*Science in China Series F: Information Sciences*



www.scichina.com
info.scichina.com
www.springer.com/scp
www.springerlink.com

# LogGPO: An accurate communication model for performance prediction of MPI programs

CHEN WenGuang†, ZHAI JiDong, ZHANG Jin & ZHENG WeiMin

Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

**Message passing interface (MPI) is the de facto standard in writing parallel scientific applications on distributed memory systems. Performance prediction of MPI programs on current or future parallel systems can help to find system bottleneck or optimize programs. To effectively analyze and predict performance of a large and complex MPI program, an efficient and accurate communication model is highly needed. A series of communication models have been proposed, such as the LogP model family, which assume that the sending overhead, message transmission, and receiving overhead of a communication is not overlapped and there is a maximum overlap degree between computation and communication. However, this assumption does not always hold for MPI programs because either sending or receiving overhead introduced by MPI implementations can decrease potential overlap for large messages. In this paper, we present a new communication model, named LogGPO, which captures the potential overlap between computation with communication of MPI programs. We design and implement a trace-driven simulator to verify the LogGPO model by predicting performance of point-to-point communication and two real applications CG and Sweep3D. The average prediction errors of LogGPO model are 2.4% and 2.0% for these two applications respectively, while the average prediction errors of LogGP model are 38.3% and 9.1% respectively.**

performance prediction, communication model, LogP, LogGPO, MPI

## 1 Introduction

Message passing interface (MPI)[1] is the de facto standard in writing parallel scientific applications on distributed memory systems. Performance prediction of MPI programs on current or future parallel systems can help people find system bottleneck or optimize programs[2,3]. To effectively analyze and predict performance of a large and complex MPI program, an efficient and accurate communication model is highly needed[4].

A series of hardware-parameterized communication models have been proposed for this target, such as the LogP model family. The LogP model[5] abstracts the communication performance of a platform by four parameters: the communication latency ($L$), the overhead ($o$), the bandwidth for small message ($1/g$), and the number of processors ($P$). The LogGP model[6] has one more parameter ($G$) than the LogP model and captures special support for long messages on most of plat-

Received September 21, 2008; accepted December 11, 2008
doi: 10.1007/s11432-009-0161-2
†Corresponding author (email: cwg@tsinghua.edu.cn)
Supported by the National High-Tech Research & Development Program of China (Grant No. 2006AA01A105)

**Citation:** Chen W G, Zhai J D, Zhang J, et al. LogGPO: An accurate communication model for performance prediction of MPI programs. Sci China Ser F-Inf Sci, 2009, 52(10): 1785–1791, doi: 10.1007/s11432-009-0161-2

forms. The effects of network contention on communication cost are analyzed in LoPC model[7] and LoGPC model[8].

Although these models show good accuracy for low-level communication libraries such as Active Message or Elan Library, hardware-parameterized models ignore the increasing effects of high-level communication libraries, such as MPI, on communication cost. To address this problem, some software-parameterized models have been proposed, such as LogGPS model[9] and $\text{Log}_n\text{P}$ model[10]. LogGPS model analyzes the communication cost needed for different protocol switching in MPI programs. $\text{Log}_n\text{P}$ model considers the cost of memory operations for non-continuous messages transmission.

Non-blocking communication is widely used to hide communication latency through overlapping useful computation in MPI programs. The potential overlap degree of computation and communication can have great impact on communication performance of MPI programs. However, most contemporary MPI implementations are not able to provide true overlap between computation and communication even with nonblocking message passing interface. In fact, significant communication overhead can be introduced at actual message transmission.

The overlap degree are determined by the strategy of communication progress, communication protocol and transfer mechanism inside MPI implementations. Traditional communication models ignore the effect of overlap on communication cost, such as overhead caused by progression strategies and underlying communication protocols. They normally assume that maximum overlap degree between computation and communication. However, this assumption does not always hold for MPI programs because either sending or receiving overhead introduced by MPI implementations can decrease potential overlap for large messages. As a result, the performance predicted with these models are unrealistic. Figure 1 shows an example. If the computation time is large enough, the communication cost can be ignored with LogGP models. In fact, significant overhead can be introduced at both sending process (PROCESS 0) and receiving process (PROCESS 1) when rendezvous protocol and dependent communication progress are taken.

