A Contention-Aware Performance Model for HPC-Based Networks: A Case Study of the InfiniBand Network

Maxime Martinasso and Jean-François Méhaut

University of Grenoble, Computer Science Laboratory LIG, Grenoble, France
{maxime.martinasso,jean-francois.mehaut}@imag.fr

Abstract. Multi-core clusters are cost-effective clusters largely used in high-performance computing. Parallel applications using message passing as a communication mechanism may introduce complex communication behaviours on such clusters. By sending and receiving data simultaneously from and to several nodes, parallel applications create concurrent accesses to the resources of the network. In this paper, we present a general model that expresses network resource sharing characterised by a dynamic contention graph. The model is based on a linear system weighted by bandwidth distribution factors called penalty coefficients that are specific to a network technology. We propose a method to solve the linear system and present an analysis to determine penalty coefficients on InfiniBand technology. We use complex network conflicts to assess the ability of the model to predict with low errors.

Keywords: Contention model, performance prediction, InfiniBand.

1 Introduction

High-Performance platforms are dedicated to execute complex scientific applications with the focus on the highest possible performance achievable. In High-Perfomance Computing (HPC) industrial sector software performance has become a keyword for developing such platforms. Reaching optimum performance for end-user applications on clusters is a difficult task that requires the use of a wide range of techniques from specialised algorithms to tuned runtime libraries. This task can be helped by several tools [8][6] that trace application events that access hardware or software resources of the platform. Nevertheless, strong expertise is required to understand and to analyse the relationship between these events and application performance. The analysis of performance-loss may be simplified if techniques exist to predict application performance and the gain obtained by using such hardware or software. Performance prediction is not only a topic of high interest for the HPC community but is also very challenging.

Clusters of multi-core computers can be seen as platforms providing resources to an application such as computation power, memory, storage and network. The network resource is an important element in performance analysis as it is a slow component of the platform. High performance networks or system-area networks, such as InfiniBand, have been developed to reduce this gap. They are an important architectural element during the design of a cluster. Even for highly specialised network hardware operated by multi-core computers, concurrent accesses are inevitable and degrade network performance. Modelling concurrent accesses and their performance degradations is a step forward to enhance performance improvements of scientific applications.

Each HPC-based network executes its own flow control mechanism when concurrent accesses occur. For instance, InfiniBand provides a credit-based flow-control mechanism on the buffer availabilities to ensure the integrity of the connection. The diversity of flow control mechanisms increases the complexity in identifying models that predict communication times. Models are either too simplistic or too tailored for a particular technology.

In this article we present a method to predict elapsed times of communications that take place concurrently on high performance networks. Our approach generalises the concept of concurrent accesses by introducing the notion of dynamic contention graphs. By creating artificial contention graphs on a network we demonstrate the feasibility of solving a linear system which calculates the elapsed times of every communication. Moreover, our method can be applied on different networks resulting in accurate models. We apply it to a widely popular HPC network: InfiniBand.

The remainder of this paper is organised as follows: in Section 2, we present the background of our study. In Section 3, we introduce our methodology based on dynamic contention graph and a sequence of linear equations. Section 4 presents a model dedicated to InfiniBand network. In Section 5, we validate its accuracy. In Section 6, we conclude and present our future directions.

2 Background

Performance prediction of communication over networks is an extensively researched topic. Contention-free models are generally based on a linear equation [10]. Such models predict the communication delay by multiplying the inverse of the bandwidth with the message size to transfer and by adding a constant value (the network latency). More refined models [3][1] decompose a similar linear equation into more parameters that express the characteristics of a communication. Such parameters can be measured and their values can depend on the message size [14]. These models do not consider contention and resource sharing.

In the aforementioned models, parameter values used to solve a linear equation are taken independently of the resource availability. In [2] previous models are enhanced by introducing a queueing system to represent contention effects. A vast collection of new parameters were introduced for this model. These parameters are difficult to evaluate and limit the applicability of such models.

In [11] contention effects are modelled by a coefficient that divides the bandwidth availability by the number of communications sharing the same link. The presented experiments are limited to mono-processor computers. Therefore, on a node, concurrent requests of network resources originating from different cores are not considered. The model avoids de facto a large set of contention cases on multi-core technology.

