CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2014)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3343

SPECIAL ISSUE PAPER

Asymmetric communication models for resource-constrained hierarchical ethernet networks

Jun Zhu[1,*,†], Alexey Lastovetsky[2], Shoukat Ali[3], Rolf Riesen[3] and Khalid Hasanov[2]

[1]*Technical University of Eindhoven, Eindhoven, The Netherlands*
[2]*University College Dublin, Dublin, Ireland*
[3]*Dublin Research Laboratory, IBM, Dublin, Ireland*

SUMMARY

Communication time prediction is critical for parallel application performance tuning, especially for the rapidly growing field of data-intensive applications. However, making such predictions accurately is non-trivial when contention exists on different components in hierarchical networks. In this article, we derive an 'asymmetric network property' on transmission control protocol (TCP) layer for concurrent bidirectional communications in a commercial off-the-shelf (COTS) cluster and develop a communication model as the first effort to characterize the communication times on hierarchical Ethernet networks with contentions on both network interface card and backbone cable levels. We develop a micro-benchmark for a set of simultaneous point-to-point message-passing interface (MPI) operations on a parametrized network topology and use it to validate our model extensively and show that the model can be used to predict the communication times for simultaneous MPI operations (both point-to-point and collective communications) on resource-constrained networks effectively. We show that if the asymmetric network property is excluded from the model, the communication time predictions will be significantly less accurate than those made by using the asymmetric network property. In addition, we validate the model on a cluster of Grid5000 infrastructure, which is a more loosely coupled platform. As such, we advocate the potential to integrate this model in performance analysis for data-intensive parallel applications. Our observation of the performance degradation caused by the asymmetric network property suggests that some part of the software stack below TCP layer in COTS clusters needs targeted tuning, which has not yet attracted any attention in literature.

Received 31 January 2014; Revised 24 June 2014; Accepted 25 June 2014

KEY WORDS: communication models; Ethernet; micro-benchmark; contention; hierarchical network

1. INTRODUCTION

Computing clusters have been the primary commodity platforms for running parallel applications. To build a cost-effective yet powerful cluster environment, high-speed networks are widely used to interconnect off-the-shelf computers. For application performance analysis on such clusters, an accurate time prediction for data sets transferred over the communication media is typically required [1]. To that end, several communication models have been proposed, for example, Hockney [2] and LogP [3]. They have been widely used to analyze the timing behaviors for point-to-point communications on parallel computers. However, these models simply see the network as a black box and can capture neither the communication *hierarchy* nor the network *resource sharing* that typically is present in modern large-scale systems. Both of these factors, network hier-

*Correspondence to: Jun Zhu, Technical University of Eindhoven, Eindhoven, The Netherlands.
†E-mail: kevin.jzhu@gmail.com

archy and resource sharing, make communication time prediction non-trivial and challenging for high-performance clusters.

On the other hand, such predictions are needed more now than ever because of the increasing importance of data-intensive applications [4, 5] that devote a significant amount of their total execution time in parallel processing to I/O or network communication, instead of computation. A good usable performance analysis of such data-intensive applications requires that the communication model reflects the network properties accurately on state-of-the-art network topologies and technologies.

In this article, we consider Ethernet-based network because, compared with custom interconnects (e.g., InfiniBand and Myrinet), it offers widespread compatibility, better cost-performance tradeoff, and a superior road map to 100-Gb standard [6, 7]. As of June 2011, 1 or 10 Gb Ethernet has been used as the communication infrastructure in over 45% of the top 500 supercomputers [8]. We use message-passing interface (MPI) as the programming model, which has become the *de facto* standard for application layer communication on distributed memory systems. On the basis of transmission control protocol (TCP) messaging protocol, MPI over 1 Gb Ethernet has shown comparable performance on network bandwidth and latency as on custom networks [9].

An example Ethernet cluster, consisting of two racks, is illustrated in Figure 1. It has a scalable tree (star bus) topology; that is, two star-configured intra-rack segments are connected using a linear backbone cable. The computer nodes in the same rack are connected to a *top-of-rack* (ToR) switch via network interface cards (NICs), and each node has a dedicated bandwidth on the intra-rack point-to-point connection. Different ToR switches are connected together to form a hierarchical network. We use this cluster as one of our testbed systems to verify network properties and the proposed communication model. In this testbed, contention may occur at different levels of the communication infrastructure:

- Network interface cards. In multi-core processors, multi-socket nodes, or a combination thereof, each NIC may be shared by multiple cores concurrently. That is, the number of NICs on each node is generally less than the number of cores, and NIC-level sharing exists.
- Backbone cable. Several inter-rack communication operations may be aggregated on the backbone that runs between racks to effectively utilize the high bandwidth available.

For instance, in Figure 1, logical communication links e_{a_1,b_1} and e_{a_2,b_2} share the same NIC on $node_{X_1}$, and e_{a_1,b_1}, e_{a_3,b_3}, and e_{a_4,b_4} share the inter-rack backbone instead. When resource contention happens on a *hierarchical* network, it makes communication times prediction more difficult.

1.1. Contributions

We derive network properties on parametrized network topology from simultaneous point-to-point MPI operations and independently discover the *asymmetric network property* [10, 11] on TCP layer

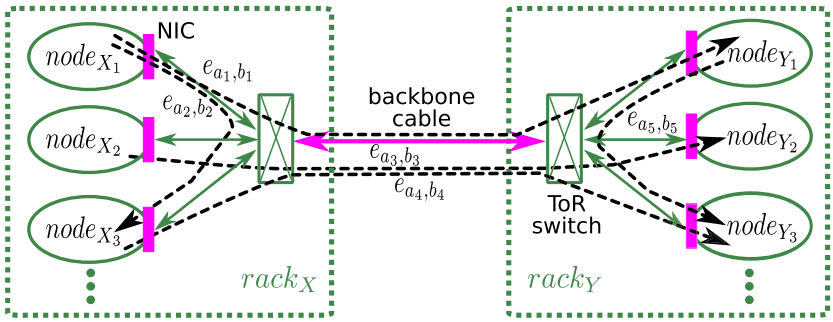


Figure 1. A tree topology platform: two star-configured racks connected via the backbone cable. The dashed arrows denote one example application with five logical communication links: $e_{a_1,b_1} - e_{a_5,b_5}$. The processes on each logical link are not explicitly labeled for clarity in the graph.

DOI: 10.1002/cpe

for concurrent bidirectional communications in an in-house COTS research Ethernet cluster. Our work is the first effort to characterize the asymmetric effect of concurrent MPI communications in *resource-constrained hierarchical* Ethernet clusters. Results from statistically rigorous experiments clearly show that our model can predict the communication times with low errors for various communication patterns, with network contention on both NICs and backbone cable levels. We show that if the asymmetric network property is excluded from the model, the communication time predictions for both point-to-point and collective communications become significantly less accurate than those made by using the asymmetric network property. In addition, a widely used Grid5000 infrastructure has been adopted to validate our experimental and theoretical findings on a more loosely coupled Ethernet platform.