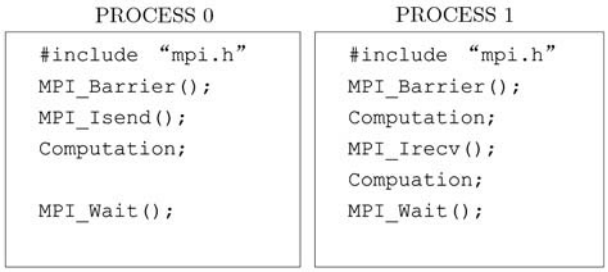


**Figure 1** An example of overlap of computation and communication.

Thus, it is clear that existing models fail to make accurate prediction for MPI programs and a new communication model is desired. This paper makes the following contributions:

• We propose the LogGPO model which extends the LogGP model by capturing the overhead caused by high-level communication library and characterize the potential overlap between computation and communication in MPI programs.

• To verify our proposed model, we design and implement a trace-driven simulator, SIM-MPI[11], which can help developers to analyze and predict performance of MPI programs. The underlying communication models used by SIM-MPI are configurable. We have implemented both LogGP and LogGPO models in the simulator.

• We compare our LogGPO model with the LogGP model on two experimental platforms with both microbenchmarks and real applications. The prediction results based on LogGPO model are significantly more accurate than LogGP model.

The rest of this paper is organized as follows: The LogGPO model is described in section 2. In section 3 we present our experimental results for model verification. Some conclusions are given in section 4.

## 2 Modeling communication cost in MPI programs

### 2.1 The structure of LogGPO model

In order to analyze the communication cost accu-

rately, we propose the LogGPO model. For the parameters in LogGP model, please refer to ref. [6].

**Difference between LogGP and LogGPO model.** Compared with the LogGP model, the LogGPO model has the following additional parameters:

• Parameter $S_{cp}$, defined as the strategy of communication progress supported in MPI implementations. The strategy can be dependent communication progress (DCP) or independent communication progress (ICP).

• Parameter $T_m$, defined as the transfer mechanism taken for actual message transmission. It can be a type of T, TA, RAT and RTA[11].

• Parameter $S$, defined as the switching point of eager protocol and Rendezvous protocol.

We redefine the parameter $O$ in the LogGP model. First, we distinguish the sending overhead and receiving overhead, $O_s$ and $O_r$. Second, besides the processor overhead, the overhead in LogGPO also includes synchronization cost, blocking time and message transmission time. The detailed parameters are listed in Table 1. $O_s$ and $O_r$ are total overhead incurred to send or receive a message. $O_{sw}$ and $O_{rw}$ are overhead incurred in the routine of MPI_Wait, which can include blocking time, synchronization time, etc. $O_i$ is the overhead of initializing the underlying device to process a message, which varies with the message size. $O'$ is the overhead for processing a control message. Normally, it has a constant value.

The strategy of communication progress, transfer mechanism and communication protocol together determine the potential overlap degree of computation and communication in MPI programs. We define the overlap ratio of computation to communication as

$$R_o = \frac{\text{comp}}{(\text{comp} + O)}.$$

Besides, the gap ($g$) has little effect on the communication cost of high-level communication routines. Therefore, we disgard $g$.

## 2.2 Communication costs under the model

In this section, we primarily analyze the communication costs and potential overlap ratio of the program. We assume that the underlying communication devices support offload function, and that the duration of user computation inserted is large enough.

**Table 1** Tokens used in modeling communication costs with LogGPO model

| Token | Description |
|---|---|
| $K$ | Message size (Byte) |
| $T_c$ | Total comm. cost for a msg. transmission |
| $O_s$ | Total overhead incurred at the sending end |
| $O_r$ | Total overhead incurred at the receiving end |
| $O'$ | Overhead for processing a control message |
| $O_i$ | Initialization overhead to start a transmission |
| $O_c$ | Overhead for memory copy |
| $O_{sw}$ | Overhead in routine of sending MPI_Wait |
| $O_{rw}$ | Overhead in routine of receiving MPI_Wait |
| $T_s$ | Arrival time of MPI_Isend |
| $T_r$ | Arrival time of MPI_Irecv |
| $T_{sw}$ | Arrival time of the sending MPI_Wait |
| $T_{rw}$ | Arrival time of the receiving MPI_Wait |
| $R_o$ | Overlap ratio of comp. to comm. |
| $R_{so}$ | Overlap ratio at the sending end |
| $R_{ro}$ | Overlap ratio at the receiving end |
| comp | The user computation |

