Modeling MPI Communication Performance on SMP Nodes: Is it Time to Retire the Ping Pong Test

William Gropp
Department of Computer Science
University of Illinois at Urbana-Champaign
wgropp@illinois.edu

Luke N. Olson
Department of Computer Science
University of Illinois at Urbana-Champaign
lukeo@illinois.edu

Philipp Samfass
Department of Computer Science
University of Illinois at Urbana-Champaign
samfass2@illinois.edu

ABSTRACT

The "postal" model of communication [3, 8] $T = \alpha + \beta n$, for sending n bytes of data between two processes with latency α and bandwidth $1/\beta$, is perhaps the most commonly used communication performance model in parallel computing. This performance model is often used in developing and evaluating parallel algorithms in high-performance computing, and was an effective model when it was first proposed. Consequently, numerous tests of "ping pong" communication have been developed in order to measure these parameters in the model. However, with the advent of multicore nodes connected to a single (or a few) network interfaces, the model has become a poor match to modern hardware. In this paper, we show a simple three-parameter model that better captures the behavior of current parallel computing systems, and demonstrate its accuracy on several systems. In support of this model, which we call the max-rate model, we have developed an open source benchmark[1] that can be used to determine the model parameters.

CCS Concepts

•**Theory of computation** → **Parallel computing models;** *Parallel algorithms;* •**Computer systems organization** → *Multicore architectures;*

Keywords

bandwidth saturation, benchmark, communication, multicore, parallel computing, performance model, symmetric multiprocessor cluster, ping pong

1. INTRODUCTION

One of the advantages of the message-passing programming model is that there are relatively simple performance models that can (and are) used to design algorithms and understand performance issues in applications. One of the earliest, sometimes called the postal model [3, 8][2], represents the communication time to send n bytes as

$$T = \alpha + \beta n, \tag{1}$$

where the term α denotes the total *latency* and β the inverse of the asymptotic *bandwidth* (for arbitrarily large n, measured in seconds per byte). While numerous other performance models have been proposed [16] and have found some use, the postal model remains the most common performance model used to analyze message-passing programs.

Consequently, many benchmarks measure α and β by sending a message to another process, followed by sending the message back. This is called a *ping pong* benchmark, and is widely used to evaluate MPI implementation performance. For many years, these benchmarks, and the postal model, have been effective ways to model and predict performance of MPI applications.

However, with the advent of multicore processors and nodes, the postal model and the ping pong benchmarks are no longer useful without some modification, as we show in this paper. Instead, additional information is needed to take into account the fact that all systems that we study do not provide the full bandwidth to all processes in a multicore node.

This paper describes the limitations of the postal model in Section 2. In Section 3 we propose a simple alternative and show measurements on several systems in Section 4, each with a different node and/or network architecture, highlighting the superiority of our new model. We close with our recommendations and conclusions in Section 5.

1.1 Related Work

There are a number of alternative performance models to the postal model. The LogP [5] model separates hardware latency from software overhead and is useful in modeling short messages in an effort to measure the impact of software overhead. The LogGP [2] model addresses the issue of long messages that is not handled in the LogP model. Hop bytes or hop count [1] includes an additional term that models the number of links (hops) over which a message travels. Other models address specific network capabilities, while simulation is sometimes used for the more difficult is-

[1] https://bitbucket.org/william_gropp/baseenv

EuroMPI '16, September 25-28, 2016, Edinburgh, United Kingdom

DOI: http://dx.doi.org/10.1145/2966884.2966919

[2] The term "postal model" was first introduced in [3] but the notion of modeling operation time in this way for vector and parallel computers was described in detail in [8]; see, e.g., equation 1.4b.

sue of contention between messages in a network.

There are also many benchmarks that measure ping pong performance. Some of the best known include mpptest [6], SkaMPI [17], Netgauge [9], the Intel IMB [11], OSU Micro-Benchmarks [15], and the Phloem MPI Benchmarks [12]. A ping pong test is also included in the HPC Challenge [13] benchmarks. In addition, a discussion of MPI performance on ccNUMA nodes appears in [14]. A comparison of some of the MPI benchmarks [7], most of which target single processor (or core) nodes, shows a mismatch between the design of these benchmarks and the behavior measured on SMP nodes. An excellent discussion of the complexities in measuring parallel performance is in [10].

2. WHAT'S WRONG WITH PING PONG

The ping pong benchmark and the communication performance parameters that it measures are widely used to compare MPI implementations and, perhaps more importantly, to predict the performance of an MPI application, including giving insight into the design of algorithms that use MPI. The associated performance model, sometimes called the *postal model* [8], is a simple two-parameter model often written in (1). In this section, we explain why this performance model no longer matches the behavior of many systems.

As an example, consider a halo exchange, which is a common communication pattern that occurs in both the explicit and implicit discretization of partial differential equations. A typical implementation has each MPI process communicating with multiple processes on different nodes, sends n bytes to each of E processes. Using the postal model to estimate the time that the exchange takes yields[3] $T = E(\alpha + \beta n)$. But measurement on many systems show much lower performance. For example, on a Cray system with XE6 nodes, the sustained halo exchange rate on 1024 processes with 2048 double floating point numbers is approximately 380 MB/sec. Yet, using a ping pong test (mpptest from [6]), we measure $\alpha = 1.6e-6$ and $\beta = 1.7e-10$, which corresponds to a sustained bandwidth of 5.9 GB/sec using the postal model. *Why is the postal model so inaccurate?*

As we show in the following sections, the key problem with the postal model is that it does not account for the interface between the node. In addition, the network cannot sustain communication at a rate of a single process node (which is the case in the ping pong measurement).

