2009 11th IEEE International Conference on High Performance Computing and Communications

Towards Predictive Modeling of Message-Passing Communication

Verdi March*† Vijayaraghavan Murali* Yong Meng Teo* Simon See†‡ James T. Himer‡
*Department of Computer Science, National University of Singapore
†Sun Microsystems Inc.
‡Department of Mechanical & Aerospace Engineering, Nanyang Technological University
email: [verdimar,mvijayar,teoym]@comp.nus.edu.sg, [simon.see,james.himer]@sun.com

Abstract

Communication has been shown to be a performance bottleneck and a limiting factor of many large parallel applications. As such, predicting the application scalability necessitates a communication performance model. This paper investigates the LogGP communication performance model for predicting message-passing communications when the system configuration (i.e., number of nodes) is varied. The cost functions for the message-passing operations are based on MVAPICH2 1.0, and the experiments are conducted on the Ranger system using up to 256 nodes connected with an InfiniBand network. For point-to-point communications, we observe that the LogGP model accurately predicts the communication performance. However, the results for three collective operations, i.e., `MPI_Barrier`, `MPI_Alltoall`, and `MPI_Bcast`, are varying. For `MPI_Bcast`, the LogGP model is able to predict its scalability up to 256 nodes, and the prediction error is at most a factor of two on 256 nodes. For the remaining collectives, the scalability — bar that of `MPI_Alltoall` on small messages ($m = 2$ bytes) — is predicted by LogGP, but the prediction error for 256 nodes is 3.5–12 times of the measured performance.

1. Introduction

Parallel applications reduce the time to solve compute-intensive problems by utilizing multiple processors during their execution. A key metric to evaluate the performance of parallel applications is the *application scalability*. This metric quantifies the increase of an application performance when system configurations or workload are increased. An application is scalable if their performance is significantly improved with a higher number of processors. Scalable applications are desirable as they utilize a parallel system efficiently in solving a scientific problem.

On peta-scale computing systems, studying the application scalability is a challenging task. Firstly, a direct measurement approach is both time-consuming and costly, as it requires an application to be run on a large number of processors. Secondly, a scalability study may be requested for systems that are yet to be deployed. To address these challenges, a number of *performance prediction* approaches are proposed [24], [10], [25]. A key factor in performance prediction is to understand the *communication cost* which has been identified as one of the limiting factors to application scalability [13], [11].

LogGP is a communication-cost model for distributed-memory system architecture (i.e., cluster systems) [7]. Each communication operation is assigned a cost function to reflect the algorithm that implements that operation. In the LogGP model, each cost function comprises of six parameters: L (network latency), o (processing overhead), g (delay between consecutive messages), G (reciprocal of network bandwidth), P (number of processors), and m (message size). The LogGP model has been applied to evaluate the performance of various MPI implementations [22], [21].

The MPI (Message Passing Interface) specification is the de-facto standard for communications on distributed-memory systems [4]. It classifies communication operations into two types: *point-to-point* and *collective*. Point-to-point operations involve two processors, whereas collective operations involve more than two processors. Recent MPI implementations such as MVAPICH2 optimizes communication cost by supporting multiple algorithms for each type of communication operations [5]. The runtime selection of algorithms is based on a pre-defined rules on the message size and number of processors. These rules are developed based on past performance studies [9], [14].

This paper presents an investigation on the LogGP model for predicting the performance of MPI communications on InfiniBand [2], a high-speed interconnect widely used in recent peta-scale computing systems [20], [6]. Two criteria are used: communication scalability as the number of processors is increased, and the error of prediction. We use MVAPICH2 1.0, IMB 3.1, and TACC Ranger as the MPI implementation, MPI application, and testbed system, respectively. A total of five MPI operations are considered, comprising of two point-to-point operations (i.e., blocking and non-blocking) and three collective operations (i.e., barrier synchronization, all-to-all, and broadcast).