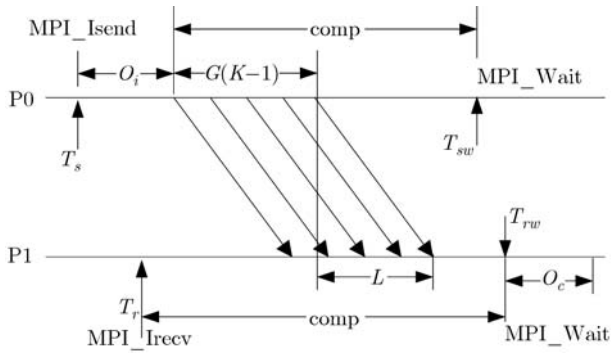


**Figure 2** Communication cost in eager protocol ($T_m = T$).

2.2.1 Eager protocol. When $K \leqslant S$, eager protocol is taken in Figure 2. For eager protocol, messages are normally copied to pre-defined buffer of receiving process directly and receiving process need to copy messages to user buffer. This process is finished at the routine of MPI_Wait. So, $O_c$ will get larger for large message. The strategy of communication progress has little effect on eager protocol. Eager protocol can get much larger overlap ratio for small messages.

$$O_s = O_i + O_{sw} = O_i,$$

$$O_r = O_c,$$
$$T_c = O_i + G(K-1) + L + O_c,$$
$$R_{so} = \text{comp}/(\text{comp} + O_i),$$
$$R_{ro} = \text{comp}/(\text{comp} + O_c).$$

2.2.2 Rendezvous protocol. When $K > S$, rendezvous protocol is taken. First, we will analyze $S_{cp}$ =DCP and $T_m$ =RAT. $O_{sw}$ and $O_{rw}$ are determined by the arrival time of MPI_Irecv. If MPI_Irecv arrives after the arrival of the REQ message, MPI_Irecv will send the ACK message to the sending process in Figure 3(a). Otherwise, the MPI_Irecv will do nothing and return immediately in Figure 3(b). MPI_Wait cannot transfer the message with RDMA WRITE until it receives the ACK message. $T_r - T_{sw}$ and $T_{rw} - T_{sw}$ are the blocking time due to load imbalance. $2O' + L$ is the synchronization cost for rendezvous protocol. These overhead decreases the potential overlap in MPI programs.

If $T_r > (T_s + O' + L)$, the overhead and communication cost are

$$O_{sw} = (T_r - T_{sw}) + 2O' + L + O_i + G(K-1),$$
$$O_s = O' + O_{sw},$$
$$O_r = 2O' + O_{rw} = 2O',$$
$$T_c = 2(2O' + L) + (T_r - T_{sw}) + O_i + G(K-1) + L = 4O' + 3L + (T_r - T_{sw}) + O_i + G(K-1).$$

If $T_r \leqslant (T_s + O' + L)$, the overhead and communication cost are:

$$O_{sw} = (T_r - T_{sw}) + 2O' + L + O_i + G(K-1),$$
$$O_s = O' + O_{sw},$$
$$O_{rw} = 2O' + L + O_i + G(K-1),$$
$$O_r = O' + O_{rw},$$
$$T_c = 2(2O' + L) + (T_{rw} - T_{sw}) + O_i + G(K-1) + L = 4O' + 3L + (T_{rw} - T_{sw}) + O_i + G(K-1).$$

For $S_{cp}$ =DCP and $T_m$ =RAT, the overlap ratio of sending process is dependent on the load balance of different progresses. The first case has larger overlap ratio due to less receiving overhead $O_r$, while the second case has a smaller overlap ratio. The total communication cost has little change for these two cases.

