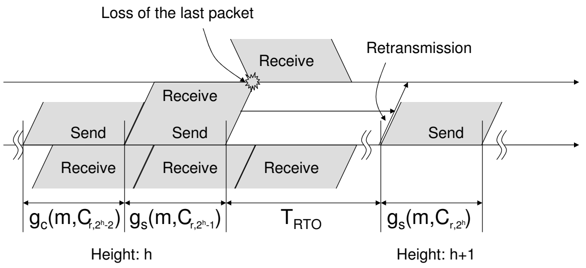


Fig. 16. Occurence of the Retransmission Timeout in Alltoall_B

the number of h which satisfies the following proposition.

$$P_S(h, h+1) \land P_C(C_{r,2^h-1}, C_{r,2^h}) \tag{24}$$

Let N_{RTO} be the number height h that satisfies this proposition. Although total execution time depends on which nodes to delay, which is random, there is a pattern that all RTO delays occur in one certain node. Thus, the worst case delay time is $N_{RTO}T_{RTO}$.

Finally, the total execution time of Alltoall_B is calculated as follows:

$$T_{\mathrm{T}}^{B} = \sum_{h=1}^{D} \left(2^{h-1} \left(2^{h-1} \frac{b(1)}{b(h)} g(m) \right) \right) + L + N_{RTO} T_{\mathrm{RTO}} \tag{25}$$

Focusing on the fact that $T_{\mathrm{RTO}} \gg L$, the execution time estimated by the PlogPT model derives the following inequation:

$$T_{\mathrm{T}}^{A} < T_{\mathrm{T}}^{B} \text{ if } \exists h \text{ satisfies proposition (24)} \tag{26}$$

$$T_{\mathrm{T}}^{A} > T_{\mathrm{T}}^{B} \text{ otherwise} \tag{27}$$

This implies that Alltoall_A can be faster than Alltoall_B in certain condition, which is a different consequence from that based on the PlogP model.

Now, the values of the PlogPT parameters of the evaluation environment are applied to these estimated expressions. The PlogPT parameters, which are related to the binary tree model, are shown in table II. In this network, the equation $P_S(h, h+1) = false$ is satisfied when h is 2 or 3.

Figure 17 shows the execution time of these algorithms and an estimated one in two cases: 4 and 32 nodes. Experimental data was taken 10 times for each case and plotted individually. In every case, the PlogPT model was able to estimate more accurately than the PlogP model.

We found that the probability of retransmission timeout occurrence becomes higher when the number of computing nodes increases. On the other hand, the timeout still occurs with a certain amount of probability when the number of nodes is small, and thus makes a certain impact on execution time. In either case, the timeout doesn't occur when the message size is smaller than a particular size. These results correspond to the expectation of RTO behavior mentioned in section III-B.3.

When the number of nodes is large, the RTO happens even in Alltoall_A with a low probability. One possible reason for