Another well-known, but often unconsidered issue, is that MPI implementations typically use different methods depending on the message length. It is common for short messages to be sent "eagerly" while longer messages depend on a rendezvous. Consequently, the resulting bandwidth may be different; in addition, the rendezvous approach requires more control messages, adding to the latency (the α term in (1)). When designing algorithms and implementations, understanding the different regimes is important; indeed, many MPI implementations allow user control (within limits) of the size of messages that are sent eagerly.

There are other effects that are also important that we do not consider in the proposed model. These include network distance — e.g., multiple hops over the network, sometimes modeled by adding a "hop-count" [1], different network bandwidths in some high-radix, switched networks such as a mix of electrical and optical links depending on the distance and route [4], differences between inter-node and intra-node communication, and network contention caused by messages sharing the same links. All of these effects are usually much smaller than the impact of multiple MPI processes on the same node communicating at the same time, which can be $10\times$ what the postal model predicts.

[3]On systems such as the IBM Blue Gene, there are multiple independent links, so if all links are used, the time estimate divides this by the number of links used.

3. MODELING MPI ON SMPS

The problem with the ping pong benchmark is that it assumes that the communication performance between two MPI processes is independent of the execution of other MPI processes. The utility of the model depends on the correctness of this assumption, since in a typical MPI application, communication often happens in phases during which most if not all processes are communicating with another process.

Thus, we refine the model in two ways:

1. Recognize the eager/rendezvous threshold (potentially including "short" threshold for very short messages), and
2. Take into account the limits of bandwidth into and out of a node.

The first means that there are essentially two models (three in the case of a separate short protocol) and that each should be measured separately. The second implies that these terms establish a maximum communication rate for all processes on a node.

To model the maximum communication rate we modify the two-parameter postal model, which is given as

$$T = \alpha + n\beta. \tag{2}$$

We consider the maximum rate at which k processes can send data out of a node. If the processes do not limit each other, the aggregate rate is just kR_C, where R_C is the rate that each process can achieve in sending or receiving a message. Now, if we also take into account that the maximum rate that data can leave (or enter) the node and enter (respectively exit) the network (the *injection bandwidth*) is R_N, then the maximum bandwidth for k simultaneously communicating processes on a node is $\min(R_N, kR_C)$. Using this as the maximum asymptotic bandwidth, we modify the postal model to be

$$T = \alpha + kn/\min(R_N, kR_C), \tag{3}$$

since kn bytes are being sent.

A refinement of this model,

$$T = \alpha + kn/\min\left(R_N, R_{C_b} + (k-1)R_{C_i}\right), \tag{4}$$

recognizes that the additional bandwidth achievable by processes after the first may be different (typically lower) than the bandwidth that a single process can sustain. However, as we show, this refinement offers at best a minor improvement in accuracy. As a result, we recommend the use of the simpler three-parameter model given in (3).

It is important to note that for large R_N, each of (3) and (4) reduces to $T = \alpha + n/R_c$, or $\beta = 1/R_c$, which is the typical postal model or ping pong test. Conversely, if $R_N = R_c$, that is, the rate that any one process is able to send data is

the speed at which data can be injected into the network, then $T = \alpha + kn/R_n$. That is, the time for large messages is (asymptotically) proportional to the number of concurrent sending processes on each node.

An improved model recognizes the impact of the message overhead on the peak rate, and can be formulated as follows: The sustained rate is the minimum of R_N, the peak rate of the NIC, and the rate that is sustained by having each process send at a rate of R_c, but including the latency term. That is,

$$R = \frac{kn}{T} = \min\left(R_N, \frac{kn}{\alpha + n/R_c}\right).$$

Solving for T, this performance model is

$$T = \frac{kn}{\min\left(R_N, \frac{kn}{\alpha + n/R_c}\right)}. \quad (5)$$

For n large, the α term becomes negligible and

$$T \approx \frac{kn}{\min\left(R_N, \frac{kn}{n/R_c}\right)} = \frac{kn}{\min(R_N, kR_c)}.$$

Similarly, for n small and $\alpha \gg n/R_c$, we have

$$T \approx \frac{kn}{\frac{kn}{\alpha + n/R_c}} = \alpha + n/R_c.$$

While Equation (5) can be a more accurate model (as we show in Figure 5 for the IBM Blue Gene/Q), the three-parameter model in (3) is nearly as effective and is easier to work with. As a result, we use this simpler equation in our comparisons.

3.1 Consequences

Our new model suggests several changes in how algorithms are designed. Under the postal model in (1), it often makes sense to implement a halo exchange as

```
MPI_Isend for each edge
MPI_Irecv for each edge
MPI_Waitall
```

However, the new model identifies the inefficiencies. Specifically, in the postal two-parameter model, for E edges of length n, $T = E(\alpha + \beta N)$. In contrast, in the new model, $T = E(\alpha + kN/\min(R_N, kR_c))$ for k MPI processes on each node, where $\beta = 1/R_c$ as shown above. If we assume N is large enough to ignore the latency (α) term and that k is large enough that the injection bandwidth dominates ($R_N < kR_c$), then the ratio of the time estimates for the postal model (2) and the new max-rate model (3) is

$$\frac{T_{pingpong}}{T_{new}} \approx \frac{\beta}{k/R_N}. \quad (6)$$

For R_N similar to R_c, this implies that the communication time is approximately k times the predicted postal model time. This matches what we observe in practice, for example on Cray systems (cf. Section 4).