1.2. Overview

The remainder of this article is structured as follows. Section 2 discusses some related work. Our MPI micro-benchmark and platform are introduced in Section 3. Section 4 introduces the notations used in this article. Section 5 introduces some network properties derived from benchmarking. We propose our communication model in Section 6 and present the corresponding experiments in Section 7. Finally, Section 8 concludes the article.

2. RELATED WORK

Many models have been proposed in the parallel and distributed computing literature to characterize the communication times for parallel computers. In the *Hockney model* [2], the point-to-point communication time to send a message of size m is captured as $\alpha + \beta * m$, where α is the latency for each message and β is the inverse of the network bandwidth. Culler *et al*. [3] describe *LogP* communication model for small messages. The model is aware of the finite network capacity, which is the upper bound on messages in parallel transmission on the network. Hoefler *et al*. [12] investigate how this model can be modified to capture the timing behaviors for small messages on InfiniBand networks. In [13], *LogGP* is developed, as an extension of the LogP model, to better fit large messages. In *LogGPS* [14], the synchronization needed in large messages transmission, has been taken into consideration, and the message-size-dependent CPU overhead has been further captured in another extension *LogGOPS* [15] as well. In [16, 17], a heterogeneous model *LMO* is proposed, which separates the variable contributions of the processors and network in communication timing analysis. However, all of the aforementioned work assumes that there is no contention on the network. When multiple communication transactions happen concurrently on the same network resource, the aforementioned models assume that the available bandwidth is even shared by all concurrent communications. Let us call this the *symmetric network property*. We will show that this property does not exist for complicated hierarchical networks.

In [18], a performance model of many-to-one collective communications for MPI platforms on a single-switched Ethernet cluster is proposed. It reflects a significant and non-deterministic increase in the execution time for medium-sized messages, persistently observed for different parallel platforms and MPI implementations, which is caused by the resource contention on the root side. In [19], this model is used for optimization of MPI applications. However, it is restricted to only one type of collective communication on a flat star network and has never been extended to hierarchical network topology. In [20], a contention-aware communication timing model is proposed for multicore clusters using an InfiniBand network. This work analyzes the dynamic network contention and derives the so-called penalty coefficients to characterize the bandwidth distribution at the NIC level. This work is similar to ours in the sense that it does not recognize the symmetric network property; instead, it explicitly calculates how the total available bandwidth will be divided among the contending communications. This model focuses on a flat star network, in which all computer nodes are connected to a single switch with dedicated bandwidth. However, modern large-scale systems usually have hierarchical network topology. This is the part that had been lacking, and this is what we address in our work.

DOI: 10.1002/cpe

In network community, two-way TCP traffic has been studied on a single bottleneck link in a perfectly symmetrical setup [21]. A well-known communication performance drop on the congested link has been attributed to ACK bursts, also known as *ACK compression* phenomenon. Later, this performance issue has been re-investigated, and a new explanation is provided on *data pendulum* interactions [22]. That is, data and acknowledgement segments alternately fill one of the two-way link buffers on TCP connections, which slows down the reverse traffic. This low-level data pendulum phenomenon can be used to explain the asymmetric property independently reported in our previous work [10] and a recent research on flow-level network model [11]. While the more general network model in [11] has been mainly focused on model validation by comparing with packet-level simulation, our work has been used to model and predicate point-to-point and collective MPI communications on two specific state-of-the-art Ethernet clusters.

In [23], a predictive model for MPI communications has been studied on large-scale, Ethernet-connected system. However, it does not handle the asymmetric property on congested TCP networks as in our work. We also note that statistical methods, such as queuing theory [24], have been used to predict the non-deterministic behavior of switched Ethernet. Nevertheless, such methods rely on pre-defined characteristics of network switches and flow, which are themselves non-trivial to be captured mathematically in a cluster environment.

3. MICRO-BENCHMARK AND PLATFORMS

3.1. Micro-benchmark

We designed a point-to-point MPI micro-benchmark to measure concurrent communication times for use in our testbed. The pseudo-code for a pair of sender and receiver processes is shown in Figure 2. In each iteration, a message with a user-specified size is initialized to remove the potential memory or network cache effects. Each time a message is sent from the sender to receiver, a non-blocking receive operation is pre-posted, and the receiver records the time spent on message transmission in `tArray` for statistics. For each sender-to-receiver communication, we repeat the transmission at most `maxIter` times, where `maxIter` is some large number (set to 2000 in our experiments). Specifically, we stop if the width of the 95% CI becomes 2%, or less, of the average measured time. For clarity, this code shows only one pair of sender and receiver processes. In general, our benchmark can be set up to have any given number of pairs of sender and receiver processes for simultaneous MPI communication operations, to test a required number of simultaneous MPI communication operations.

We use OpenMPI 1.5.4 [25] as the MPI implementation. Each MPI process can be bound to any computing resource (core or node) as specified in a given communication pattern, with the support of *processor affinity* in OpenMPI. We only use large message sizes $\geqslant$ 10 MB in benchmarking, which suit data-intensive problems. Compared with communication delay, the message latency α (also called propagation delay) is negligible [20].

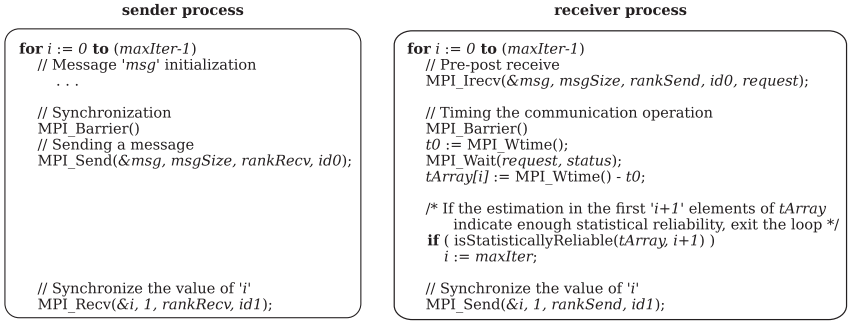


Figure 2. Micro-benchmark: communication time measurement on a process pair.

DOI: 10.1002/cpe

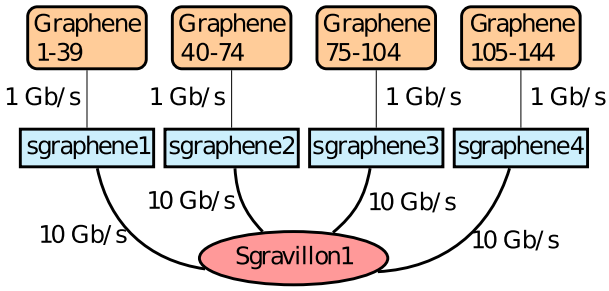


Figure 3. The Graphene cluster used for validation on Grid5000 platform.

3.2. Platform specifications

3.2.1. Main testbed. There are up to 15 nodes settled in each rack of our main experimental platform. Each node is a dual-socket six-core (Intel Xeon X5670 6C@2.93 GHz) server with an Intel 1 Gb NIC card, operated with Red Hat Enterprise Linux 5.5 x86-64. The Ethernet switch is IBM BNT Rack Switch G8264, which supports over 10 Gb Ethernet and is interoperable with clients using 1 Gb connectivity as well. The theoretical communication bandwidth on different network resources is 1 and 10 Gbps for NICs and the optical backbone respectively.