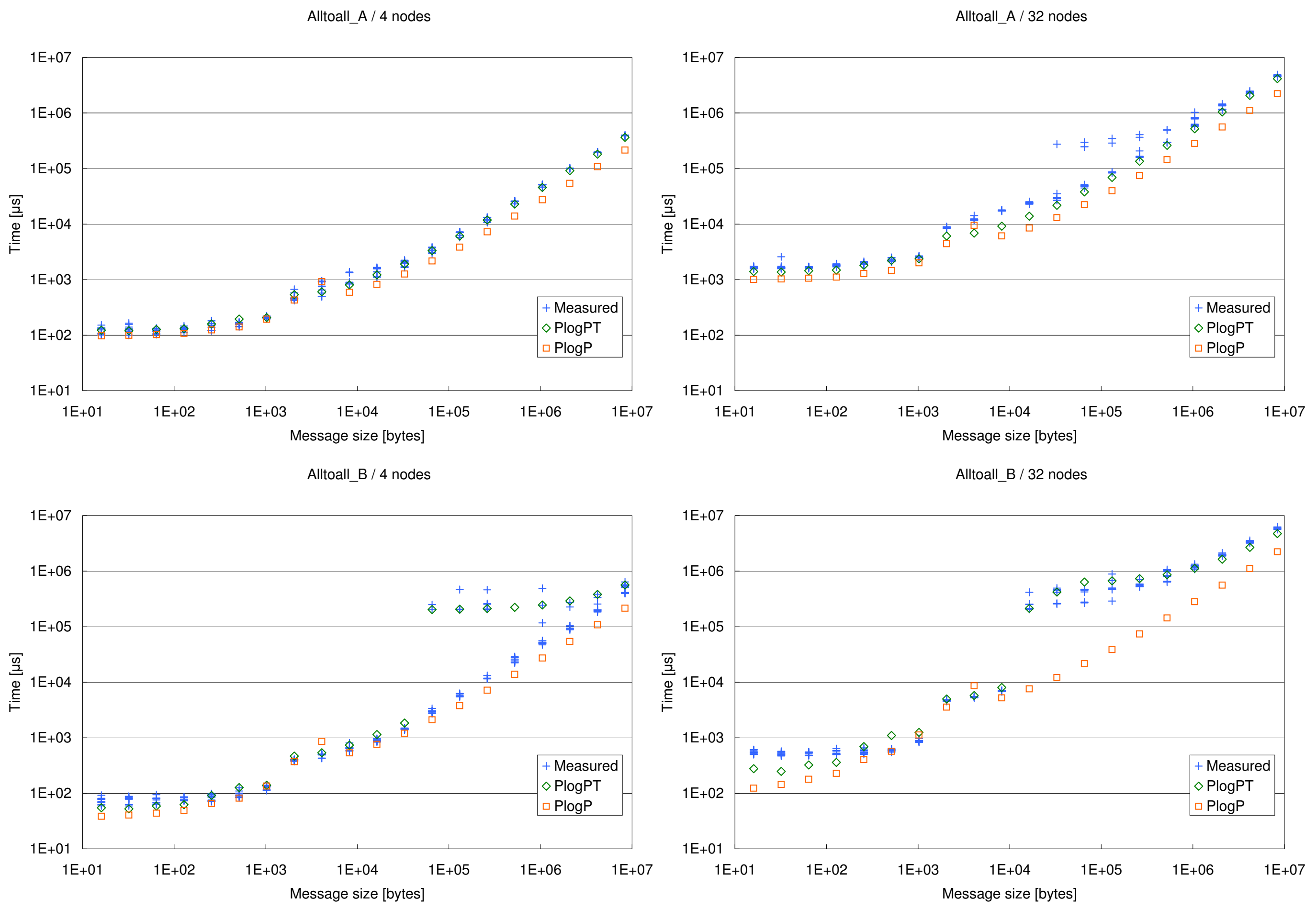


Fig. 17. Estimated execution time and actual execution time

this phenomenon is that the accumulated jitter exceeds the latency L and the loss of the last packet is caused for the same reason as that for Alltoall_B.

V. Related Work

A. Queuing extensions of logP-like models

The loGPC model [3] extends both the logP model and the logGP model to account for the impact of network contention and DMA behavior. This model uses logP parameters or logGP parameters depending on the message size, that is dominant for whichever model covers the situation. This model adds the delay from network contentions calculated from the M/G/1 queuing model.

In the logPQ model [5], which is an extension of the PlogP model, the network is modeled using the transmission, transfer, and receive queues. The bandwidths and latencies between each queue are parameterized. This model expresses the transition of messages on these queues sequentially, and shows where contention occurs and spreads into. However, the packet level simulation of these queues is needed to analyze the execution time. Information specific to underlying MPI implementations is required in order to do the packet level simulation. Furthermore, a method for analyzing queue parameters is not mentioned or trivial.

The LoGPC model and the LogPQ model assume a flow control mechanism which stops the message sending immediately when the next destination queue is full. This is true only if flow control is performed by the hardware layer. If flow is controlled by the software layer, it is not possible to reserve the bandwidth of the data link ahead of time, especially that of switches. Instead, the packet is transmitted speculatively and when contention occurs and the buffers run out, spilled packets are dropped. Those spilled packets are expected to be transmitted by the software layer. These behavior patterns are very different from those that these models expect.

B. Contention Aware PlogP Extensions

In the paper [7], the contention parameter γ is introduced into a logP-like model. In this model, the contention factor parameter γ is set equal to the number of processors P, and the time spent in communication is expressed as $T_{comm} = l + \frac{b\gamma}{W}$. The parameter W denotes the bandwidth and b denotes the message size. In this model, the bandwidth dependent element of the communication cost is proportional to the number of nodes.

In the paper [13], the slowdown of TCP/IP communication caused by contention in 100 Mbps Ethernet networks is considered. The contention factor parameter γ is introduced into the PlogP model to simulate the slowdown caused by contention, where the γ is given by observed results.

Both models simulate the bandwidth degradation caused by contention. But the contention factor is independent from the communication pattern in the network. Moreover, they do not consider the effect of an RTO caused by tail drop.