Our major observation is that the LogGP model is accurate for point-to-point operations, but it achieves varying level of accuracy for the collectives. Of the three collectives, the LogGP predicts the logarithmic scalability of `MPI_Bcast` operation with an error of up to twice of the measured performance (on 256 nodes). For the remaining operations, the LogGP can predict their scalability except for `MPI_Alltoall` with small messages ($m = 2$ bytes); however, the prediction error increases to 3.5–12 times on 256 nodes.

The remainder of this paper is organized as follows. Section 3 describes the experimental methods. Section 4.1–4.2 presents the results of MVAPICH's point-to-point and collective performance. Section 2 presents related work. Finally,

978-0-7695-3738-2/09 $25.00 © 2009 IEEE
DOI 10.1109/HPCC.2009.94

Section 5 concludes this paper.

2. Related Work

LogP [12] and its derivatives such as LogGP [7] and pLogP [16] have been proposed as performance models for evaluating the scalability of communication subsystems [8], [15]. To reduce the model's complexity, they abstract away low-level details of hardware and software components. LogP defines four parameters: L, o, g, and P where L denotes the network latency, o denotes the processor overhead per message transmission (i.e., send or receive), g denotes the minimum time interval between successive message transmissions, and P denotes the number of processors [12]. LogGP extends LogP by assuming that message size can vary and be large [7]. To this end, it introduces parameter m to denote the message size and G which is the reciprocal of network bandwidth. pLogP is the generalization of both LogP and LogGP [16].

A comparison of the accuracy of LogP, LogGP, and pLogP is presented in Pješivac-Grbović et. al. [22]. It is found that LogGP and pLogP models are accurate for various message size and number of processors. On the other hand, LogP is accurate only when the message size is close to zero byte.

MVAPICH2 is an MPI implementation optimized for InfiniBand, based on MPICH2. It supports the zero-copy optimization by exploiting the RDMA (Remote Direct Access Memory) capability of the InfiniBand. A number of performance evaluations have been conducted on MVAPICH2, and the results show its good performance on InfiniBand [17], [18]. However, the evaluations emphasize on benchmarkings instead of predictive modeling.

The LogGP performance model for MPICH2 is proposed in Pješivac-Grbović et. al. [21]. The author first identifies the algorithms that implement the MPICH2 collectives, and then develops their LogGP performance model. The performance model is further utilized to support an adaptive collective communication approach. We reuse these LogGP performance model as we find that MPICH2 and MVAPICH2 share the same implementation for the communication operations studied in this paper.

3. Methodology

Figure 1 illustrates the overall process adopted in this work. Let T_{comm} denotes the time to perform an MPI communication operation. The goal of our work is to compare and analyze the estimated T_{comm} with the measured T_{comm}. To achieve this goal, we first derive the value of system-specific parameters based on the LogMPI benchmark. Together with system-agnostic parameters, T_{comm} is then estimated according to LogGP models. The estimated performance is then compared to the performance measured by the IMB benchmark [1].

3.1. LogGP

LogGP models the communication time (T_{comm}) as a function of six parameters, i.e., $T_{comm} = f(m, L, o, g, G, P)$ [7].

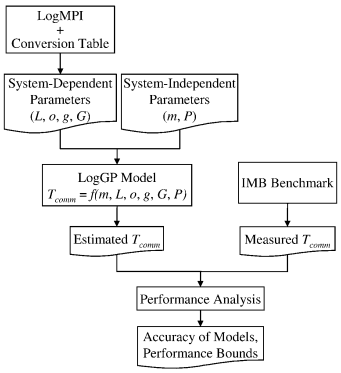


Fig. 1: Methodology of Experiments

The six parameters are described in Table 1. The parameters are further classified as *system agnostic* and *system specific*. The system-agnostic parameters, namely m and P, represents

Parameter	Description
m	Message size
L	Network latency
o	Overhead per message transmission
g	Gap between successive message transmissions
G	The reciprocal of network *bandwidth*
P	Number of processors

TABLE 1: LogGP Parameters

the application characteristics and are not affected by the underlying system architecture[1]. The system-specific parameters, namely L, o, g, and G, are affected by the system architecture (both the compute and communication subsystems). Hence, to estimate T_{comm} on a particular system, these system-specific parameters must first be measured on that system.