The NIC settings in Linux on each node are tuned to run for gigabit speed [26]. In our hybrid 1/10 Gb Ethernet cluster environment, the network round-trip time (RTT) between two nodes on different racks is 450 μs. In TCP communication, the size of the socket buffer limits the number of packets to be sent yet not acknowledged by the receiver. The socket buffer size is at least RTT $\times$ bandwidth. We set the TCP socket buffer size in OpenMPI to 3 MB (MCA parameters `btl_tcp_sndbuf` and `btl_tcp_rcvbuf`).

3.2.2. Validation testbed. In addition to the main testbed, we have carried out some experiments on Grid5000 platform as well to validate our proposed communication model on a different platform. Grid5000 is a large-scale parallel and distributed infrastructure consisting of 20 clusters distributed over nine sites in France and one in Luxembourg. Our experiments were performed on Nancy site, which consists of three clusters: Graphene, Griffon, and Graphite. The Graphene cluster is used in our experiments. The cluster is equipped with 144 nodes, and each node has four cores of CPU Intel Xeon X3440 and 16 GB of memory. The nodes in the Graphene cluster are connected to four individual switches for each group in a hierarchical way as illustrated in Figure 3. That is, these switches are connected to the main Nancy router via 10 Gb/s links, and the nodes inside each group are interconnected to their own switches via 1 Gb/s network.

4. PRELIMINARIES

Here, we introduce the notations on application and platform, which will be used to formalize network properties and communication models in the following sections. Let $|\cdot|$ denote the cardinality of a set or vector and $(\cdot)^{-1}$ the inverse of a value. The index of a vector starts at 0, and the *i-th* element of a vector $\vec{V}$ is denoted as $\vec{V}[i]$.

4.1. Application

An application is a collection of concurrent point-to-point MPI operations, denoted as a tuple (P, E). A finite set P contains an even number of processes, which are connected in pairs via a finite set $E \subseteq P^2$ of edges, with $|P| = 2 * |E|$. Each edge $e_{a,b} \in E$ denotes a logical communication link (*dependency*) between a process pair: a sender p_a and a receiver p_b. A message size $m_{a,b}$ is assigned to the communication operation on $e_{a,b}$ to denote the data volume needed for network transmission. In an application, all the communication operations start simultaneously, and no communication operation has a logical dependency with any other. Once a transmission is finished, the

DOI: 10.1002/cpe

corresponding logical link is removed from the application at run-time. The goal of this article is to predict the communication time $t_{a,b}$, which is required to finish the data transmission, for each link $e_{a,b}$ in an application.

4.2. Platform

The network infrastructure of our experimental cluster has a tree topology, as illustrated in Figure 1. In general, the cluster consists of a set R of racks. The set N of computing nodes is the union of all nodes in individual racks. There is one NIC configured for each node, and all nodes in the same rack communicate via the ToR switch. A full-duplex communication network is employed in our work. We denote the inverse bandwidth of the NIC, electrical backbone, and optical backbone as β_N, β_E, and β_O, respectively.

4.3. Mapping: application to platform

The mapping process binds each process in MPI applications to a computing node in our platform. The binding function is defined as $\mathcal{B}^N : P \rightarrow N$, which associates every process $p \in P$ to a $node \in N$ to which it is bound. Similarly, how processes are bound to racks is defined $\mathcal{B}^R : P \rightarrow R$. For instance, in Figure 1, $\mathcal{B}^N(p_{a_5}) = node_{Y_1}$, and $\mathcal{B}^R(p_{a_5}) = \text{rack}_Y$. In our work, we only map up to one process to each core to eliminate the unnecessary context switching overhead.

4.4. Network contention

When multiple processes are bound to one (multi-core) node or rack, several simultaneous logical links may share the same NIC or inter-rack backbone. In full-duplex network, we distinguish resource contention on incoming links from that on outgoing links. The set of incoming logical links that are bound to the same resource $x \in N \cup R$ are denoted as $\mathcal{E}^+(x)$. The degree of this set $|\mathcal{E}^+(x)|$ is simply denoted as $\delta^+(x)$, where $\delta^+(\cdot)$ is the indegree function for a resource defined as $\delta^+ : N \cup R \rightarrow \mathbb{N}_0$. Similarly, $\mathcal{E}^-(\cdot)$ and $\delta^-(\cdot)$ are defined for outgoing logical links of a resource. For instance, $\mathcal{E}^-(node_{X_3}) = \{e_{a_4,b_4}\}$ and $\delta^-(\text{rack}_X) = 3$ in Figure 1.

For simultaneously bidirectional communication (Section 5.2), we also distinguish logical links with resource sharing at the *same* direction (*with-flow*) to logical links on the *reverse* direction (*contra-flow*).

4.4.1. Congestion factors. To detect the communication *bottleneck* in a *hierarchical* network, we associate with $e_{a,b}$ a vector $\vec{S}_{a,b}$ of sets of logical links, defined as follows

$$\vec{S}_{a,b} =< \mathcal{E}^- \left(\mathcal{B}^R(a)\right), \mathcal{E}^- \left(\mathcal{B}^N(a)\right), \mathcal{E}^+ \left(\mathcal{B}^N(b)\right) > \tag{1}$$

where $\vec{S}_{a,b}$ includes three sets of with-flow logical links of $e_{a,b}$ on the shared resource. For instance, the link e_{a_1,b_1} in Figure 1 has

$$\vec{S}_{a_1,b_1} =< \left\{e_{a_1,b_1}, e_{a_3,b_3}, e_{a_4,b_4}\right\}, \left\{e_{a_1,b_1}, e_{a_2,b_2}\right\}, \left\{e_{a_1,b_1}\right\} >$$

To detect the congestion bottleneck of a logical link on network resources, a congestion factor $k_{a,b}$ is defined as follows

$$k_{a,b} = \max \left\{k \mid k = \vec{\beta}[i] \cdot |\vec{S}_{a,b}[i]|, \ \forall i \in \left[0, |\vec{S}_{a,b}|\right)\right\} \tag{2}$$

where $\vec{\beta}$ is a vector with platform-dependent inverse bandwidth values and $\vec{\beta}[i]$ the *i-th* inverse bandwidth of the network resource on which the set $\vec{S}_{a,b}[i]$ of logical links are sharing. For instance, when the cluster in Figure 1 has an optical backbone cable, $\vec{\beta} =< \beta_O, \beta_N, \beta_N >$, and $k_{a_1,b_1} = \max\{3\beta_O, 2\beta_N, \beta_N\}$. That is, the congestion factor is configuration aware in a hybrid network.