2.1 Elements Influencing Network Contention

Our objective is to propose a methodology to identify models that can accurately predict communication times in a congested network. Many elements should be considered to achieve this goal:

- The first element that we consider is the dynamic of contention behaviours over the network. Contention behaviours are directly linked to the application that is executed on the cluster. The application is responsible for triggering communications that may interfere with each other to access the network resources leading to time delays.
- The second element is the network technology. Each network has its own flow control that regulates the concurrent accesses to its resource. They directly affect and characterise delays caused by contention behaviours.
- The third element is the topology of the network. Many cluster topologies exist, for instance, nodes may be linked to a single switch that may be linked to other switches. Depending on how many network components a communication passes through, its delay depends on the availability of each of the network components.

In our methodology we will focus on the first two elements. The third element, i.e. the topology of the network, will be restricted to one switch connecting several nodes. In [9] we were focused on HPC clusters with dual-core computers and contention graphs based on a mesh of communications. In this paper, we generalise this work to any contention graph and any network technology.

3 Methodology

To address the dynamic of the contention behaviour we introduce the notion of a contention graph that characterises the dynamic usage of network resources by an application. In addition, we add weights named penalty coefficients to a contention graph. The penalty coefficients are factors mirroring the effects on performance of network control flow mechanisms that distribute network bandwidth. We present a technique to determine penalty coefficients by only referring to contention graphs. Finally, our model solves a sequence of linear equations for each communication. Each linear equation represents the communication as part of the dynamic of a contention graph. Linear equations are weighted with penalty coefficients.

3.1 Dynamic Contention Graph

Network contention is characterised by the quantity of data to transfer. Communication delays may change due to other communications present on the network. This variation depends on the sources and destinations, as well as the

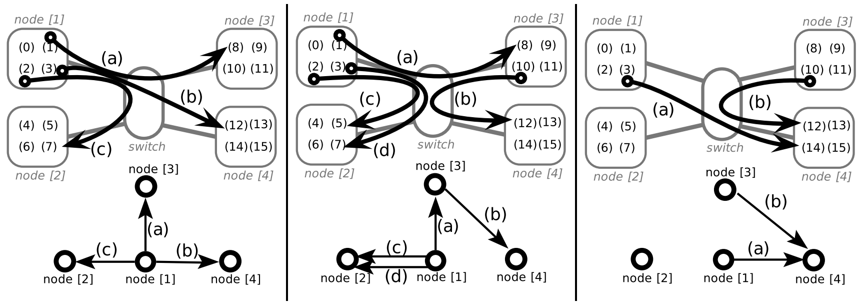


Fig. 1. Examples of an MPI application with 16 ranks spawn on 4 nodes with 4 cores each. During the run of the application, communications start and end creating several contention graphs that are modelled over a dynamic directed graph K.

message size of the communications. For instance, two communications sending data from the same node or to an identical node may create different contention effects on the network by an overlapping demand of network resources. Since the contention effects are related to the communications that are being processed, we introduce the notion of a dynamic contention graph.

A dynamic contention graph is a dynamic directed graph, denoted K, on which the nodes are the cluster nodes and the edges are the communications between the nodes. A contention graph may have parallel edges, however, as our analysis does not consider internal communications, a contention graph does not have any self-loop edges. A communication can start or end at any time modifying the current contention graph. In the same manner, K is modified when an edge is added or removed. In order to reflect the fact that K evolves in the course of different steps, we will denote a contention graph at step s as K_s. Therefore, a step of a contention graph is a static directed graph. Examples of contention graphs and their representation with K are presented in Figure 1.

On multi-core computers and also often on HPC platforms, one MPI task is spawned on one core. Therefore, we bind the maximum degree of K, i.e. the maximum number of simultaneous outgoing and ingoing communications from/into a compute node, to the maximum number of cores per node.

3.2 Sequence of Linear Models

During the execution of an application, communications are creating contention graphs, which may be divided into several steps. While transferring data, one communication can be part of several contention graph steps. We modelled the elapsed time of a specific communication during a specific contention graph step by a single linear equation. In the event that this communication is part of more than one step, its total elapsed time will be equal to the sum of all linear equations modelled in each step. The elapsed time T_i of a communication i of K sending m_i bytes can be approximated by the following formula:

$$T_i = \sum_{j \in K_s} t_{i,j} = \alpha \sum_{j \in K_s} \rho_{i,j} * m_{i,j} \tag{1}$$

$$\sum_{j \in K_s} m_{i,j} = m_i \tag{2}$$

with K_s is a contention graph at step s, $t_{i,j}$ the time spent by the communication i in K_s with $t_{i,j} = \alpha * \rho_{i,j} * m_{i,j}$, α is the inverse of the network bandwidth, $m_{i,j}$ the amount of bytes transferred during $t_{i,j}$ and $\rho_{i,j}$ ($\rho_{i,j} \geq 1$) a coefficient called penalty, which acts as the delay caused by other communications in K_s. To reflect the progress of communications over steps of K, we introduce the following relation between $t_{i,j}$:

$$t_{1,1} = T_1 \text{ and for } \forall j \text{ such that } i \geq j > 1,\ \ t_{i,j} = T_i - T_{i-1} \tag{3}$$