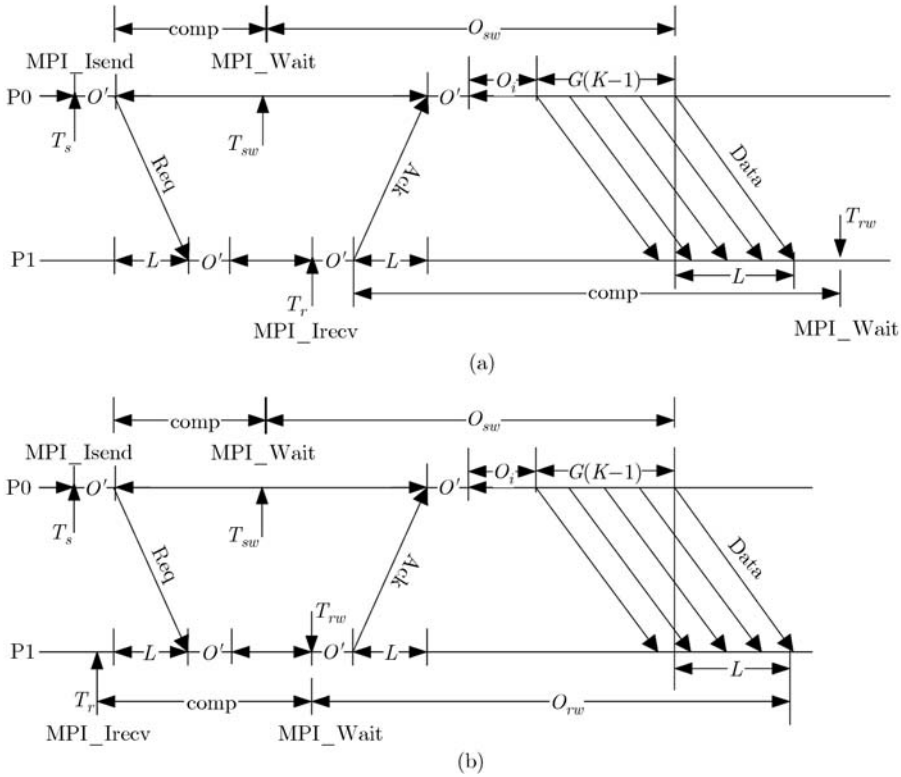


**Figure 3** Communication cost in rendezvous protocol ($T_m$ = RAT, $S_{cp}$=DCP). (a) $T_r > (T_s + O' + L)$; (b) $T_r \leqslant (T_s + O' + L)$.

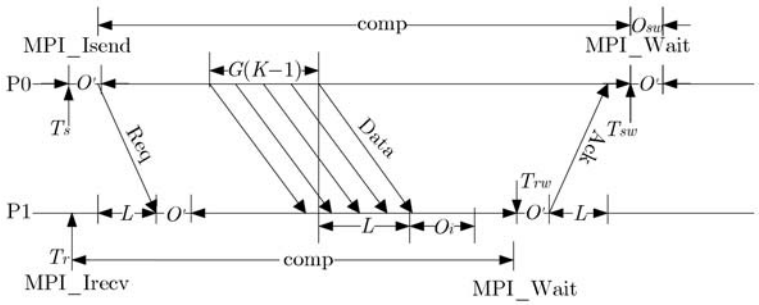


**Figure 4** Communication cost in rendezvous protocol ($T_m$= RTA, $S_{cp}$= ICP).

Next, we analyze the communication cost and overlap ratio with the support of independent communication progress. Normally, interruption or helper thread can be used to make independent progress. From Figure 4, we can find that the sending end has a much larger overlap ratio. And the receiving end is determined by the value of $O_i$. Although the receiving end can make progress independently, it must introduce overhead when processing interruption, etc.

The overhead and communication cost are calculated here:

$$O_s = O' + O_{sw} = 2O',$$
$$O_r = O' + O_{rw} = 2O' + O_i,$$
$$T_c = 2(2O' + L) + O_i + G(K-1) + L.$$

## 3 Experimental details

### 3.1 Experimental platforms

Our experimental platforms are two cluster servers, named Explorer3 and Dawning 4000A. Explorer3 is a 128-node 1.3 GHz Itanium2 cluster server connected by 10 Gbps Infiniband network. MPI over Infiniband is MVAPICH-0.9.2. The Dawning 4000A server consists of 512 nodes connected with Myrinet. Each node has four 2.2 GHz AMD Opteron850 processors. MPI over GM uses GM-2.1.2 with MPICH-1.2.6.

In order to analyze the performance of MPI programs with our model, we have designed and implemented a simulator SIM-MPI, which can be downloaded from website (http://www.hpctest.org.cn/resources/sim-mpi.tgz). The details about SIM-MPI can be found from its documents. The LogGP parameters are tested with logp_mpi benchmark tool provided by Kielman et al.[12].

### 3.2 Performance prediction with LogGPO

3.2.1 Point-to-point communication. We predict the performance of point-to-point communication with the derived LogGPO model parameters and compare this prediction with that of LogGP model. We use the Intel MPI Benchmark version 3.0 (IMB-3.0) to test the time of ping-pong program.