DOI: 10.1002/cpe

Similarly, vector $\vec{\bar{S}}_{a,b}$ of sets of contra-flow logical links of $e_{a,b}$ and the *reverse* congestion factor $\bar{k}_{a,b}$ are defined

$$\vec{\bar{S}}_{a,b} =< \mathcal{E}^{+}\left(\mathcal{B}^{R}(a)\right), \mathcal{E}^{+}\left(\mathcal{B}^{N}(a)\right), \mathcal{E}^{-}\left(\mathcal{B}^{N}(b)\right) > \tag{3}$$

$$\bar{k}_{a,b} = \max\left\{k \mid k = \vec{\beta}[i] \cdot |\vec{\bar{S}}_{a,b}[i]|, \forall i \in \left[0, |\vec{\bar{S}}_{a,b}|\right)\right\} \tag{4}$$

5. NETWORK PROPERTIES

We derive some network properties from MPI benchmarking on a resource-constrained network platform. In this section, we characterize the inverse bandwidth $\beta_{a,b}$ of each logical link $e_{a,b}$ in a particular communication pattern, which is time independent. However, when some communication operations finish earlier, the specified communication pattern may vary at different time instance t.

5.1. Unidirectional communication: fairness property

Given an application with a number $|E|$ of point-to-point MPI operations, we design the following unidirectional experiments to inspect the arbitration fairness on different levels of the network hierarchy:

A. Intra-rack communication—all sender processes are mapped onto one node, while the matching receiver processes are mapped onto another node in the same rack.
B. Inter-rack communication—all sender processes are mapped onto different nodes in rack_X, while the matching receiver processes are mapped onto different nodes in rack_Y.

We observe that when $|E|$ increases, the average bandwidth for logical links on the shared NIC (experiment A) decreases and that the bandwidth is fairly distributed over all links. The same is for the optical fiber backbone (experiment B) when $|E| > 10$. When $|E| <= 10$, the 10-Gb optical fiber is not saturated, and therefore, the average bandwidth is almost constant (940 Mbps), giving a measured maximum aggregate bandwidth $\beta_O^{-1} = 9.4$ Gbps. For experiment B using electrical copper backbone, the results are similar to those for a NIC. However, the bandwidth distribution in experiment B does not really depend on copper versus optical; it is the bandwidth of physical links in the hierarchical network that matters.

Mathematically, the inverse bandwidth $\beta_{a,b}$ based on a *fairness property* of each logical link $e_{a,b}$ can be captured as

$$\beta_{a,b} = \begin{cases} \beta \cdot |E|, \text{ if } \beta = \beta_O \text{ and } |E| > 10 \text{ or } \beta = \beta_E \\ \beta_E, \text{ if } \beta = \beta_O \text{ and } |E| \leqslant 10 \end{cases} \tag{5}$$

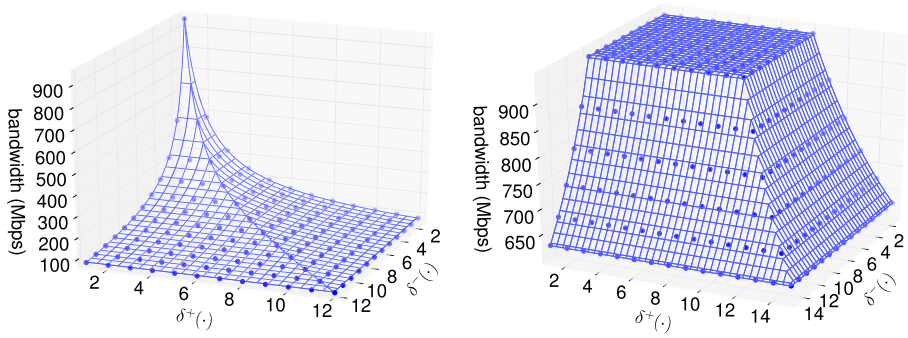
where β is the inverse bandwidth of the physical link on which $e_{a,b}$ is located.

5.2. Bidirectional communication: asymmetric property

In a full-duplex network, the network resources might be shared by multiple communication logical links in both directions simultaneously. To study bidirectional communication on shared network resources, we swap the mapping policy for some of the sender and receiver processes in the experiments in Section 5.1. When the number of *incoming* $\delta^{+}(\cdot)$ and *outgoing* $\delta^{-}(\cdot)$ logical links vary on the shared node or rack, the average bandwidth of each link is illustrated in Figure 4, with a standard deviation of up to 2%. We omit the results using electrical copper backbone (experiment B) for clarity, which are similar to those for a NIC. The inverse bandwidth $\beta_{a,b}$ of each logical link $e_{a,b}$ can be captured

$$\beta_{a,b} = \begin{cases} \beta \cdot \delta_{max}(\cdot), \text{ if } \beta = \beta_O \text{ and } \delta_{max}(\cdot) > 10 \text{ or } \beta = \beta_E \\ \beta_E, \text{ if } \beta = \beta_O \text{ and } \delta_{max}(\cdot) \leqslant 10 \end{cases} \tag{6}$$

DOI: 10.1002/cpe



(a) Shared NIC (b) Shared optical fiber backbone

Figure 4. The average bandwidth for bidirectional logical links on (a) shared network interface card (NIC) and (b) shared optical fiber backbone, with deviations up to 2%.

where $\delta_{max}(\cdot) = \max\left(\delta^+(\cdot), \delta^-(\cdot)\right)$. The results clearly show that the total duplex bandwidth may not be achieved, when $\delta^+(\cdot)$ and $\delta^-(\cdot)$ do not match (are asymmetric) on the shared resource. For instance, when $\delta^+(\cdot) = 12$ and $\delta^-(\cdot) = 1$ in (a), the total bidirectional bandwidth is $\frac{13}{12} \cdot 940$ Mbps, instead of $2 \cdot 940$Mbps (according to a fair dynamic bandwidth allocation in full-duplex mode). For instance, when $\delta^+(\cdot) = 2$ and $\delta^-(\cdot) = 1$ in (a), the two incoming and one outgoing logical links all have bandwidth 470 Mbps, instead of 470, 470, and 940 Mbps respectively (according to a fair dynamic bandwidth allocation in full-duplex mode). We have validated this property using a TCP network testing tool Iperf [27] with the same experimental settings on TCP layer.

6. COMMUNICATION MODEL AND TIME PREDICTION

Here, we present our communication model. The inverse bandwidth for logical links is first derived. Then, we capture the temporal dynamics in communication patterns and predicate the communication times.

6.1. Resource-constrained bandwidth

Let $\bar{E}$ be the set that stores the links for which $\beta_{a,b}$ has been calculated, E the set of links not yet calculated. $PQ(E)$ is a priority queue based on elements $\forall e_{a,b} \in E$, which is ranked by the descending order of a multi-key

$$\mathcal{K}_{PQ(E)} =< \max(k_{a,b}, \bar{k}_{a,b}),\ k_{a,b},\ \bar{k}_{a,b} > \tag{7}$$

The most congested logical link in a hierarchical network is the one with the highest value of $\mathcal{K}_{PQ(E)}$, which depends both on network capacity and the number of logical links on the sharing network resources.

On the basis of the derived network properties and heuristics from benchmarking, we propose Algorithm 1 to calculate the inverse bandwidth for logical links with simultaneous MPI communications. The analysis flow works iteratively in a bottleneck-driven manner. In each iteration, the most congested logical link in E is proposed to be analyzed (line 5), and the bandwidth of this logical link is calculated differently when the network congestion is caused by either with-flow or contra-flow traffic:

- When congestion happens on with-flow traffic (line 7), the bandwidth is calculated on the basis of the fairness properties on the with-flow bottle network resource (lines 9 and 10).
- Otherwise (line 11), the bandwidth is calculated on the basis of the asymmetric properties on the contra-flow bottle network resource (lines 13–17). Heuristically, when contra-flow congestion happens on a non-saturated physical link, the contra-flow congestion on the logical link is disabled, and the logical link is sent back to the queue to be re-analyzed (lines 19 and 20).