The problem of the evaluation of T_i resides in the number of unknown variables present in the model. If a communication is part of s contention graph steps, the model has $2 * s$ unknowns to be determined: s penalty coefficients and s sizes of the data transferred during each step. However, if the penalty coefficients are known for every K_s and only the initial message size m_i of each communication i is known, then it is possible to calculate T_i for all i. By solving T_i in an increasing order of i, one knows from (3) the value of $t_{i,j}$ with $j < i$ and thus the value of $m_{i,j}$. From (2) one can deduce the size $m_{i,i}$ and then $t_{i,i}$ leading to the result T_i by (1). In the next subsection we introduce a method to approximate the penalty coefficients.

3.3 Approximation of Penalty Coefficients

Penalty coefficients are factors which divide bandwidth and thus increase communication delays. Their values depend on the underlying flow control that manages the over-subscription of a resource. Therefore, an identical contention graph step generates different penalty values depending on the network technology.

Penalty coefficients are directly related to the shape of a contention graph. In our model we consider that one contention graph step creates a unique set of penalties. Since dedicated HPC networks are highly deterministic, this consideration does not appear to be a strong restriction. However, if we consider a network topology with several switches, this condition should be revised.

To determine penalty coefficients of a contention graph we use real experiences on a cluster. We have developed a simple MPI benchmark that creates contention graphs. This benchmark spawns as much MPI processes as available cores. An MPI process is either only sending data or receiving data. By selecting a subset of processes that send or receive data and by binding them to an MPI communicator we can create a contention graph. We configure our benchmark to create a specific kind of contention graph that we call static-synchronous contention graph. In a static-synchronous contention graph communications start at the same time and have a same amount of data to transfer. This benchmark gathers statistical data over elapsed times of every communication.

To approximate penalty coefficients, we use the reverse approach as presented in the previous subsection 3.2 to solve T_i. By knowing an approximation of every T_i of K and the initial size $m_i = m$ of every communication we are able to approximate $\rho_{i,j}$ for any static-synchronous contention graph. The maximum number of steps of such graph is equal to the number of its communications.

A static-synchronous contention graph of s steps has a number of communications that decreases while the number of steps increases. Static-synchronous graphs with only one communication have a penalty coefficient equal to 1. A static-synchronous graph of two communications implies a linear system which is directly solvable. Therefore, it is possible to calculate penalty coefficients of any static-synchronous graphs with two communications. Static-synchronous graphs of three communications have, in the general case, three steps. We are interested in calculating penalty coefficients of the first step: $\rho_{1,1}$, $\rho_{2,1}$, $\rho_{3,1}$. To compute these penalty coefficients we need to know the penalty coefficients of its two other steps: $\rho_{2,2}$, $\rho_{2,3}$ and $\rho_{3,3}$. The last step has only one communication remaining, therefore $\rho_{3,3} = 1$. For the values of $\rho_{2,2}$ and $\rho_{2,3}$, we can use the penalty coefficients of a static-synchronous graph of two communications that represents the graph at this step. Therefore, it becomes possible to compute $\rho_{1,1}$, $\rho_{2,1}$, $\rho_{3,1}$ at the first step by solving the system below:

$$\begin{cases} \rho_{1,1} = T_1/(\alpha * m) \\ \rho_{2,1} = T_1/(\alpha * m_{2,1}) \text{ with } \begin{cases} m_{2,1} = m - m_{2,2} \\ m_{2,2} = (T_3 - T_2)/(\alpha * \rho_{2,2}) \end{cases} \\ \rho_{3,1} = T_1/(\alpha * m_{3,1}) \text{ with } \begin{cases} m_{3,1} = m - m_{3,2} - m_{3,3} \\ m_{3,2} = (T_3 - T_2)/(\alpha * \rho_{3,2}) \\ m_{3,3} = (T_2 - T_1)/(\alpha * \rho_{3,3}) \end{cases} \end{cases}$$

By generalising this method, we can recursively determine penalties for a n-communication static-synchronous graph. For its $(n-1)$ last steps we can apply the penalty coefficients of $(n-1)$ static-synchronous graphs having respectively $(n-1)$ to 1 communications. The shape of each of these static-synchronous graphs should correspond to the shapes of the respective steps in the n-communication graph to which their penalties are applied to.