The four system-specific parameters for one particular system are obtained based on microbenchmarks run on that system. This work uses LogMPI as the microbenchmark [3]. However, as LogMPI implements the pLogP (parameterized LogP) model [16], the measurements by LogMPI needs to be converted to LogGP according to the conversion rules presented in Figure 2 [16]. The left-hand-side of the formulas

$$L = L' + g'(1) + os(1) - or(1) \quad (1)$$

$$o = \frac{os(1) + or(1)}{2} \quad (2)$$

$$g = g'(1) \quad (3)$$

$$G(m) = \frac{g'(m)}{m} \quad (4)$$

Fig. 2: Converting System-Specific Parameters from pLogP to LogGP

denotes the four system-specific LogGP parameters. The right-hand-side of the formulas consist of pLogP's parameters, except m, where:

[1]It is assumed that each processing elements (e.g,. a CPU core) is assigned one MPI process only.

- L' denotes network latency
- $os(m)$ denotes overhead to send an m-byte message
- $or(m)$ denotes overhead to receive an m-byte message
- $g'(m)$ denotes gap between consecutive transmissions of an m-byte message
- $G'(m)$ denotes reciprocal of network bandwidth in sending m-byte messages

3.2. MPI Performance Models based on LogGP

As illustrated in Figure 3, the cost of point-to-point communication (i.e., `MPI_Send`/`MPI_Recv` and `MPI_Isend`/`MPI_Irecv`) is modeled as $T = L+2o+(m-1)G$. Point-to-point communications can implement *eager* or *rendezvous* protocols in order to minimize the overhead of sending sending "small" messages and "large" messages, respectively. However, these protocols are not captured in the LogGP model.

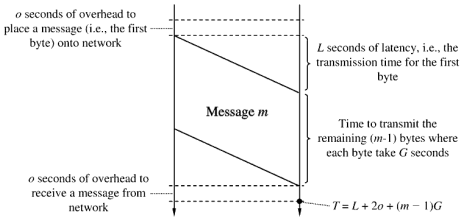


Fig. 3: Point-to-Point Communication Model

MPI collective operations can be classified into three categories, namely *synchronization* (e.g., `MPI_Barrier`), *data movement* (e.g., `MPI_Bcast`, `MPI_Alltoall`, etc.), and *collective computation* (e.g., `MPI_Allreduce`, etc.). In this paper, we consider `MPI_Barrier`, `MPI_Bcast`, and `MPI_Alltoall` on MVAPICH2 1.0. Due to the space limitation, `MPI_Allgather` is discussed in the extended version of this paper [19].

MVAPICH2 supports the collective communications based on the MPICH2's implementation. In MVAPICH2, each collective operation is implemented[2] as one or more point-to-point operations. Furthermore, MVAPICH2 supports several algorithms for each collective. At run time, a particular algorithm is selected based on the message size (m) and the number of MPI processes involved in a collective communication (P). In the following, we briefly discuss the MVAPICH2 implementation of four MPI collective operations, assuming that an *MPI intra-communicator* is used and *pof*2 denotes "power-of-two".

1) `MPI_Barrier` is implemented using on the *Hensgen's dissemination* algorithm [14].
2) `MPI_Alltoall` is implemented using three algorithms: *Bruck's k-port indexing* [9], *linear*, and *pairwise exchange*.

   The algorithm for a particular `MPI_Alltoall` invocation is chosen based on two parameters: message size (m) and number of processes (P).

m	$P < 8$	$P \geq 8$
$m \leq 256$ Bytes	Linear	Bruck
$256 < m \leq 32$ KB	Linear	Linear
$m > 32$ KB	Pairwise Exchange	Pairwise Exchange

TABLE 2: `MPI_Alltoall` in MVAPICH2 1.0

3) `MPI_Bcast` is implemented using two algorithms: *binomial tree (BT)* and *scatter/allgather*. The scatter/allgather can be further classified as *BT/RD* (binomial-tree/recursive-doubling) and *BT/R* (binomial-tree/ring).