DOI: 10.1002/cpe

Algorithm 1: Inverse bandwidth for logical links.

Output: Inverse bandwidth $\beta_{a,b},\ \forall e_{a,b} \in E$

```
1  /* Initialization: set Ē to store calculated links */
2  Ē := ∅
3  while E ≠ ∅ do
4      /* To retrieve the most congested link in PQ(E) */
5      e_{a,b} := PQ(E).pop()
6      β_{a,b} := 0
7      if k_{a,b} ⩾ k̄_{a,b} then
8          foreach i ∈ [0, |S⃗_{a,b}|] do
9              if β⃗[i] · |S⃗_{a,b}[i]| == k_{a,b} then
10                 β_{a,b} := max(β_{a,b}, inverseBandwidth(i))
11     else
12         foreach i ∈ [0, |S⃗⃗_{a,b}|] do
13             if β⃗[i] · |S⃗⃗_{a,b}[i]| == k̄_{a,b} then
14                 E′ = S⃗⃗_{a,b}[i] ∩ Ē
15                 /* If contra-flow links saturate the physical link */
16                 if Σ{β⁻¹_{x,y} | ∀e_{x,y} ∈ E′} == β⃗⁻¹[i] then
17                     β_{a,b} = min{β_{x,y} | ∀e_{x,y} ∈ E′}
18                 else
19                     S⃗⃗_{a,b}[i] := ∅
20                     PQ(E).insert(e_{a,b})
21     /* To update E and Ē, once k_{a,b} is calculated */
22     if β_{a,b} ≠ 0 then
23         E := E \ {e_{a,b}}
24         Ē := Ē ∪ {e_{a,b}}
25 where
26 Function inverseBandwidth(i)
27     E′ = S⃗_{a,b}[i] ∩ Ē, E″ = S⃗_{a,b}[i] ∩ E
28     β̂ = β⃗[i]
29     /* The used bandwidth on the congested physical link */
30     k′ = Σ{β̂⁻¹_{x,y} | ∀e_{x,y} ∈ E′}
31     /* To calculate link bandwidth based on fairness property */
32     return β_{a,b} := max(β_E, |E″| / (β̂⁻¹ − k′))
```

While E is not empty, the algorithm explores the links iteratively until all the logical links are analyzed. In the worst case, two iterations may be needed to calculate $\beta_{a,b}$ for each link, and Algorithm 1 terminates in at most $2 \cdot |E|$ iterations. Furthermore, our communication model could be extended to more complex topologies, such as 2D mesh, in which similar network contention happens in different regions in the network.

6.2. Temporal dynamics and communication times

To predict the time required for each communication operation, we propose Algorithm 2. It formalizes how the communication times are calculated, which depend on message sizes and the derived inverse bandwidth of logical links (Algorithm 1). In [20], a similar mechanism has been implemented as a sequence of linear model. In the lifetime of this algorithm, a timer t (initialized in line 1) is used in analysis.

Once some logical links finish the communication operations and are removed from the application (lines 12–14), the time-aware inverse bandwidth $\beta_{a,b}(t)$ for each logical link is updated at run-time invoking Algorithm 1. That is, the bandwidth of other logical links may be redistributed dynamically. The size of the message transmitted in the current *step* is temporally stored in $\Delta m_{a,b}$. The predicted communication time $T^{\text{pred}}_{a,b}$ for each communication operation is calculated until all logical links are removed from analysis.

DOI: 10.1002/cpe

Algorithm 2: Communication times for logical links.

Output: Predicted time $T_{a,b}^{\text{pred}},\ \forall e_{a,b} \in E$

$t := 0$
$step := 0$
while $E \neq \emptyset$ **do**
  $step := step + 1$
  /* *The earliest time any communication could finish data transmission* */
  $\Delta t = min\{t' \mid t' = m_{a,b} \cdot \beta_{a,b}(t), \forall e_{x,y} \in E\}$
  $t := t + \Delta t$
  foreach $\forall e_{a,b} \in E$ **do**
    $\Delta m_{a,b} = \Delta t \cdot \beta_{a,b}^{-1}(t)$
    /* *To update the left message size* */
    $m_{a,b} := m_{a,b} - \Delta m_{a,b}$
    if $m_{a,b} == 0$ **then**
      $E := E \setminus \{e_{a,b}\}$
      $T_{a,b}^{\text{pred}} := t$

Table I. Communication times analysis—example application.

$m_{a,b}$(MB)		e_{a_1,b_1} 10	e_{a_2,b_2} 20	e_{a_3,b_3} 20	e_{a_4,b_4} 20	e_{a_5,b_5} 20
Step 1	$\beta_{a,b}^{-1}$	313.3	626.6	313.3	313.3	626.6
	$\Delta m_{a,b}$	10	20	10	10	20
	$T_{a,b}^{\text{pred}}$	0.255	0.255	—	—	0.255
Step 2	$\beta_{a,b}^{-1}$	—	—	470.0	470.0	—
	$\Delta m_{a,b}$	—	—	10	10	—
	$T_{a,b}^{\text{pred}}$	—	—	0.425	0.425	—
$T_{a,b}^{\text{meas}}$		0.252	0.245	0.406	0.443	0.270
$\epsilon_{a,b}$(%)		1.2	4.0	4.7	−4.1	−4.6

6.3. Case study

Assuming an electrical copper backbone is used for inter-rack communication in the cluster environment, here, we use the application in Figure 1 as an example to demonstrate how communication times are predicted. For each communication link $e_{a,b}$, an initial message size $m_{a,b}$ is specified, and all the communication operations start simultaneously. Two dynamic steps are needed to calculate the predicted communication times, as illustrated in Table I. Three links e_{a_1,b_1}, e_{a_2,b_2}, and e_{a_5,b_5} finish the data transmission at the end of step 1, and the link bandwidth is redistributed for e_{a_3,b_3} and e_{a_4,b_4} in step 2. Let $T_{a,b}^{\text{meas}}$ be the measured time from benchmarking, we compute the error, $\epsilon_{a,b} = \frac{|T_{a,b}^{\text{pred}} - T_{a,b}^{\text{meas}}|}{T_{a,b}^{\text{meas}}}$, for each communication link $e_{a,b}$.

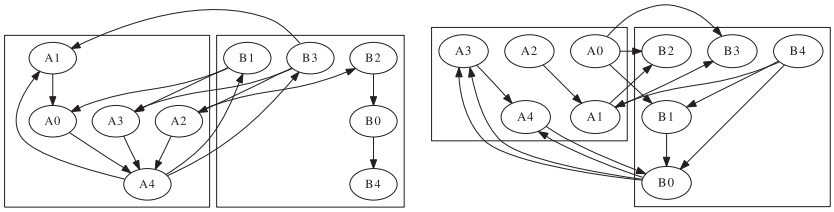
7. EXPERIMENTS AND RESULTS

7.1. Experiments design

We conducted our experiments on the main testbed with two racks connected via an optical backbone. That is, the cluster environment has been configured with 1 Gb Ethernet for intra-rack communication and 10 Gb Ethernet for inter-rack communication. Each time the same number of nodes are configured in both racks, with a total number of nodes $|N|$ up to 30.