This modeling motivates a different approach to the halo exchange. A relatively simple change is to overlap communication with computation, as in

```
MPI_Isend for each edge
MPI_Irecv for each edge
compute on interior
MPI_Waitall
compute on edges
```

This requires that the communication time exceeds the time in the "compute on interior" step, and that communication time is given by the new model. An alternative is to use a smaller number of processes to communicate; this strategy also requires a more sophisticated load balancing strategy, since different processes will have varying amounts of work.

4. EXPERIMENTAL RESULTS

In this section we develop a benchmark and derive measurements of internode communication for several, current architectures. The measurements are used to determine the parameters in our proposed model. The results also highlight the discussion presented in Section 2 on the limitations of the traditional two-parameter model.

In the following, we first describe the experimental setup and present data obtained on Blue Waters[4], which uses a Cray Gemini interconnect. We fit our proposed model to the data, using least-squares, and highlight the value in using an extended set of parameters in the model in the case of multicore nodes. We then summarize the experiments on Mira[5], a Blue Gene/Q system, and two clusters, one using a Cisco UCS Fabric Interconnect and one with InfiniBand networking.

4.1 Experimental Setup

Based on the hardware topology for each system in our benchmark, pairs of communicating MPI ranks on separate nodes are determined to communicate according to a blocking ping pong exchange pattern. We measure a loop over multiple ping pong exchanges using `MPI_Wtime()`. In the following figures, we use the max-reduce time over all participating MPI processes to determine the aggregate effective bandwidth defined as $R = \frac{kn}{T}$. In order to test the hypothesis of this paper, we vary both the number of communicating pairs of processes and the message sizes.

To determine the model parameters in the two, three, and four parameter models (equations (2), (3), (4), respectively) we use a non-linear least squares method[6]. Because of the wide range in message sizes and hence communication times in our experiments, instead of the standard error function, we minimize the weighted variant $\sum_i (y_i - f(\vec{p}, k_i, n_i))^2/n_i$, where f is the model function, $\vec{p}$ are the model parameters, k_i is the number of communicating pairs and n_i the message size for the ith data point. In the following experiments we compute the relative error in the model fit as a function of the message size and the number of communicating pairs (cf. Figure 2).

In all except for one of the tested systems (Taub Cluster with InfiniBand) different communication phases (short, eager and rendezvous) were apparent, so we fitted them separately[7] since a single fit over all message sizes would not

[4]https://bluewaters.ncsa.illinois.edu/blue-waters

[5]https://www.alcf.anl.gov/mira

[6]A variant of the Levenberg-Marquardt algorithm is used, as implemented in the Python library `scipy.optimize.leastsq`

[7]The `buflimit` routine from mpptest is used to compute thresholds: http://www.mcs.anl.gov/research/projects/mpi/mpptest/; also included in

adequately handle the bandwidths for the different protocols. Both the postal model, (2), and the max-rate models, (3) and (4), effectively capture the eager and short phases. However, in the case of large message sizes (rendezvous), the postal model does not fit the trends observed in the measured data. As we will show in detail for Blue Waters, a two-parameter ping pong fit will either significantly over- or underestimate the actual aggregate bandwidth due to its assumption of a fixed bandwidth independent of the number of communicating pairs. In contrast, we also observe that the max-rate three-parameter and four-parameter models succeed to identify the upper limit in the available aggregate bandwidth, correctly scaling in the case of concurrent communication. In the following section we show that both max-rate models yield similar results, although the four-parameter model is more accurate in the case of two communicating pairs since it assumes a base bandwidth.

4.1.1 Blue Waters Cray XE6

In this first test, the benchmark is compiled with the Cray C compiler, v 8.4.6, with optimization flag `-O3`. The tests are executed on 8 Cray XE6 nodes with 16 MPI processes mapped round robin *by socket*. We present the communication results for two neighboring nodes (in the z-direction, resulting in different Gemini hubs) in the 3-dimensional torus topology. In addition, we set the short-eager threshold at 32 bytes and the eager-rendezvous threshold at 1024 bytes. In order to test whether the postal model is capable of predicting the available bandwidth for our data, we estimate α and β by computing fits for three different data sets: a single pair of communicating processes, all 16 communicating pairs, and the complete set of observed data (1 up to 16 communicating pairs).

In Figure 1 we present both the measured aggregate bandwidth (Figure 1a) as well as the fit of this data to each of the models presented in the previous section. The error in the model fit is given in Figure 2, where it is important to note that different scales are used in each figure in order to identify the locations of the largest error.

In both the measured data and modeled data of Figure 1, the short-eager and eager-rendezvous protocol thresholds are clearly identified. We also note that the postal model does not adequately capture the bandwidth, particularly at large message sizes. Indeed, on closer inspection of the errors in Figure 2, we note that the max-rate, three-parameter and extended max-rate four-parameter models fit with a maximum relative error of 0.24 and 0.18, respectively. In contrast, the postal, two-parameter models given in Figures 2c, 2d, and 2e yields larger errors (0.88, 7.26, and 3.49), resulting in a misrepresentation of the bandwidth by an order-of-magnitude in many cases.

The extended max-rate model yields small relative errors for the bulk part of the data. Yet, the largest errors of our model are observed for the case of one pair of communicating processes in the short and eager regimes and for a large number of active process pairs right at the eager-rendezvous threshold. The eager-rendezvous threshold varies between systems and can have a large impact on the interpretation of the ping pong results. For example, Figure 7 shows a large threshold of around 25Kb; identifying the threshold is straightforward, but is an important aspect to obtaining an accurate model. Furthermore, we see that the improved four-parameter max-rate model (4) results in only modest improvements for large message sizes. This suggests that the simpler, three-parameter max-rate model is sufficient to identify high-quality performance expectations in practice.