   The algorithm for a particular `MPI_Bcast` invocation is chosen based on two parameters: message size (m) and number of processes (P).

m	$P < 8$	$P \geq 8$	
		$P = pof2$	$P \neq pof2$
$m \leq 12$ KB	BT	BT	BT
12 KB $< m \leq 512$ KB	BT	BT/RD	BT/R
$m > 512$ KB	BT	BT/R	BT/R

TABLE 3: `MPI_Bcast` in MVAPICH2 1.0

Table 4 shows the LogGP performance models for these collective communication [21], [26][3], with an additional parameter δ denoting the reciprocal of local memory bandwidth. We assume that P is a power of two and logarithmic operations are in base-two[4]. Further details on the models can be found in [21], [26]. We also show the complexity of each performance model for small messages and large messages. For small messages, the complexity is a function of P and is derived by assuming $m = 1$ and $G = 0$; thus, the complexity of LogP-based models. For large messages, the complexity depends on m in addition to P.

3.3. Experimental Testbed

Our experiments are done on TACC's Ranger computing infrastructure. At most 256 Sun Blade X6420 nodes are used. Each node has four quad-core AMD Opteron 2.0 GHz processors; thus a total of 16 cores per node. Nodes are connected via a full-CLOS InfiniBand network consisting of two Sun Magnum switches, providing a 1-GB/second point-to-point bandwidth (theoretical peak). The software stack consists of CentOS 4.4 (operating system), OFED 1.2.5.4 (InfiniBand software), MVAPICH2 1.0 (MPI implementation), and PGI 7.1 (compiler). IMB [1] is selected as the communication-intensive application which has has a negligible computational

[2] As of this date, MVAPICH2 do not yet support collective communications using hardware multicast. Instead, hardware-multicast collectives is supported in MVAPICH.

[3] For `MPI_Barrier`, the performance of Hensgen algorithm is approximated using the cost function of Bruck dissemination algorithm. The Hensgen algorithm is an extension of the Bruck algorithm; it introduces double buffering, but does not alter the sequence of point-to-point operations. Hence, their asymptotic performance is similar to each other.

[4] In addition, MVAPICH2's implementation do not internally segment messages, and hence, $n_s = 1$ for all the models.

MPI Collective	Algorithm	Performance Model
MPI_Barrier	Hensgen's Dissemination	$T = \lceil log(P) \rceil (L + o + g)$
MPI_Alltoall	Bruck's Indexing	$T = \log(P)(o + \frac{P}{2}m - 1)G + \delta\frac{P}{2}m + max\{g, L + o\} + \delta Pm$
	Linear	$T = L + 2o + (m - 1)G + 2(P - 1)G$
	Pairwise Exchange	$T = (P - 1)(L + o + (m - 1)G + g)$
MPI_Bcast	Binomial	$T = \lceil log(P) \rceil (L + 2o + (m - 1)G)$
	Scatter + Gather	$T = (\log(P) + P - 1)(L + 2o) + 2\frac{P-1}{P}mG$

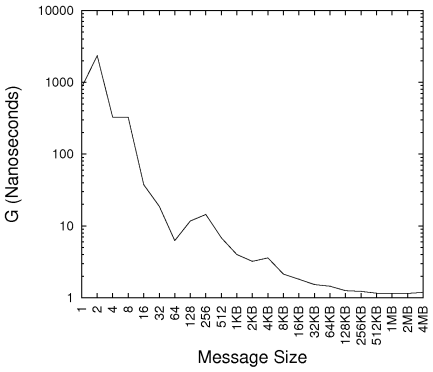
TABLE 4: Collective Communication Models for Intra Communicators

workload. In addition, to ensure that all messages go through the InfiniBand network, only one processor core per node is used.