In our model we assume that a step of a contention graph is comparable to a static-synchronous graph, in other words, that the transition between steps, i.e. the network reconfiguration of its contention behaviour, is immediate. Within this assumption, when a new contention graph step starts, every communication enters the new step at the same time (synchronous) and their contention behaviour (i.e. penalty coefficients) will not change (static) until the step ends and a following step starts. Therefore, we are able to approximate every penalty coefficient of each step of a contention graph.

By analysing a set of hundreds of contention graphs and their penalty coefficients, we present a model that approximates penalties based on the shape of a contention graph. This model is applicable for InfiniBand network.

4 Modelling Penalty Coefficients over InfiniBand

In the top 500 list, about 35% of the clusters were using InfiniBand[5] in 2009, which has increased to about 42% in 2010. Research on InfiniBand is mainly covering routing strategies in order to increase application performance.

In [13] the impact of static routing in multistage InfiniBand networks is presented. The study focuses on evaluating the available bisection-bandwidth between nodes obtained through different switches. Similarly to [11], it considers concurrent communications between different pairs of nodes and not concurrent communications that are going to or are initiated from different cores of one node. In our study, however, we demonstrate that such concurrent communications create significant contention delays even with a single switch.

An analysis based on a LogP model for small message size communication performance over InfiniBand was also discussed in [12].

4.1 InfiniBand Network Testbed

Our experiments were carried out on a cluster based on 34 nodes with 8 cores per node. Each node is connected with a InfiniBand Mellanox Technologies MT26418 card to 3 Voltaire Grid Director 9024 switches. Our MPI implementation is OpenMPI 1.2.7 using the OpenFabrics low-level library.

Routing paths in InfiniBand network are statically fixed. Taking into account the routing strategy leads to complex modelling. Our intention is to explore the possibilities in approximating penalty coefficients. In order to limit the influence of static routing strategy in our analysis we select a set of nine nodes that are connected to a unique switch. Each switch has an internal fat-tree topology that guarantees an efficient distribution of the bandwidth between the links. Thus the variance of experimented results remains low, as well as the distortion in the approximation of the penalty coefficients.

InfiniBand flow control uses a credit-based mechanism. A network card that receives data sends messages called *link credit* to inform the sender about the available buffer size remaining. If a receiver buffer is full then the destination network card stops sending *link credit* messages and the source network card will hold in a *back pressure* state delaying the entire set of its outgoing communications. The communication is resumed when sufficient buffer size is available.

4.2 Penalty Coefficients and Model for InfiniBand

Upon analysing the penalty coefficients of several hundreds of static-synchronous contention graphs we propose the model below. A contention graph step is represented by a connected directed graph $K(V, E)$ where V is the set of nodes, i.e. the cluster compute nodes, and E is the set of edges, i.e. the communications involved in the contention graph step. As usual, we use $\delta^+(v)$ to denote the outdegree of a vertex v of K and, respectively, $\delta^-(v)$ to denote the indegree of a vertex v of K. We use $V^+(v) \subset V$ as the out-neighbourhood of vertex v (set of vertices being a destination vertex of an edge outgoing from v). We extend the notion of $V^+(v)$ to a set of out-neighbourhood edges of an edge $e = (s, d)$

as $E^+(e) = \{(s,d') : d' \in V^+(s)\} \subset E$. We also consider a specific set of edges being the edges ingoing to the destination vertex d of an edge (s,d). This subset is defined as follow: $E^{\bowtie}(e) = \{(s',d) : s' \neq s\} \subset E$. The penalty coefficient $\rho(e)$ of an edge $e = (s,d) \in E$, being oriented from s to d, is calculated as follows:

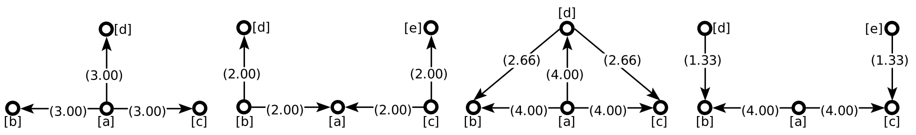
$$\rho(e) = \max_{E^+(e)} \{\delta^+(s) + k(e)\} \tag{4}$$

with $k(e)$ defined as follow:

(a) $k(e) = 0$ if $(\delta^-(d) \leq \delta^+(s)$ and $\delta^+(s) = \delta^+(s'))$ or $\delta^-(d') = 0$ such that $e' = (s',d) \in E^{\bowtie}(e)$ and $e'' = (s,d') \in E^+(e)$ respectively;

(b) $k(e) = \frac{1}{max(\rho(e')-1)}$ if $\delta^+(s) = 1$ with $e' = (s',d) \in E^{\bowtie}(e)$;

(c) $k(e) = \displaystyle\sum_{\forall e'=(s,d') \in E^+(e) \wedge \forall e''=(s'',d') \in E^{\bowtie}(e')} \frac{1}{\delta^+(s'')}$ otherwise.



item (a) item (a) item (c) item (b) & (c)

Fig. 2. Penalty coefficients of four contention graphs

In Figure 2 we apply the model above on four contention graphs. In the first graph, the outgoing degree of node [a] is 3, $\delta^+(a) = 3$. Nodes [b], [c] and [d] have an ingoing degree of 1, $\delta^-(b) = \delta^-(c) = \delta^-(d) = 1$ therefore from item (a) k is nullified for all edges. From (4) each edge has a penalty of 3. In the second graph k is also nullified, because for both edges (b,a) and (c,a): $\delta^-(a) = \delta^+(b) = \delta^+(c)$. For the third graph neither item (a) or item (b) is applicable. The penalty coefficients of edges (a,b) (a,c) (a,d) are then $max(\delta^+(a) + \frac{2}{\delta^+(d)}, \delta^+(a) + \frac{2}{\delta^+(d)}, \delta^+(a)) = 4$. Similarly, edges (d,b) and (d,c) have each a penalty coefficient equal to $max(\delta^+(d) + \frac{2}{\delta^+(a)}, \delta^+(d) + \frac{2}{\delta^+(a)}) = 2.66$. Finally, the last example combines the item (b) and item (c). Edges (a,b) and (a,c) have a penalty of 4 from item (c). Item (b) is applicable for edges (d,b) and (e,c) as $\delta^+(d) = \delta^+(e) = 1$. Their penalty coefficients are equal to $1 + \frac{1}{max(\rho((a,b)),\rho((a,c))-1} = 1.33$.

Discussion: In our model, the contention factor depends mainly on the number of outgoing communications from a node. Inside a node, the network card is the first network component that is shared. The network card allocates a first distribution of bandwidth capacity. We model this distribution as a fair distribution among the communications (4). In addition, bandwidth may also be reduced when communications share the same destination node. On a receiving card of a node, the buffer allocation depends on the bandwidth requested by each ingoing communication to the node. Therefore, in our model we characterise this value by the function k that adds a factor proportional to the bandwidth requests made by others communications having the same destination node (item (c)). However, k can be nullified if n communications that are sending data to the same destination also share their source node with n outgoing communications. Alternatively, k can also be nullified if a communication does not suffer contention on the receive node (item (a)). Finally, if a network card is in a *back pressure* mode (the receiver blocks the sender) then its outgoing communications are delayed. This effect is modelled by choosing the maximum delay over all communications outgoing from the same node. Furthermore, if a communication with a full bandwidth allocation reaches a node, we model its allocated bandwidth on the receive node as the maximum bandwidth available (item (b)).

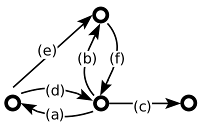
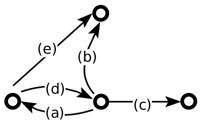
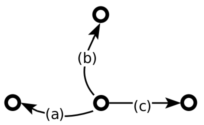
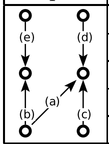
dynamic graph steps	com	penalty	data remaining [o]	time spent [s]
	a	3.5	11983700	0.0160590
	b	3.5	11983700	0.0160590
	c	3.5	11983700	0.0160590
	d	3.33	11534300	0.0160590
	e	3.33	11534300	0.0160590
	f	1.5	0	0.0160590
	a	3.5	4294170	0.0297984
	b	3.5	4294170	0.0297984
	c	3.5	4294170	0.0297984
	d	2.33	0	0.0297984
	e	2.33	0	0.0297984
	f	-	0	0.0160590
	a	3.0	0	0.0363749
	b	3.0	0	0.0363749
	c	3.0	0	0.0363749
	d	-	0	0.0297984
	e	-	0	0.0297984
	f	-	0	0.0160590

Fig. 3. Example to determine communication times of a dynamic contention graph. All communications have an identical size of 20MB and start at the same time.