To construct one test, we in turn consider each one of the nodes on both racks. For each such node, $node_{\text{src}}$, we randomly select a different node, $node_{\text{dst}}$, from the set N. We then include the directed point-to-point communication $node_{\text{src}} \rightarrow node_{\text{dst}}$ (as one instance of process pair in our benchmark) in the test with a 50% probability. For each $node_{\text{src}}$, we perform this matching process d times. It

DOI: 10.1002/cpe



(a) (b)

Figure 5. The communication patterns of two test instances, when $|N| = 10$ and $d = 3$.

ensures that we obtain random communication patterns in the test and that our results do not depend on a 'lucky' selection of communication patterns. In our experiments, each new test is constructed independently of the previous tests and most likely has a very different communication pattern. As illustrated in Figure 5, two random instances of the test, generated when $(|N|, d) = (10, 3)$, show quite different communication patterns. To further ensure that the goodness of our reported results is not a result of biased selection of input, we consider an experiment completed only when enough tests have been performed to give us a certain level of confidence in the average value of ϵ.

7.1.1. Validation experiments. For the validation experiments on Grid5000, we have used nodes only from the first two switches (sgraphene1 and sgraphene2; Figure 3). During the experiments, whole Graphene and Griffon (which is also connected to the main Nancy router) clusters were reserved to minimize possible noises. The communication patterns are the same as in the experiments on the main testbed.

7.2. Experimental results: point-to-point communication

We have designed nine experiments, each with a different set of values for parameters $|N|$ and d, as illustrated in Table II. In each experiment, the number of communication links is $N_{\text{trials}} > 500$. A total of 354 randomly generated communication patterns are tested. These communication patterns are non-trivial, with the maximum number of logic links $|E|$ in one pattern up to 57 and the maximum indegree $\delta^+(\cdot)$ or outdegree $\delta^-(\cdot)$ (i.e., the number of concurrent links) on the congested network resource up to six for NICs and 10 for the optical backbone.

The distribution of the estimation error ϵ on each cluster of experiments is illustrated in Figure 6. For communication times prediction based on our proposed model (the first row in Figure 6), when d varies from 1 to 3 in experiment settings, there are 83.2%, 77.3%, and 72.1% of communication links respectively, which fall within the margin of error $|\epsilon| \leqslant 10\%$. On the contrary, when the asymmetric property is not considered (the second row in Figure 6), that is, only with fairness property, the percentage of these links fall to 66.8%, 58.0%, and 68.8%. The prediction error with pure fairness property can be as worse as -80%, which means the predicted times are five times lower than the measured ones.

For the validation experiments on Grid5000, the estimation error ϵ has shown similar distributions as illustrated in Figure 7. For communication times prediction based on our proposed model, when d varies from 1 to 3 in experiment settings, there are 82.92%, 67.87%, and 64.16% of communication links respectively, which fall within the margin of error $|\epsilon| \leqslant 10\%$. On the other hand, when the asymmetric property is not considered, the percentage of these links fall to 78.22%, 64.06%, and 57.34% respectively.

We can see that our model is quite accurate from the averaged value for error $|\epsilon|$ on both the main testbed and Grid5000. Within less than 0.2% imprecision, the largest average error on the main testbed occurs for experiment 9, being approximately 9.5% of the measured value, and the largest average error on Grid5000 occurs for experiment 8, being approximately 11.87% of the measured value.

A clear trend emerges from Table II. As d increases, both the average number of dynamics steps in timing analysis and the average error increase. This phenomenon produces three distinct clusters

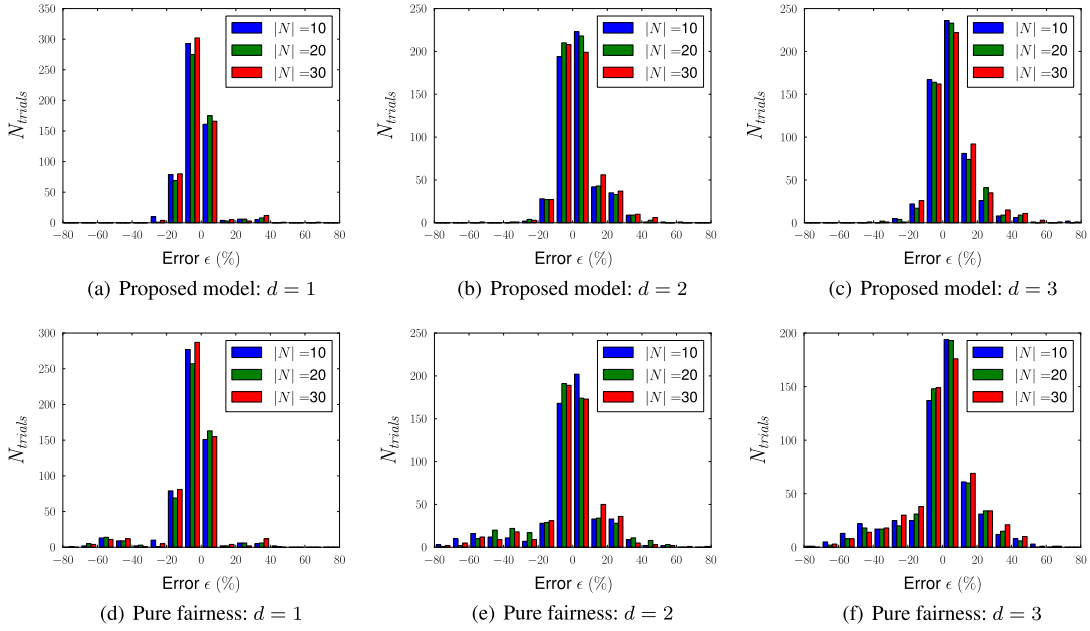
DOI: 10.1002/cpe

Table II. Experimental results on communication times prediction.

	Settings				Main testbed results (average)				Grid5000 results (average)			
	$\|N\|$	d	N_{trials}	*steps*	$\pi(\epsilon)$ (%)	$\|\epsilon\|$ (%)[a]	$\|\epsilon'\|$ (%)[b]	$\|\epsilon''\|$ (%)[c]	$\pi(\epsilon)$ (%)	$\|\epsilon\|$ (%)[a]	$\|\epsilon'\|$ (%)[b]	$\|\epsilon''\|$ (%)[c]
1	10	1	558	1.6	0.148	3.83	3.17	10.74	0.194	4.50	4.47	6.73
2	20	1	537	2.0	0.165	3.55	3.18	8.59	0.208	4.95	5.52	7.96
3	30	1	574	2.4	0.158	3.97	3.93	7.13	0.185	4.70	4.37	6.74
4	10	2	536	2.7	0.116	6.58	6.62	12.17	0.131	6.68	6.96	10.48
5	20	2	550	3.6	0.122	6.66	6.66	11.33	0.114	8.89	8.96	12.88
6	30	2	551	4.1	0.123	7.63	7.62	11.49	0.098	10.15	10.52	12.44
7	10	3	554	3.8	0.104	7.62	7.63	13.16	0.079	8.78	8.84	11.80
8	20	3	554	5.1	0.097	8.48	8.48	12.17	0.090	11.87	11.86	13.53
9	30	3	572	6.2	0.097	9.54	9.54	13.00	0.079	10.91	11.05	12.66

[a]The averaged $|\epsilon|$, based on our proposed model, for randomly generated communication patterns.
[b]The averaged $|\epsilon'|$, based on our proposed model, for patterns with asymmetric communication.
[c]The averaged $|\epsilon''|$, based on fairness estimation, for patterns with asymmetric communication.