The four system-specific parameters (i.e., L, o, g, and G) are obtained by running LogMPI benchmarks on two processors and then applying the conversion rules. The values obtained on the experimental testbed are shown in Figure 4.

Parameter	Blocking	Non-Blocking
L	1.95 μs	1.75 μs
o	0.65 μs	0.85 μs
g	0.65 μs	0.85 μs

(a) L, o, and g



(b) G for Various Message Sizes

Fig. 4: Values of L, o, g, and G

4. Experimental Results

This sections present and discuss the performance of five MPI operations based on measurements and estimations. The five operations are *blocking point-to-point*, *non-blocking point-to-point*, *barrier synchronization*, *all-to-all*, and *broadcast*.

The *communication performance* is quantified as the time to complete an operation for message size m and number of processors P. The measured performance is the *average* performance of each operations measured by the IMB benchmark; the number of measurements is based on the default IMB setting:

1) For $0 < m < 32$ KB, the number of measurements per operation is 1,000 times.
2) For 64 KB $< m <$ 4096 KB, the number of measurements per operation is $640/2^{\lfloor \log_2 m \rfloor - 16}$ times.

On the other hand, the estimated performance is calculated according to the cost functions assigned by the LogGP model to each MPI operation. All results will be plotted in the log-log scale (i.e., both x-axis and y-axis are in power-of-two).

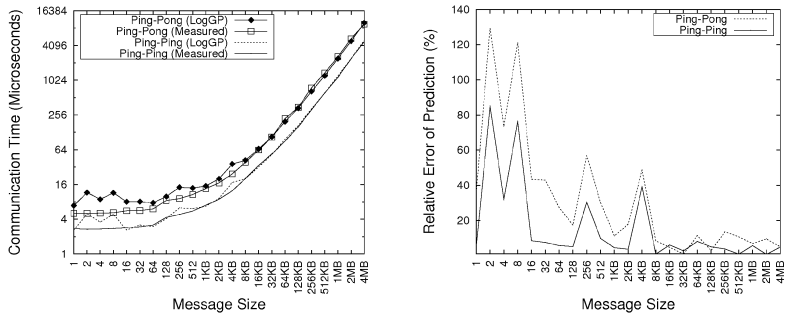
4.1. Point-to-Point Communication

Figure 5(a)–5(b) compares the measured and predicted performance, and the error of performance prediction. As shown in Figure 5(a), the LogGP model accurately estimates the scalability of MVAPICH2's point-to-point communication. However, the relative error is higher for small messages ($m \leq 8$) and is most apparent on the blocking `MPI_Isend`/`MPI_Irecv` results. This suggests that the value of G when measured on $m \leq 8$ bytes, is less accurate than for $m \geq 8$ bytes, because the values are dilated by random system jitters during the LogMPI measurements. For $m > 8$ bytes, the average of relative error ($E\%$) is less than 20%, while the absolute error (E) averages at about 11–44 times L (Table 5(c)).

4.2. Collective Communication

In this section, we present the experimental results of three collective communications in MVAPICH2 and discuss their scalability as P is increased. The accuracy of the LogGP model is based on the predicted asymptotic scalability up to 256 nodes and the prediction error on 256 nodes. Two message sizes are considered, namely *small* ($m = 2$ bytes) and *large* ($m = 1$ MB). An exception is for the barrier synchronization that use only small messages with $m = 0$ byte. The selected message sizes are based on the study by Shalf et. al. [23]. Due to the space limitation, the results on *medium* messages ($m = 2$ KB) is presented in the extended version of this paper [19]. In the LogGP model, it is assumed that $G = 0$ for $m \leq 8$, based on our finding in Section 4.1. In addition, $\delta = 0.125 * 10^{-9}$; this is derived by assuming the memory bandwidth to be 8 GB/second which is 80% of the peak per-socket memory bandwidth in quad-core Opteron systems.