Figure 5(a) shows the P2P communication prediction with LogGPO model and LogGP model on the Explorer3 platform. On this platform the transfer mechanism of TA is used for small messages (<4 KB) and RPUT is used for large messages. Moreover, independent progress is not supported for large messages. So, a large overhead will be introduced in the course of communication which can overlap with message transmission. The LogGP model ignores this characteristic, so the predicted result is inaccurate. For messages larger than 4 KB, the predicted result with LogGP model is almost 3 times of the actual communication cost. The reason is that both the sending and receiving routines overlap with message transmission after the synchronization point. Our LogGPO shows high accuracy for all messages. The average relative error of our model is 3.2% for all of messages on Explorer3 platform. Figure 5(b) shows the results on Dawning 4000A platform. Our model shows a high accuracy on Dawning 4000A platform also, and the average relative error of our model is 3.7%. The mechanism of T is used for small

messages (<32 KB) at this platform and RAT is used for large messages. Independent progress is also not supported on this platform. So, for small messages, the LogGP model also shows acceptable predicted accuracy. But, for large messages, the LogGP model has significant errors.

3.2.2 Real applications. In this section, we compare the predicted accuracy of different communication models for applications with SIM-MPI simulator on Explorer3 Platform. The NPB CG kernel[13] and Sweep3D[4] are taken for validation. The CG kernel is used to find an estimate of the largest eigenvalue of a symmetric positive definite sparse matrix with a random pattern of nonzero. The data set is class C. Sweep3D is a three-dimensional particle transport problem. We collected the communication and computation traces of applications with SIM-MPI instrumentation library. These traces are used to predict the performance of MPI programs based on underlying communication models. The results are shown in Figure 6.

The LogGPO model is very accurate for both CG and Sweep3D programs. For CG program, the average error of LogGPO model is 2.4% and the maximum error is 6.6%, while those of the LogGP model are 38.3% and 65.3% respectively. For Sweep3D program, the average error of LogGPO model is 2.0% and the maximum error is 6.8%, while those of the LogGP are 9.1% and 14.1% respectively.

From Figure 6, the error of LogGP model for CG becomes larger as the number of processes increases, while the error decreases for Sweep3D. This is due to the communication patterns in applications. The message size in CG of class C is very large. Most of them are 150 or 300 KB. Furthermore the volume of communication increases with the number of processes. So, the communication overhead accounts for a large percentage of

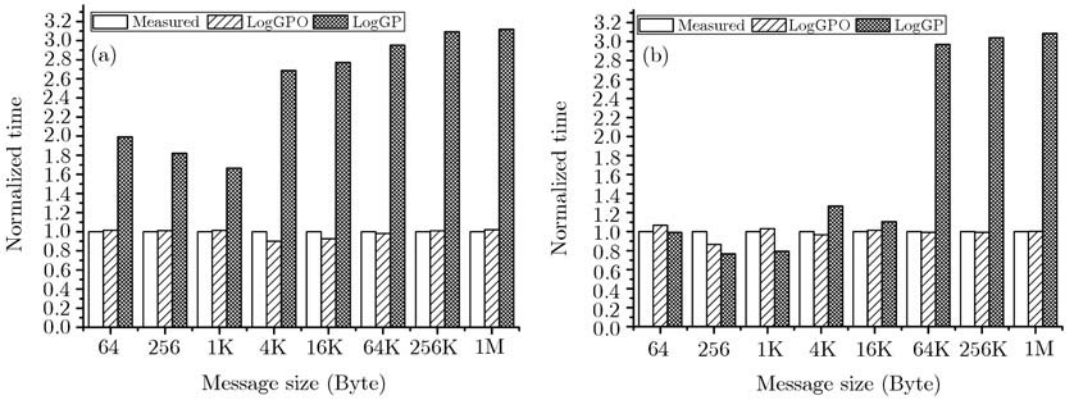


**Figure 5** Measured vs. predicted time of IMB-3.0 Ping-Pong communication with LogGPO and LogGP models. (a) Exploer3 platform; (b) Dawning 4000A platform.