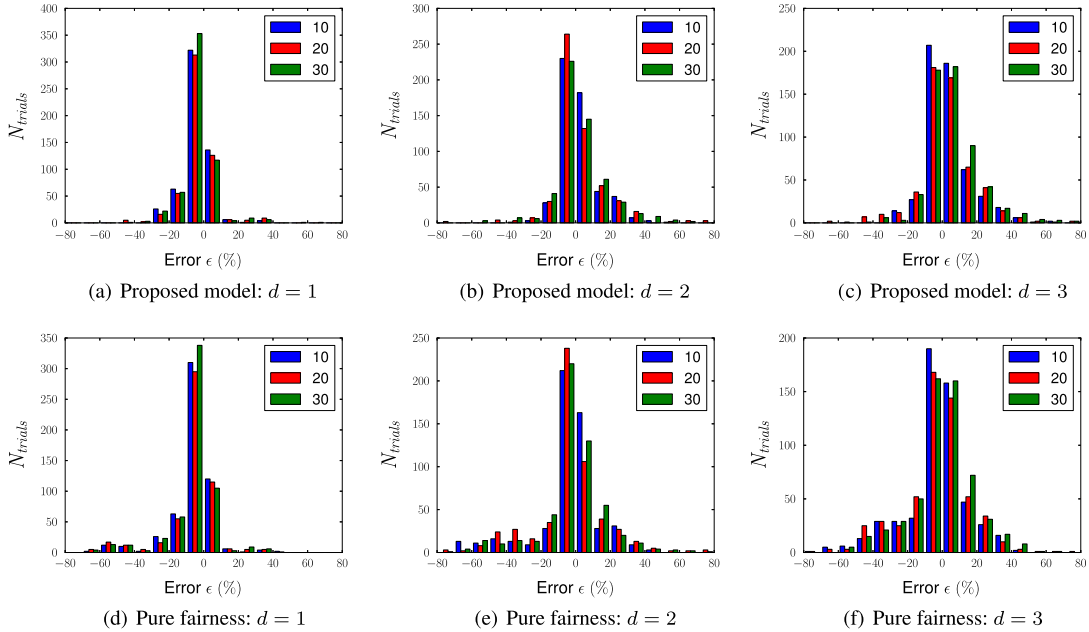
DOI: 10.1002/cpe



(a) Proposed model: $d = 1$ (b) Proposed model: $d = 2$ (c) Proposed model: $d = 3$

(d) Pure fairness: $d = 1$ (e) Pure fairness: $d = 2$ (f) Pure fairness: $d = 3$

Figure 6. The histogram of errors on communication times prediction.



(a) Proposed model: $d = 1$ (b) Proposed model: $d = 2$ (c) Proposed model: $d = 3$

(d) Pure fairness: $d = 1$ (e) Pure fairness: $d = 2$ (f) Pure fairness: $d = 3$

Figure 7. The histogram of errors on communication times prediction on Grid5000.

of data, as separated by horizontal lines in Table II. Our explanation is that an increase in d increases the communication interference on the shared Ethernet medium, which degrades the predictability on communication times in hierarchical networks.

In the last two columns, for those communication patterns with bidirectional communications on the shared network resources, we compare the averaged errors of our proposed model ϵ' and prediction on the basis of pure fairness property ϵ''. Our proposed model shows a much better accuracy in prediction on both platforms for experiments.

DOI: 10.1002/cpe

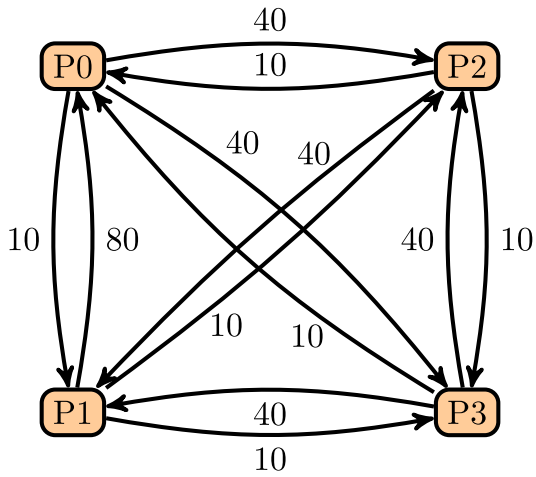


Figure 8. The communication pattern of the Alltoallv experiment. Weights show message sizes in MB.

7.3. Experimental results: collective communication

In addition to the experimental results where we analyze the impact of the asymmetric property using a large number of randomly generated communication patterns, we have also experimented with communication patterns implementing different MPI collective communication operations. The design of one of those experiments can be seen in Figure 8. It implements an $MPI\_Alltoallv$ functionality and is conducted on our main testbed. Four nodes from one rack are used. We measure the execution time of our implementation and compare it with predictions given by the symmetric and asymmetric models. Again, we set the number of iterations to 2000 in our experiment and stop if the width of the 95% CI becomes 2%, or less, of the average measured time.

The measured execution time is 1.095. The time predicted by the asymmetric model is 1.106, and by the model with pure fairness property is 0.851. It can be seen that our asymmetric model predicts the execution time with the error of 1%, while the prediction error of the model with pure fairness property is 20%. This quick experiment demonstrates the importance of the use of the asymmetric model for performance analysis of irregular collective communications, as its accuracy in this case is by the order of magnitude higher than that of the model with pure fairness property.

8. CONCLUSIONS AND FUTURE WORK

In this article, we derive an 'asymmetric network property' on TCP layer for concurrent bidirectional communications on Ethernet clusters and develop a communication model to characterize the communication times accordingly. We conduct statistically rigorous experiments to show that our model can be used to predict the communication times for simultaneous point-to-point MPI operations on resource-constrained networks quite effectively. In particular, we show that if the asymmetric network property is excluded from the model, the accuracy of the communication time prediction drops significantly. Our observation of the performance degradation caused by the asymmetric network property suggests that some part of the software stack below TCP layer in COTS clusters needs targeted tuning, which has not yet attracted any attention in literature.

As the future work, we plan to generalize our model for more complex network topologies and study how MPI collective communications (e.g., *MPI_Alltoallv* as studied in Section 7.3) could benefit from the proposed communication model from the optimization of the underlying point-to-point communication. We would also like to investigate how the asymmetric network property can be tuned below TCP layer in Ethernet networks.