4.2.1. Small Messages. Figure 6 shows that the logarithmic scalability of `MPI_Bcast` for $m = 2$ bytes (i.e., recursive-doubling algorithm) can be accurately predicted by the LogGP model, and the prediction error on 256 nodes is twice of the measured performance. For `MPI_Barrier` (binomial algorithm), their logarithmic scalability is still predictable using LogGP (bar some random noises); however, the prediction error on 256 nodes increases to 4 times of the measured performance. For `MPI_Alltoall`, its linear scalability is

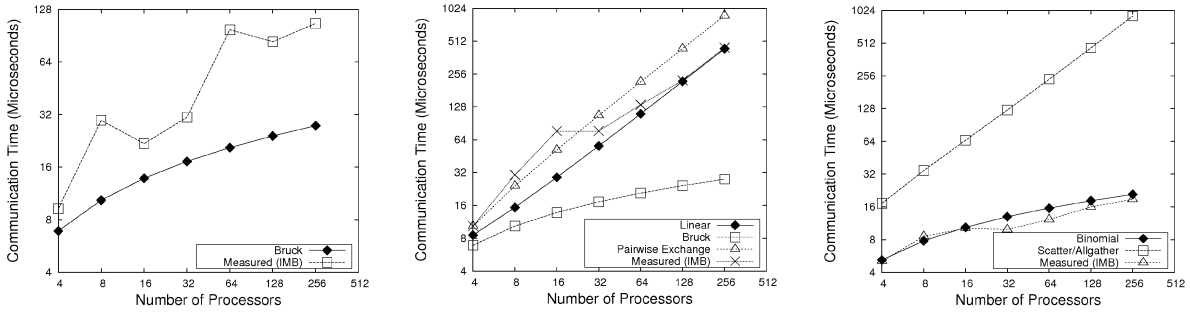


(a) Estimated and Measured Performance (b) Relative Errors

Operations	E%	E
Blocking	19%	78 μs
Non-Blocking	8%	19 μs

(c) Mean of Errors

Fig. 5: Point-to-Point Results



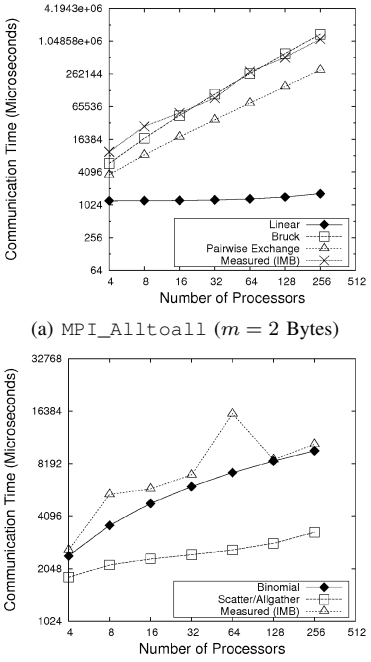
(a) `MPI_Barrier` ($m = 0$ Byte) (b) `MPI_Alltoall` ($m = 2$ Bytes) (c) `MPI_Bcast` ($m = 2$ Bytes)

Fig. 6: Collective Communications on Small Messages

predicted as logarithmic scalability instead. A further discussion is presented below.

For `MPI_Alltoall` (Figure 6(b)), the measured performance shows two distinct trends, one for $P < 32$ and another for $P \geq 32$. However, this result does not conform to the implementation in MVAPICH2 1.0. Firstly, the shift in our results occurs at $P = 32$, though MVAPICH2 switches from the linear algorithm to Bruck's k-port indexing at $P = 8$. Since the message size is constant at $m = 2$ bytes, the root cause of this issue might not be due to whether the underlying `MPI_Sendrecv` is eager or rendezvous. However, a more thorough investigation is required to verify this claim. Secondly, the LogGP model predicts logarithmic scalability for the Bruck's k-port indexing algorithm which is used for $P \geq 8$ and $m \leq 256$ bytes. However, our result shows a linear scalability for $8 \leq P < 32$ and $P \geq 32$.