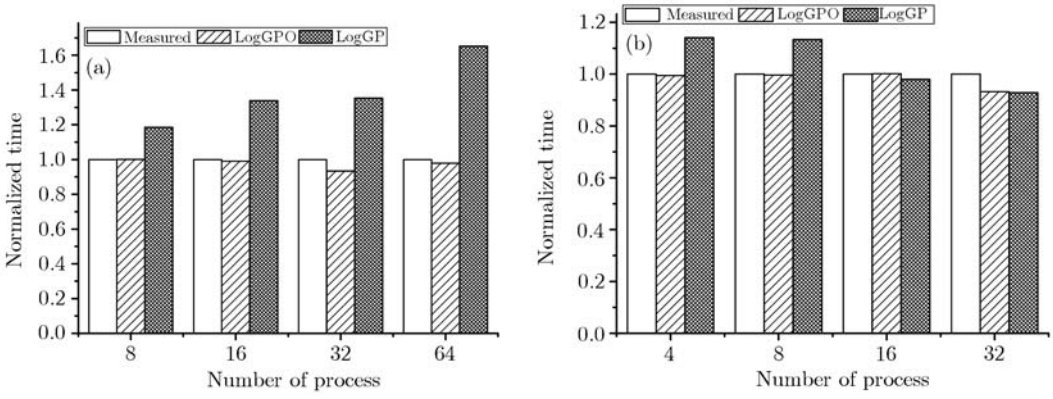


**Figure 6** Measured vs. predicted time of real applications with LogGPO and LogGP models. (a) NPB CG Class C; (b) Sweep3D: 50×50×50.

total execution time. In contrast, the message size in Sweep3D is between 2 and 3 KB. The communication overhead accounts for a small percentage of total execution time.

## 4 Conclusions

In this paper, we propose a new software-parameterized communication model, the LogGPO model, which captures the overlap between computation and communication. We verify the LogGPO model by predicting the performance of micro-benchmarks and applications with the SIM-MPI simulator. In the future, we plan to integrate the impact of network contention into our model.


1 Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. University of Tennesses, Knoxville, TN, June 1995

2 Petrini F, Kerbyson D J, Pakin S. The case of the missing supercomputer performance: Achieving optimal performance on the 8192 processors of ASCI Q. In: Proceedings of the 2003 ACM/IEEE Conference on Supercomputing. Washington, DC: IEEE Computer Society, 2003

3 Kerbyson D J, Alme H J, Hoisie A. Predictive performance and scalability modeling of a large-scale application. In: Proceedings of the 2001 ACM/IEEE conference on Supercomputing. New York: ACM, 2001

4 Sundaram-Stukel D, Vernon M K. Predictive analysis of a wavefront application using LogGP. SIPLAN Notices, 1999, 34(8): 141–150

5 Culler D. LogP: Towards a realistic model of parallel computation. In: Proceedings of 4th Symp Principles and Practice of Parallel Programming. New York: ACM, 1993. 1–12

6 Alexandrov A, Ionescu M F, Schauser K E, et al. LogGP: Incorporating long messages into the logP model—one step closer towards a realistic model for parallel computation. In: Proceedings of 7th ACM Symposium on Parallel Algorithms and Architectures. New York: ACM, 1995

7 Frank M, Agarwal A, Vernon M K. LoPC: Modeling contention in parallel algorithms. In: Proceedings of 6th ACM SIGPLAN symposium on Principles and practice of parallel programming. New York: ACM, 1997

8 Moritz C A, Frank M I. LoGPC: Modeling network contention in message-passing programs. IEEE Trans Parall Distr Syst, 2001, 12(4): 404–415

9 Ino F, Fujimoto N, Hagihara K. LogGPS: a parallel computational model for synchronization analysis. In: Proceedings of the Eighth ACM SIGPLAN Symposium on Principles and Practices of Parallel Programmng. New York: ACM, 2001

10 Cameron K W, Ge R. Predicting and evaluating distributed communication performance. In: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing. Washington, DC: IEEE Computer Society, 2004

11 Al-Tawil K, Moritz C A. Performance modeling and evaluation of MPI. J Parall Distr Comput, 2001, 61(2): 202–223

12 Kielmann T, Bal H E, Verstoep K. Fast measurement of LogP parameters for message passing platforms. In: Proceedings of 15th IPDPS 200 Workshops on Parallel and Distributed Processing. London: Springer-Verlag, 2000

13 Bailey D. The NAS parallel benchmarks. Technical Report, Report RNR-91-002 revision 2, NASA Ames Research Center, 1991