The model parameters for our extended max-rate model are detailed in Table 1. For the short phase, the resulting R_N is large enough to not be a limiting factor in the model (we thus label it ∞). As a result, the performance model reduces to a modified form of the extended max-rate model (4):

$$T = \alpha + kn/(R_{C_b} + (k-1)R_{C_i}).$$

Likewise, since the negative parameters values are likely an artifact, the model is further reduced in this situation to a two-parameter model:

$$T = \alpha + kn\beta,$$

where $\beta = 1/R_{C_b}$. This resembles the standard postal model (2), but assumes communication dependent on k, the number of processing elements. For the eager phase, parameter R_N is again determined large, thus not impacting the model. In this case, the message size is not large enough to saturate the NIC for 16 simultaneously communicating pairs. However, for the rendezvous phase, when message sizes are larger, saturation is visible in the data. Accordingly, our max-rate model yields an R_N which limits the peak performance for large message sizes when more cores communicate.

As a summary, the (total) relative errors in the model fits are given in Table 2 for each communication protocol. We see that uniformly the max-rate models more accurately describe the aggregate bandwidth in the system.

4.2 Results on Other Systems

In this section we highlight the robustness of the new max-rate model by executing the benchmark and fitting the data on a series of machines with different network interconnects.

Illinois Taub Cluster with Infiniband

The Illinois Taub Cluster[8] is a QDR InfiniBand based system of dual socket nodes, each with Intel X5650 six-core processors. The results of the benchmark and model are shown in Figure 3, where we consider two network protocols: TCP using MPICH and InfiniBand using MVAPICH2. It is important to note that these tests were run during heavy system use, leading to noisy InfiniBand results; consequently we present the highest performing data in these runs. In the case of both TCP and InfiniBand we use a single fit of the model to obtain the max-rate model parameters in (3). The results in Figure 3 show that the max-rate performance model captures the bandwidth trends in multicore communication across network protocols and in the presence of data irregularity (e.g. in the case of TCP).

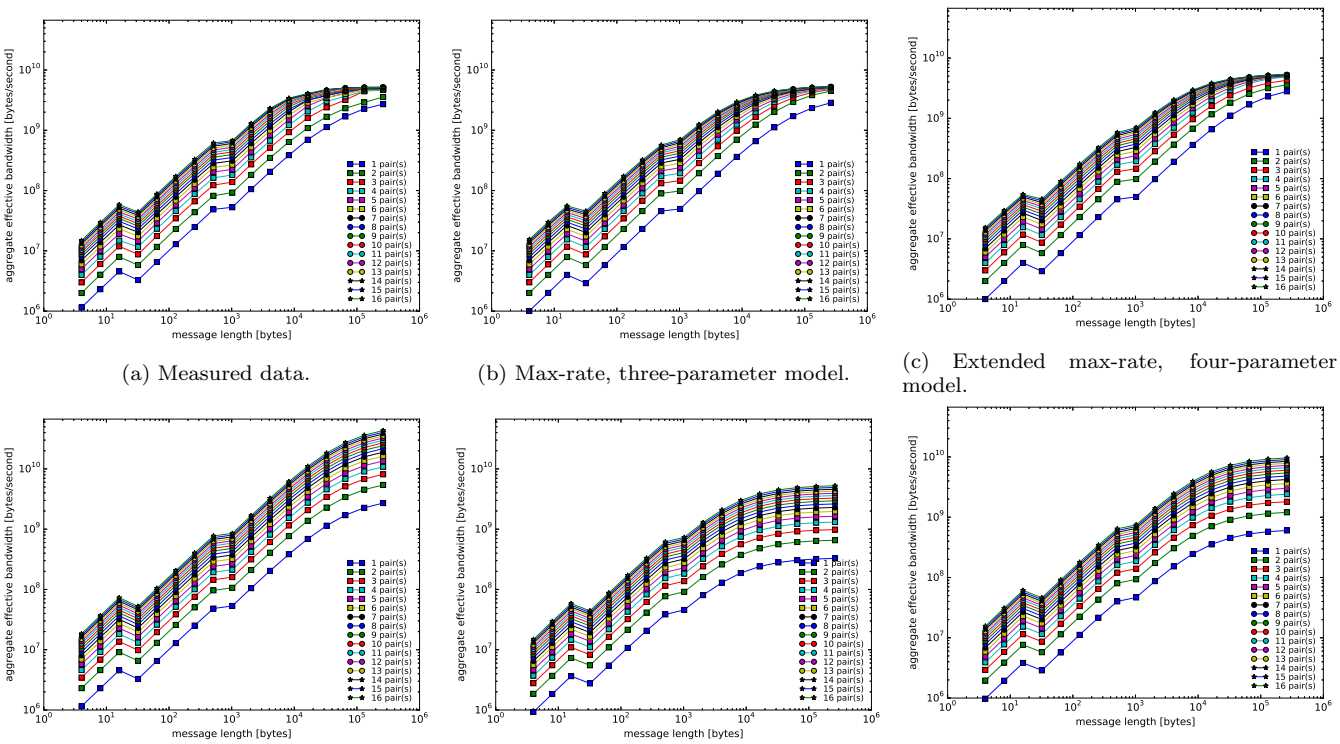
Mira IBM Blue Gene/Q

Mira, a 10-petaflop IBM Blue Gene/Q machine at Argonne National Laboratory, yields two distinct communication phases, eager and rendezvous protocols, as shown in Figure 4a. As a result, we fit the three-parameter model (3) to each of these regions.