5 Examples and Validation

Before evaluating the accuracy of our model, we will introduce one example to calculate the elapsed time of one arbitrary chosen contention graph. We consider the dynamic contention graph presented in Figure 3. This dynamic contention graph has six communications and for simplicity all communications start at the same time and have a data transfer size of 20MB. By executing our benchmark (presented in 3.3), we measure an effective bandwidth of 1.82 GB/s on our testbed evaluating $\alpha = 5.105 \times 10^{-10}$. A single communication of 20MB under no contention lasts for 0.010706 seconds.

By solving the linear system step by step we deduce the time of each communication. The first communication to end is communication f after a time of 0.016059 seconds. During that time the communications a, b and c have sent 8987820 bytes. Then the contention graph goes to a new step (without f), in which the remaining data size per communication is deducted and new penalty coefficients are applied. The second step ends when communications d and e terminate at time 0.0297984. The final step has only three communications each of them having 4294170 bytes remaining. At this step the penalty coefficient per communication is 3, i.e. each communication accesses one third of the available bandwidth. The last step ends at time 0.0363749. By executing our benchmark and creating this arbitrary contention graph, we measure the time of each communication on our testbed. We compare the measured times to the predicted times leading to an absolute error of 3% for communications a, b and c, an absolute error of 1% for d, e and 2% for f.

Graph A		$\rho(e)$	T_p	T_m	err.
	a	5, 4, 2	0.036132	0.036328	0%
	b	5, 4, 2	0.036132	0.036326	0%
	c	2.5, 2.5	0.026765	0.027653	-3%
	d	2.5, 2.5	0.026765	0.027651	-3%
	e	1.5	0.013382	0.013413	0%

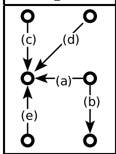
Graph B		$\rho(e)$	T_p	T_m	err.
	a	5, 2	0.043894	0.045236	-2%
	b	5, 2	0.043894	0.045228	-2%
	c	3.5	0.037471	0.040073	-6%
	d	3.5	0.037471	0.040072	-6%
	e	3.5	0.037471	0.040071	-6%

Fig. 4. Examples of model validation: $\rho(e)$ is the list of penalty coefficients, T_p is the predicted time, T_m is the measured time and err. is the relative error.

In Figure 4, we present the measured times and predicted times of two other dynamic contention graphs with identical communication properties as in aforementioned example. Graph A shows communications a and b starting from the same node having different conflicts on their receive nodes. Communications of graph B are mainly congested in an ingoing conflict of four communications. For these two graphs, our model accurately predicts elapsed times.

The previous examples display a set of heterogeneous communication elapsed times. These times are not simple to determine and their variation shows the complexity in accurately predicting them. In a last validation analysis of our model, we investigate the model prediction for a set of dynamic contention graphs that are derived from random directed graphs. These dynamic contention graphs are categorised in groups following their number of edges. Figure 5 displays, per group, the measured times of their communications. For instance, we measured 60 contention graphs of 5 edges leading to evaluating 300 communication times. Figure 5 also displays, per group, the maximum value of the absolute average error per graph and the maximum value of the absolute error over all communications. The absolute average error for a graph is the average of the absolute

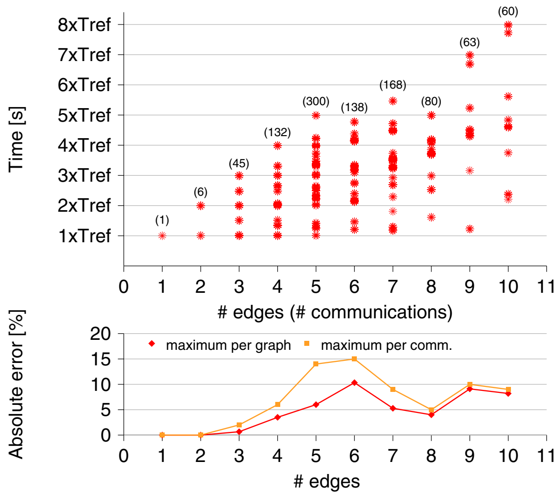


Fig. 5. Validation on several contention graphs. $T_{ref} = 0.010706$ is the reference time of a contention-free communication. For graphs with 1, 2 and 3 edges, times are rational factor of T_{ref}. However, for a higher number of edges, communication times are more largely spread. Our predictions are within an acceptable range of errors.

error among its communications. The figure shows that we accurately predicted communication times with an acceptable error below 15% over a total of nearly 1000 measured communications.