DOI: 10.1002/cpe

ACKNOWLEDGEMENTS

The authors would like to thank anonymous reviewers for their in-depth review and constructive comments and thank Dr. Kiril Dichev and Dr. Konstantinos Katrinis for useful discussions. The research in this article was supported by Irish Research Council for Science, Engineering and Technology (IRCSET) and IBM, grant number IRCSET-IBM-2010-06.

REFERENCES

1. Bell G, Gray J, Szalay A. Petascale computational systems. *Computer* 2006; **39**:110–112. DOI: 10.1109/MC.2006.29. (Available from: http://dl.acm.org/citation.cfm?id=1110638.1110681) [Accessed on 24 January 2014].
2. Hockney RW. The communication challenge for MPP: Intel Paragon and Meiko CS-2. *Parallel Computation* 1994; **20**:389–398. (Available from: http://dx.doi.org/10.1016/S0167-8191(06)80021-9) [Accessed on 24 January 2014].
3. Culler D, Karp R, Patterson D, Sahay A, Schauser KE, Santos E, Subramonian R, von Eicken T. LogP: towards a realistic model of parallel computation. *SIGPLAN Notices* 1993; **28**:1–12. DOI:10.1145/173284.155333.
4. Gokhale M, Cohen J, Yoo A, Miller WM, Jacob A, Ulmer C, Pearce R. Hardware technologies for high-performance data-intensive computing. *Computer* 2008; **41**(4):60 –68. DOI: 10.1109/MC.2008.125.
5. Johnston WE. High-speed, wide area, data intensive computing: a ten year retrospective. *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing*, HPDC '98, Chicago, Illinois, USA, 1998; 280–291. (Available from: http://dl.acm.org/citation.cfm?id=822083.823206) [Accessed on 24 January 2014].
6. IEEE P802.3ba 40Gb/s and 100Gb/s Ethernet Task Force. (Available from: http://www.ieee802.org/3/ba/) [Accessed on 24 January 2012].
7. Schlansker M, Chitlur N, Oertli E, Stillwell PM, Jr., Rankin L, Bradford D, Carter RJ, Mudigonda J, Binkert N, Jouppi NP. High-performance Ethernet-based communications for future multi-core processors. *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, SC '07, Reno, Nevada, USA, 2007; 37:1–37:12. DOI: 10.1145/1362622.1362672.
8. TOP500 supercomputer site. (Available from: http://top500.org) [Accessed on 24 January 2012].
9. Majumder S, Rixner S. Comparing Ethernet and Myrinet for MPI communication. *Proceedings of the 7th Workshop on Workshop on Languages, Compilers, and Run-Time Support for Scalable Systems*, LCR '04, Houston, Texas, USA, 2004; 1–7. DOI: 10.1145/1066650.1066659.
10. Zhu J, Lastovetsky A, Ali S, Riesen R. Communication models for resource constrained hierarchical ethernet networks. *11th International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar'2013)*, Lecture Notes in Computer Science 8374, Springer: Lecture Notes in Computer Science 8374, Springer, Aachen, Germany, 2013; 259–269.
11. Velho P, Schnorr L, Casanova H, Legrand A. On the validity of flow-level TCP network models for grid and cloud simulations. *ACM Transactions on Modeling and Computer Simulation* 2013; **23**(3):1–25. (Available from: http://hal.inria.fr/hal-00872476) [Accessed on 24 January 2014].
12. Hoefler T, Mehlan T, Mietke F, Rehm W. LogfP - a model for small messages in InfiniBand. *Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS), PMEO-PDS'06 Workshop*, Rhodes Island, Greece, 2006; 1–6.
13. Alexandrov A, Ionescu MF, Schauser KE, Scheiman C. LogGP: incorporating long messages into the LogP model – one step closer towards a realistic model for parallel computation. *Proceedings of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '95, Santa Barbara, California, USA, 1995; 95–105. DOI: 10.1145/215399.215427.
14. Ino F, Fujimoto N, Hagihara K. Loggps: a parallel computational model for synchronization analysis. *SIGPLAN Notices* 2001; **36**(7):133–142. DOI: 10.1145/568014.379592.
15. Hoefler T, Schneider T, Lumsdaine A. LogGOPSim - simulating large-scale applications in the LogGOPS model. *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ACM, Chicago, Illinois, USA, 2010; 597–604.
16. Lastovetsky A, Mkwawa I, O'Flynn M. An accurate communication model of a heterogeneous cluster based on a switch-enabled ethernet network. *Proceedings of the 12th International Conference on Parallel and Distributed Systems*, ICPADS'2006, Minneapolis, Minnesota, USA, 2006; 15–20. DOI:10.1109/ICPADS.2006.24.
17. Lastovetsky A, Rychkov V, O'Flynn M. Accurate heterogeneous communication models and a software tool for their efficient estimation. *International Journal of High Performance Computing Applications* 2010; **24**:34–48. DOI:10.1177/1094342009359012.
18. Lastovetsky A, O'Flynn M. A performance model of many-to-one collective communications for parallel computing. *Proceedings of the 21st International Parallel and Distributed Processing Symposium*, IPDPS'2007, Long Beach, California USA, 2007; 1–8. DOI:10.1109/IPDPS.2007.370574.
19. Lastovetsky A, O'Flynn M, Rychkov V. Optimization of collective communications in heterompi. *Proceedings of the 14th European PVM/MPI Users' Group Meeting*, EuroPVM/MPI'07, Paris, France, 2007; 135–143.
20. Martinasso M, Méhaut JF. A contention-aware performance model for HPC-based networks: a case study of the InfiniBand network. *Proceedings of Euro-Par 2011*, Bordeaux, France, 2011; 91–102.
21. Zhang L, Shenker S, Clark DD. Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic. *SIGCOMM Computer Communication Review* 1991; **21**(4):133–147. DOI: 10.1145/115994.116006.

22. Heusse M, Merritt SA, Brown TX, Duda A. Two-way tcp connections: old problem, new insight. *SIGCOMM Computer Communication Review* 2011; **41**(2):5–15. DOI:10.1145/1971162.1971164.
23. Bedaride P, Degomme A, Genaud S, Legrand A, Markomanolis G, Quinson M, Stillwell Lee, Suter F, Videau B. Toward better simulation of MPI applications on Ethernet/TCP networks. *PMBS13 - 4th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, Denver, États-Unis, 2013; 1–12 (Anglais). (Available from: http://hal.inria.fr/hal-00919507) [Accessed on 24 January 2014].
24. Choi BY, Song S, Birch N, Huang J. Probabilistic approach to switched Ethernet for real-time control applications. *Proceedings of International Conference on Real-Time Computing Systems and Applications*, Cheju Island, South Korea, 2000; 384 –388. DOI:10.1109/RTCSA.2000.896415.
25. OpenMPI site. (Available from: http://www.open-mpi.org) [Accessed on 24 January 2012].
26. Breno HL. Tuning 10Gb network cards on Linux. *Ottawa Linux Symposium*, 2009; 169–184.
27. Iperf site. (Available from: http://sourceforge.net/projects/iperf/) [Accessed on 24 January 2012].

DOI: 10.1002/cpe