Figure 4 underscores that our model is qualitatively accurate, showing the saturation of the available bandwidth from

https://bitbucket.org/william_gropp/baseenv

[8]https://campuscluster.illinois.edu/hardware/#taub



(a) Measured data.

(b) Max-rate, three-parameter model.

(c) Extended max-rate, four-parameter model.

(d) Postal, two-parameter model (1 pair).

(e) Postal, two-parameter model (16 pairs).

(f) Postal, two-parameter model (1–16 pairs).

Figure 1: Aggregate effective bandwidth and fitted models on Blue Waters (Cray XE6) with 1 (bottom) to 16 (top) communicating pairs.

Table 1: Max-rate model parameters (4) for Blue Waters (Cray XE6).

Phase	α [sec]	R_N [bytes/sec]	R_{C_b} [bytes/sec]	R_{C_i} [bytes/sec]
short	4.0×10^{-6}	∞	6.3×10^{8}	-1.8×10^{7}
eager	1.1×10^{-5}	∞	1.7×10^{9}	6.2×10^{7}
rendezvous	2.0×10^{-5}	5.5×10^{9}	3.6×10^{9}	6.1×10^{8}

the node into the network. However, the model is not able to reproduce the sharp ridge demonstrated in the measured data of Figure 4a. Instead, Figure 5 shows that the modified model (5) gives a closer fit in this particular case. Nevertheless, we recommend using the max-rate model in general due to its ease of use and the flexibility in generalizing to other systems.

Arcetri Cisco Cluster

The Arcetri UCS Balanced Technical Computing Cluster at Cisco Systems employs a Cisco Fabric Interconnect and dual 12-core Intel Haswell 2680 chips per node. This allows us to scale the number of communicating pairs to 24. The results are consistent with the other machines: the max-rate, three-parameter model effectively captures the behavior of the bandwidth for multiple processing elements. In this run in particular, the eager-rendezvous threshold is high (cf. Figure 7), yet fitting the model to each region identifies the salient features of the performance.

5. CONCLUSIONS

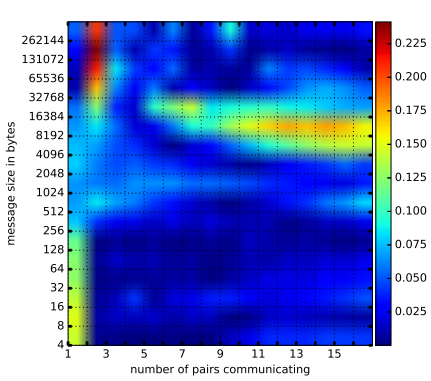
We have shown that the postal model fails to capture the communication behavior of multicore nodes and that a simple addition to the model provides a much more accurate performance model. While the proposed model has its own limitations (for example, it doesn't include effects of shared caches between cores or interactions between different chips in a multi-chip node), it is successful in providing insight into the performance of message-passing applications and to guidance into the design of algorithms.

We believe that it is time to retire the postal model and use the proposed three-parameter model,

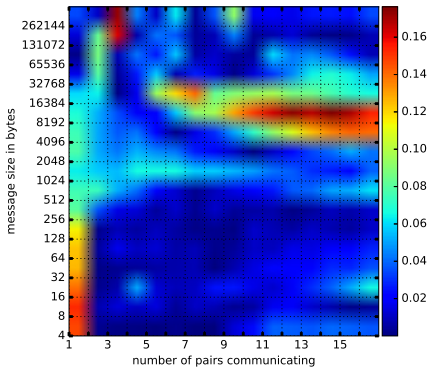
$$T = \alpha + kn/\min(R_N, kR_c)$$

instead. Further, vendors can easily provide this information—they often already provide α and R_C by using one of the existing standard benchmarks; R_N, the maximum rate that can be sustained by the network interface is also often described in the vendor literature, and as we have shown, can be reliably measured by a simple benchmark, which we have provided.

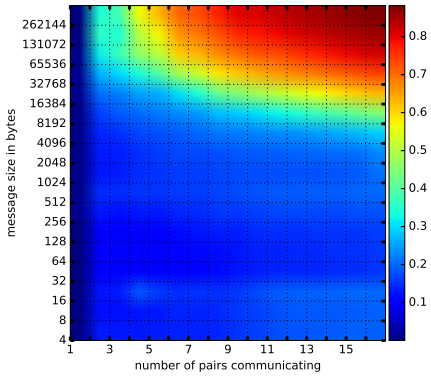
We also believe that eager/rendezvous thresholds should



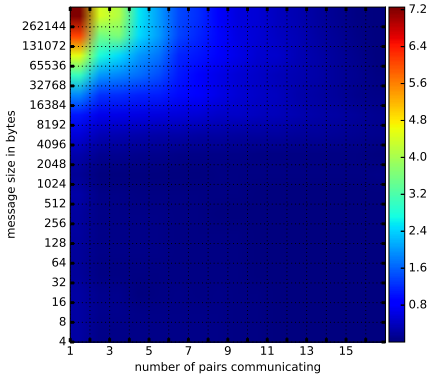
(a) Max-rate, three-parameter model.