6 Conclusion and Future Work

In this paper we introduce a new methodology for assessing contention over HPC-based networks. This method models congestion over a network by a contention graph and a linear system weighted by delay factors called penalty coefficients. We propose a technique that determines penalty coefficients from experimented contention graphs. We applied this technique on an InfiniBand network. By analysing the penalty coefficients of contention graphs we approximate their values only by referring to the contention graph. Finally, we accurately predict the communication times of nearly a thousand communications requiring only one parameter: the effective bandwidth of the network.

We applied our methodology on a network topology restricted to one switch. Our future work will focus on extending our model to consider network topology and to identify models for large-scale applications running on hundreds of cores. In addition, we will incorporate our model into an existing simulator [4], which replays events of an application and determines dynamic contention graphs. Besides, all-to-all collective operations generate congested communications [7]. It will be interesting to model their performance following our methodology.

We consider applying this methodology on other network technologies. By modelling several networks, it will be possible to compare application performance on those networks.

Acknowledgement. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see `https://www.grid5000.fr`).

References

1. Alexandrov, A., Ionescu, M., Schauser, K., Scheiman, C.: LogGP: Incorporating Long Messages into the LogP model for Parallel Computation. Journal of Parallel and Distributed Computing 44(1), 71–79 (1997)
2. Moritz, C.A., Frank, M.I.: LoGPC: Modeling Network Contention in Message-Passing Programs. IEEE Transactions on Parallel and Distributed Systems 12(4), 404–415 (2001)
3. Culler, D., Karp, R., Patterson, D., Sahay, A., Santos, E., Schauser, K., Subramonian, R., von Eicken, T.: LogP: a practical model of parallel computation. Commun. ACM 39(11), 78–85 (1996)
4. Casanova, H., Legrand, A., Quinson, M.: SimGrid: a Generic Framework for Large-Scale Distributed Experiments. In: 10th IEEE International Conference on Computer Modeling and Simulation (2008)
5. InfiniBand Trade Association: InfiniBand Architecture Specification, Release 1.2.1
6. Intel Corporation: Intel Trace Analyzer and Collector 8.0 Reference Guide
7. Steffenel, L.A., Martinasso, M., Trystram, D.: Assessing Contention Effects on MPI_Alltoall Communications. In: Cérin, C., Li, K.-C. (eds.) GPC 2007. LNCS, vol. 4459, pp. 424–435. Springer, Heidelberg (2007)
8. Geimer, M., Felix, W., Wylie, B.J.N., Ábrahám, E., Becker, D., Mohr, B.: The Scalasca performance toolset architecture. Concurrency and Computation: Practice and Experience 22(6), 702–719 (2010)
9. Martinasso, M., Méhaut, J.-F.: Model of concurrent MPI communications over SMP clusters. Tech. Rep. 00071352, HAL-INRIA (2006)
10. Hockney, R.W.: The Communication Challenge for MPP: Intel Paragon and Meiko CS-2. In: Parallel Computing, vol. 20, pp. 389–398. North-Holland, Amsterdam (1994)
11. Kim, S.C., Lee, S.: Measurement and Prediction of Communication Delays in Myrinet Networks. Journal of Parallel and Distributed Computing 61(11), 1692–1704 (2001)
12. Hoefler, T., Mehlan, T., Mietke, F., Rehm, W.: LogfP - A Model for small Messages in InfiniBand. In: Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium, IPDPS (2006)
13. Hoefler, T., Schneider, T., Lumsdaine, A.: Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks. In: Proceedings of the 2008 IEEE International Conference on Cluster Computing, pp. 116–125 (2008)
14. Kielmann, T., Bal, H.E., Verstoep, K.: Fast Measurement of LogP Parameters for Message Passing Platforms. In: IPDPS 2000: Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing, pp. 1176–1183 (2000)