C. Application of Communication Models

The paper [14] reports that the Hockney [15], LogP [1], and LogGP [2] models provide useful insights into various aspects of different collective communication algorithms, though these models do not consider network congestion. This conclusion depends on whether or not bottleneck network links exist. As already shown in Figure 17, unpredicted performance was observed in such a model.

VI. Conclusion

We have proposed the PlogPT model, which is an extension of the PlogP model, to express communication performance in a commodity network, such as TCP/IP over Ethernet. In the PlogPT model, the network topology is modeled as a binary tree connection where an edge, an intermediate node, and a leaf node are a network link, a switch, and a computer, respectively. The two links of the intermediate node have the same bandwidth. All bandwidths of links are estimated by executing test programs. Unlike other traditional estimation programs, the network bandwidth is evaluated using the bidirectional communication pattern.

Unlike the PlogP model, the network gap, expressing the time of message transfer, is expressed by $g_c(m, C)$ for each communication C that has a message size of m. $g_c(m, C)$ is derived from throughput $b_c(C, t)$ calculated based on the other communications that share the same communication path. The occurrence of RTO (retransmission timeout), which is dominant in the execution time of communications, is predicted based on a combination of predicates which shows the change in the number of intermediate switches $P_S(h(C), h(C'))$ and messages in the buffer $P_B(C)$ for all transitions from communication C to communication C'. As a result, the accuracy of execution time estimation is improved.

The execution time of two all-to-all communication algorithms have been estimated and have been compared to the actual execution time. The result shows that all estimates are very close to the actual execution time, while the existing model, the PlogP model, does not estimate them accurately.

There are some unresolved problems for the PlogPT model. The fact that the accumulation of jitter affects the RTO condition is not included in this model. We have stabilized the PlogPT parameters by measuring them several times. Other approaches to stabilizing parameters are needed.

References

[1] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: Towards a realistic model of parallel computation," in *Principles and Practice of Parallel Programming*, 1993, pp. 1–12.

[2] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman, "LogGP: Incorporating long messages into the LogP model for parallel computation," *Journal of Parallel and Distributed Computing*, vol. 44, no. 1, pp. 71–79, 1997.

[3] C. A. Moritz and M. I. Frank, "LoGPC: Modeling network contention in message-passing programs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 4, pp. 404–415, 2001.

[4] T. Kielmann, H. E. Bal, and S. Gorlatch, "Bandwidth-efficient collective communication for clustered wide area systems," in *IPDPS 2000*, Cancun, Mexico, May 2000, pp. 492–499.

[5] T. Touyama and S. Horiguchi, "Performance evaluation of practical parallel computation model LogPQ," *ispan*, vol. 00, p. 216, 1999.

[6] Cisco Systems, Inc, "Cisco Catalyst 4500 series line cards data sheet," http://www.cisco.com/en/US/products/hw/switches/ps4324/products_data_sheets_list.html.

[7] M. J. Clement, M. R. Steed, and P. E. Crandall, "Network performance modeling for PVM clusters," 1996.

[8] Y. Ishikawa, "YAMPI, yet another MPI implementation," http://www.il.is.s.u-tokyo.ac.jp/yampi/.

[9] T. Kielmann, H. E. Bal, and K. Verstoep, "Fast measurement of LogP parameters for message passing platforms," *Lecture Notes in Computer Science*, vol. 1800, pp. 1176–1178, 2000.

[10] Infiniband Trade Association, "Infiniband specification," http://www.infinibandta.org/specs/.

[11] Myricom, inc., http://www.myri.com/.

[12] M. Matsuda, T. Kudoh, Y. Kodama, R. Takano, and Y. Ishikawa, "TCP adaptation for MPI on long-and-fat networks," in *Proceedings of the 2005 IEEE International Conference on Cluster Computing (Cluster 2005)*, 2005.

[13] L. Barchet-Estefanel and G. Mounié, "Performance characterisation of intra-cluster collective communications," in *Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2004)*, 2004, pp. 254–261.

[14] J. Pješivac-Grbović, T. Angskun, G. Bosilca, G. E. Fagg, E. Gabriel, and J. J. Dongarra, "Performance analysis of MPI collective operations," in *Proceedings of the 2005 IEEE International Conference on Cluster Computing (Cluster 2005)*, 2005.

[15] R. W. Hockney, "The communication challenge for mpp: Intel paragon and meiko cs-2," *Parallel Computing*, vol. 20, no. 3, pp. 389–398, 3 1994.