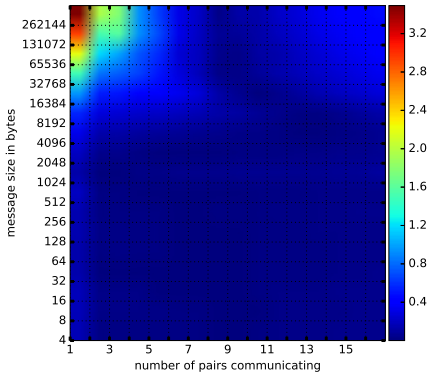
(b) Extended max-rate, four-parameter model.



(c) Postal, two-parameter model (1 pair).



(d) Postal, two-parameter model (16 pairs).

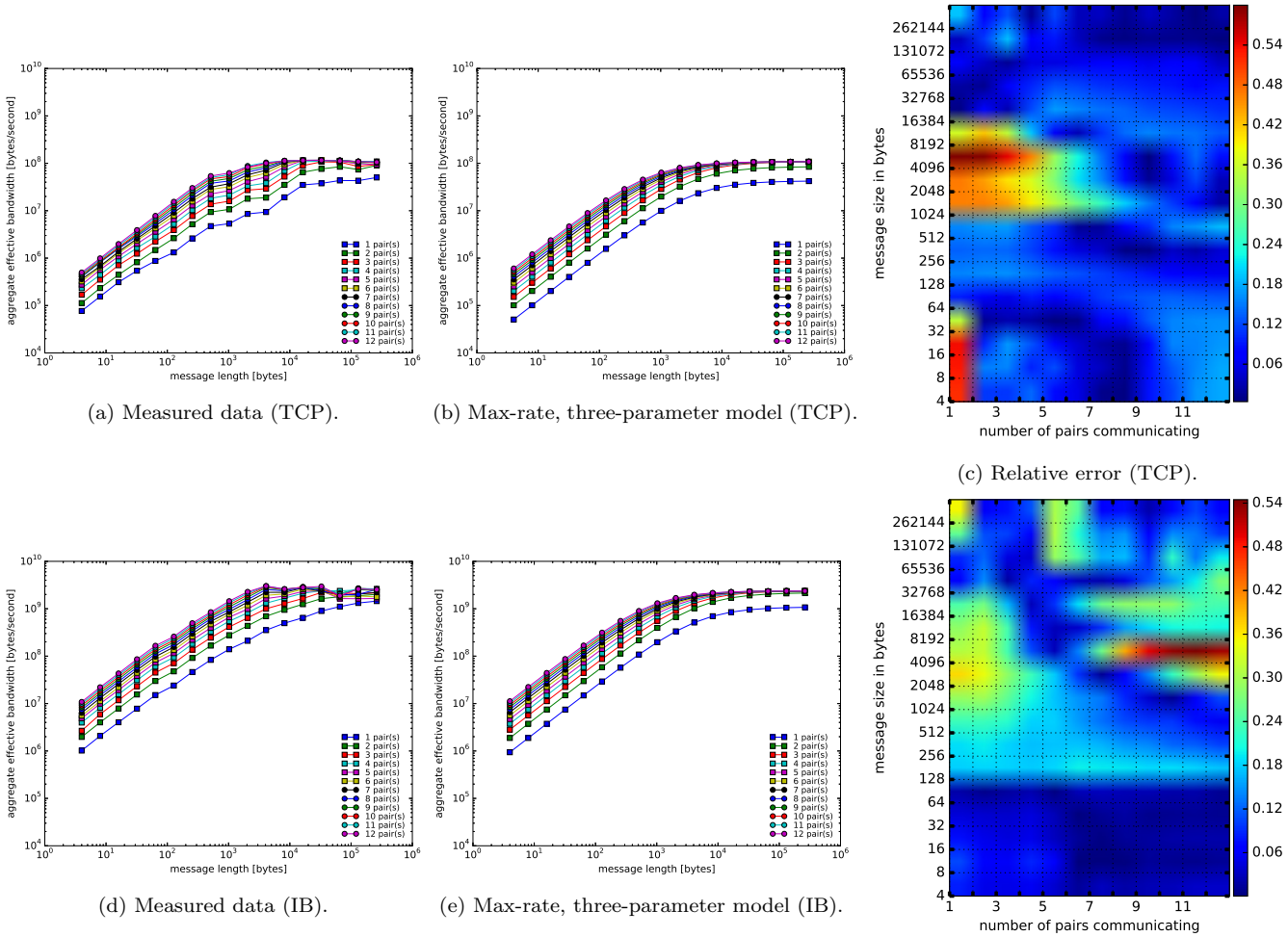


(e) Postal, two-parameter model (1–16 pairs).

Figure 2: Relative error in the least-squares fit of the models in Figure 1. *note: The color scheme scales are different for each figure.*

Table 2: Relative error sums for the different fit variants.

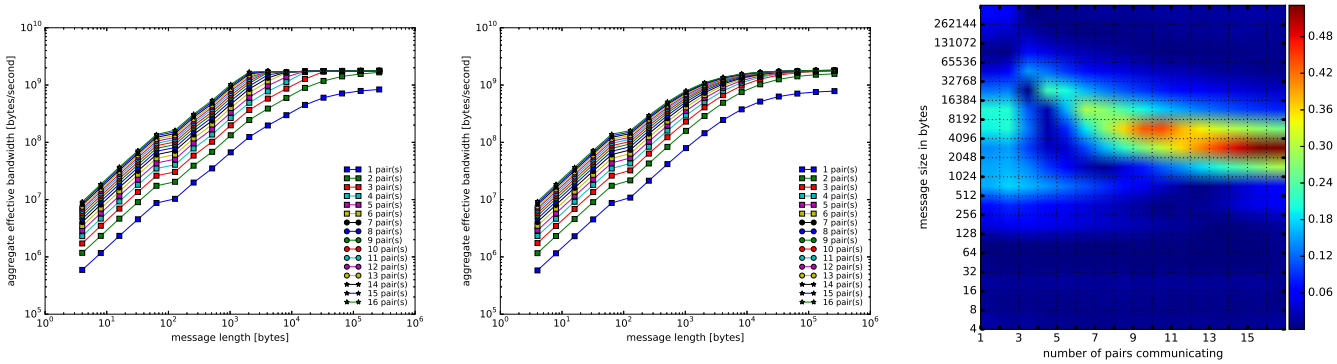
Phase	max-rate four-parameter	max-rate three-parameter	postal two-parameter (1 pair)	postal two-parameter (16 pairs)	postal two-parameter (complete set)
short	1.23	1.31	7.40	2.59	1.81
eager	1.71	1.85	10.97	3.54	2.37
rendezvous	7.60	8.37	58.78	109.39	46.87