4.2.2. Large Messages. As shown in Figure 7, the LogGP model accurately predicts `MPI_Bcast` according to the binomial-tree algorithm for 256 nodes. Bar the spike when $P = 64$, the results for `MPI_Bcast` shows that the performance for $P \geq 8$ and $m = 1$ MB resembles that of the binomial-tree algorithm; this is consistent with the implementation in MVAPICH2 1.0. It also predicts the linear scalability of `MPI_Alltoall` (pairwise exchange) with an error of 4 times of measured performance on 256 nodes.



(a) `MPI_Alltoall` ($m = 2$ Bytes)

(b) `MPI_Bcast` ($m = 2$ Bytes)

Fig. 7: Collective Performance on Large Messages ($m = 1$ MB)

5. Conclusions

This paper presents a study on the accuracy of LogGP in modeling message-passing performance in the context of mod-

ern interconnect technology. Based on the experiment on 256 nodes with InfiniBand interconnect, we first observe that the LogGP model is accurate for point-to-point communications (with $G = 0$ for small messages such as $m < 16$ bytes). Secondly, our results show a gap between the predicted and measured collective communication cost. For `MPI_Bcast`, the LogGP model is able to predict its scalability up to 256 nodes, and the prediction error is at most a factor of two on 256 nodes. For the remaining collectives, i.e., `MPI_Barrier` and `MPI_Alltoall`, their scalability can be predicted by LogGP, except for `MPI_Alltoall` on small messages ($m = 2$ bytes) where the linear scalability is predicted as logarithmic scalability. However, the prediction errors for 256 nodes are 3.5–12 times the measured performance. Our ongoing work includes identifying the root cause of the prediction error, developing a more accurate modeling technique, and to re-evaluate the performance of various message-passing communication algorithms on recent and future technologies.

Acknowledgment

This work is supported in part by Sun Microsystems, Inc. and the National University of Singapore under grant number R252-000-298-720.

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper.

References

[1] IMB 3.1. http://www3.intel.com/cd/software/products/asmo-na/eng/219848.htm.

[2] InfiniBand Trade Association. http://www.infinibandta.org.

[3] LogMPI. ftp://ftp.cs.vu.nl/pub/kielmann/logp_mpi.tar.gz.

[4] MPI forum. http://www.mpi-forum.org.

[5] MVAPICH. http://mvapich.cse.ohio-state.edu.

[6] TACC Ranger. http://www.tacc.utexas.edu/resources/hpcsystems/#constellation.

[7] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. J. Scheiman. LogGP: Incorporating long messages into the LogP model for parallel computation. *Journal of Parallel and Distributed Computing*, 44(1):71–79, Aug. 1997.

[8] C. Bell, D. Bonachea, Y. Cote, J. Duell, P. Hargrove, P. Husbands, C. Iancu, M. L. Welcome, and K. A. Yelick. An evaluation of current high-performance networks. In *Proc. of the 17th IEEE Intl. Works. on Parallel and Distributed Processing (IPDPS)*, page 28, Apr. 2003.

[9] J. Bruck, C.-T. Ho, S. Kipnis, E. Upfal, and D. Weathersby. Efficient algorithms for all-to-all communications in multiport message-passing systems. *IEEE Trans. on Parallel and Distributed Systems*, 8(11):1143–1156, Nov. 1997.

[10] M. Casas, R. Badia, and J. Labarta. Automatic analysis of speedup of MPI applications. In *Proc. of the 22nd Intl. Conf. on Supercomputing (ICS)*, pages 349–358, June 2008.

[11] M. Crovella, R. Bianchini, T. LeBlanc, E. Markatos, and R. Wisniewski. Using communication-to-computation ratio in parallel program design and performance prediction. In *Proc. of the 4th IEEE Symp. on Parallel and Distributed Computing Processing (IPDPS)*, pages 238–245, Dec. 1992.