(a) Measured data (TCP). (b) Max-rate, three-parameter model (TCP). (c) Relative error (TCP).

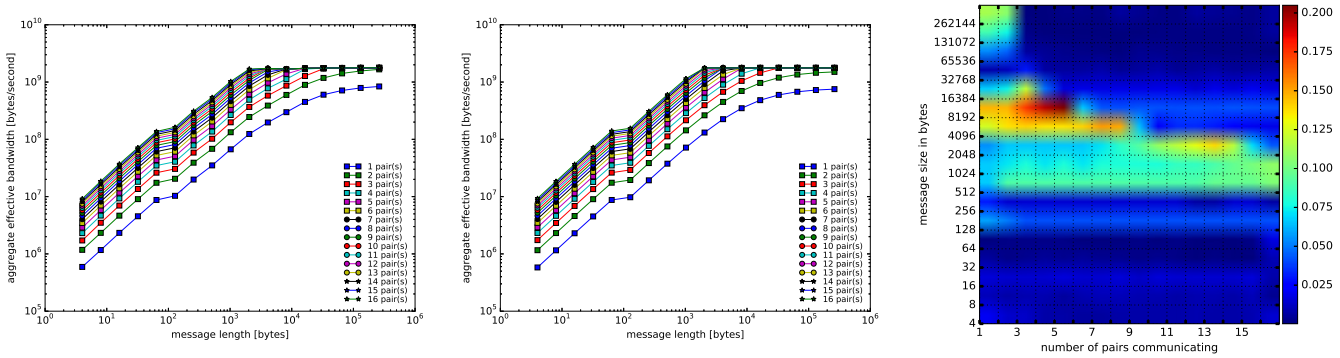
(d) Measured data (IB). (e) Max-rate, three-parameter model (IB). (f) Relative error (IB).

Figure 3: Aggregate effective bandwidth on the Illinois Taub Cluster with InfiniBand. 1 (bottom line) to 12 (top line) communicating processing pairs using TCP/MPICH 3.1.3 (top row) and InfiniBand/MVAPICH2.1 (bottom row).



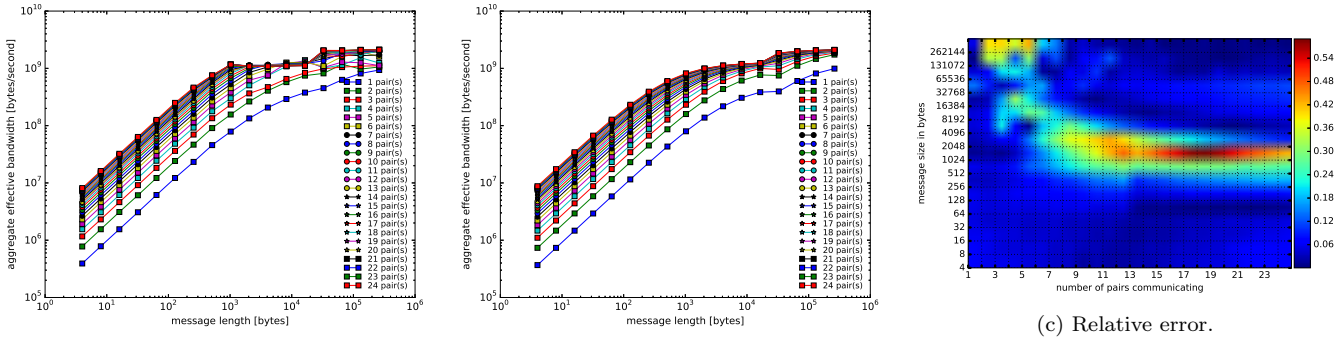
(a) Measured data. (b) Max-rate, three-parameter model. (c) Relative error.

Figure 4: Aggregate effective bandwidth on IBM Blue Gene/Q. The number of communicating pairs increases from bottom to top.



(a) Measured data. (b) Modified max-rate model. (c) Relative error of the model fit.

Figure 5: Aggregate effective bandwidth on IBM Blue Gene/Q. The number of communicating pairs increases from bottom to top.



(a) Measured data. (b) Max-rate, three-parameter model. (c) Relative error.

Figure 6: Aggregate effective bandwidth results the Cisco cluster. The number of communicating pairs increases from bottom to top.

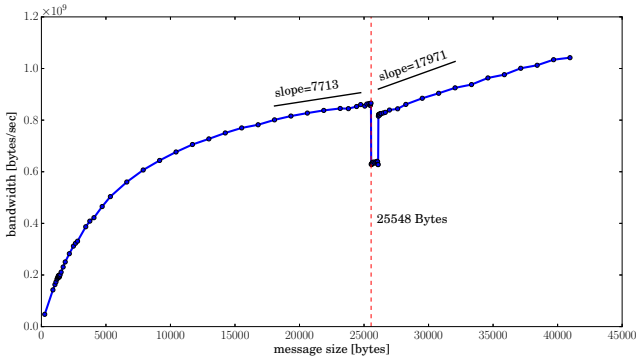


Figure 7: Bandwidth rates using mpptest on the Cisco cluster. `buflimit` identifies the threshold at 25548 bytes.

be identified and different model parameters used in each regime. While the decision to use eager, rendezvous, or another protocol is up to the MPI implementation (and there are other choices, such as a eager with a NACK to reject messages that are too long), most MPI implementations choose among a few protocols depending on the message length, and this should be taken into account in algorithm analysis and design.

As an example of the use of this model in algorithm design, consider the question of how many processes on a node should communication at the same time to minimize the communication time. This requires achieving the maximum bandwidth out of the node. Using the new model, the minimum time is achieved when the number of communication MPI processes is $k = R_N/R_c$, assuming no communication/computation overlap.

In summary, we hope the community adopts this model for communication time as it gives a much more accurate approximation to the performance of MPI on multicore nodes and is nearly as easy to use as the venerable postal model.

5.1 Acknowledgments

This research was supported in part by ExxonMobil Contract EM08150.9. This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07–25070) and the state of Illinois. The authors are grateful for the access to the Arcetri UCS Balanced Technical Computing Cluster at Cisco Systems.

6. REFERENCES

[1] T. Agarwal, A. Sharma, and L. V. Kalé. Topology-aware task mapping for reducing communication contention on large parallel machines. In *Proceedings of the 20th International Conference on Parallel and Distributed Processing*, IPDPS'06, pages 145–145, Washington, DC, USA, 2006. IEEE Computer Society.

[2] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: Incorporating long messages into the LogP model—one step closer towards a realistic model for parallel computation. In *Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '95, pages 95–105, New York, NY, USA, 1995. ACM.

[3] A. Bar-Noy and S. Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. In *Proceedings of the Fourth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '92, pages 13–22, New York, NY, USA, 1992. ACM.

[4] A. Bhatele, N. Jain, W. D. Gropp, and L. V. Kale. Avoiding hot-spots on two-level direct networks. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 76:1–76:11, New York, NY, USA, 2011. ACM.

[5] D. E. Culler, R. M. Karp, D. Patterson, A. Sahay, E. E. Santos, K. E. Schauser, R. Subramonian, and T. von Eicken. LogP: A practical model of parallel computation. *Commun. ACM*, 39(11):78–85, Nov. 1996.

[6] W. D. Gropp and E. Lusk. Reproducible measurements of MPI performance characteristics. In J. Dongarra, E. Luque, and T. Margalef, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, volume 1697 of *Lecture Notes in Computer Science*, pages 11–18. Springer Verlag, 1999. 6th European PVM/MPI Users' Group Meeting, Barcelona, Spain, September 1999.

[7] N. A. W. A. Hamid and P. D. Coddington. Comparison of MPI benchmark programs on shared memory and distributed memory machines (point-to-point communication). *IJHPCA*, 24(4):469–483, 2010.

[8] R. Hockney and C. Jesshope. *Parallel Computers: Architecture, Programming and Algorithms. 1981.*

[9] T. Hoefler, T. Mehlan, A. Lumsdaine, and W. Rehm. Netgauge: A network performance measurement framework. In R. H. Perrott, B. M. Chapman, J. Subhlok, R. F. de Mello, and L. T. Yang, editors, *HPCC*, volume 4782 of *Lecture Notes in Computer Science*, pages 659–671. Springer, 2007.

[10] S. Hunold, A. Carpen-Amarie, and J. L. Träff. Reproducible MPI micro-benchmarking isn't as easy as you think. In J. Dongarra, Y. Ishikawa, and A. Hori, editors, *EuroMPI/ASIA*, page 69. ACM, 2014.

[11] Intel Corporation. Getting started with Intel MPI Benchmarks 4.1.

[12] Phloem MPI benchmarks. https://asc.llnl.gov/sequoia/benchmarks/PhloemMPIBenchmarks_summary_v1.0.pdf.

[13] P. Luszczek, J. J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. Mccalpin, D. Bailey, and D. Takahashi. Introduction to the HPC Challenge Benchmark Suite. Technical report, 2005.

[14] H. Mierendorff, K. Cassirer, and H. Schwamborn. Working with MPI benchmarking suites on ccNUMA architectures. In J. Dongarra, P. Kacsuk, and N. Podhorszki, editors, *Recent Advances in Parallel Virutal Machine and Message Passing Interface*, number 1908 in Springer Lecture Notes in Computer Science, pages 18–26, Sept. 2000.

[15] OSU Micro-Benchmarks 5.3. http://mvapich.cse.ohio-state.edu/benchmarks/.

[16] J. Pješivac-Grbović, T. Angskun, G. Bosilca, G. E.

Fagg, E. Gabriel, and J. J. Dongarra. Performance analysis of mpi collective operations. *Cluster Computing*, 10(2):127–143, 2007.

[17] R. Reussner, P. Sanders, L. Prechelt, and M. Müller. SKaMPI: A detailed, accurate MPI benchmark. In V. Alexandrov and J. Dongarra, editors, *Recent advances in Parallel Virtual Machine and Message Passing Interface*, volume 1497 of *Lecture Notes in Computer Science*, pages 52–59. Springer, 1998.