[12] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a realistic model of parallel computation. In *Proc. of the 4th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPOPP)*, pages 1–12, May 1993.

[13] D. L. Eager, J. Zahorjan, and E. D. Lazowska. Speedup versus efficiency in parallel systems. *IEEE Trans. on Computers*, 38(3):408–423, Mar. 1989.

[14] D. Hensgen, R. Finkel, and U. Manber. Two algorithms for barrier synchronization. *Intl. Journal of Parallel Computing*, 17(1):1–17, Feb. 1988.

[15] T. Hoefler, L. Cerquetti, T. Mehlan, F. Mietke, and W. Rehm. A practical approach to the rating of barrier algorithms using the LogP model and Open MPI. In *Proc. of the 34th Intl. Conf. on Parallel Processing Works. (ICPP)*, pages 562–569, June 2005.

[16] T. Kielmann, H. E. Bal, and K. Verstoep. Fast measurement of LogP parameters for message passing platforms. In *Proc. of the 14th IEEE Intl. Works. on Parallel and Distributed Processing (IPDPS)*, pages 1176–1183, May 2000.

[17] R. Kumar, A. Mamidala, and D. K. Panda. Scaling alltoall collective on multi-core systems. In *Proc. of the 2008 Works. on Communication Architecture for Clusters (CAC 2008), in conjunction with the 22nd Intl. Parallel and Distributed Processing Symp. (IPDPS)*, Apr. 2008.

[18] A. R. Mamidala, R. Kumar, D. De, and D. K. Panda. MPI collectives on modern multicore clusters: Performance optimizations and communication characteristics. In *Proc. of the 8th IEEE Intl. Symp. on Cluster Computing and the Grid (CCGrid)*, May 2008.

[19] V. March, V. Murali, Y. M. Teo, S. See, and J. T. Himer. Towards predictive modeling of message-passing communication. Technical report, Dept. of Computer Science, National University of Singapore (NUS), 2009.

[20] S. Matsuoka. The road to TSUBAME and beyond. *High Performance Computing on Vector Systems 2007*, pages 265–267, Oct. 2007.

[21] J. Pješivac-Grbović. *Towards Automatic and Adaptive Optimizations of MPI Collective Operations*. PhD thesis, Dept. of Computer Science, The University of Tennessee, Dec. 2007.

[22] J. Pješivac-Grbović, T. Angskun, G. Bosilca, G. E. Fagg, E. Gabriel, and J. Dongarra. Performance analysis of mpi collective operations. In *Proc. of the 19th IEEE Intl. Works. on Parallel and Distributed Processing (IPDPS)*, page 28, Apr. 2005.

[23] J. Shalf, S. Kamil, L. Oliker, and D. Skinner. Analyzing ultra-scale application communication requirements for a reconfigurable hybrid interconnect. In *Proc. of the 2005 ACM/IEEE Conf. on Supercomputing (SC)*, page 17, Nov. 2005.

[24] I. Sharapov, R. Kroeger, G. Delamarter, R. Cheveresan, and M. Ramsay. A case study in top-down performance estimation for a large-scale parallel application. In *Proc. of the 11th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPOPP)*, pages 81–89, Mar. 2006.

[25] R. Susukita, H. Ando, M. Aoyagi, H. Honda, Y. Inadomi, K. Inoue, S. Ishizuki, Y. Kimura, H. Komatsu, M. Kurokawa, K. J. Murakami, H. Shibamura, S. Yamamura, and Y. Yu. Performance prediction of large-scale parallel system and application using macro-level simulation. In *Proc. of the 2008 ACM/IEEE Conf. on Supercomputing (SC)*, page 20, Nov. 2008.

[26] R. Thakur, R. Rabenseifner, and W. Gropp. Optimization of collective communication operations in MPICH. *Intl. Journal of High Performance Computing Applications*, 19(1):49–66, Feb. 2005.