A PCIe Congestion-Aware Performance Model for Densely Populated Accelerator Servers

Maxime Martinasso*, Grzegorz Kwasniewski†, Sadaf R. Alam*, Thomas C. Schulthess*‡§, Torsten Hoefler†

*Swiss National Supercomputing Centre, ETH Zurich,
6900 Lugano, Switzerland

†Department of Computer Science, ETH Zurich,
Universitätstr. 6, 8092 Zurich, Switzerland

‡Institute for Theoretical Physics, ETH Zurich,
8093 Zurich, Switzerland

§Computer Science and Mathematics Division,
Oak Ridge National Laboratory, USA

***Abstract*—MeteoSwiss, the Swiss national weather forecast institute, has selected densely populated accelerator servers as their primary system to compute weather forecast simulation. Servers with multiple accelerator devices that are primarily connected by a PCI-Express (PCIe) network achieve a significantly higher energy efficiency. Memory transfers between accelerators in such a system are subjected to PCIe arbitration policies. In this paper, we study the impact of PCIe topology and develop a congestion-aware performance model for PCIe communication. We present an algorithm for computing congestion factors of every communication in a congestion graph that characterizes the dynamic usage of network resources by an application. Our model applies to any PCIe tree topology. Our validation results on two different topologies of 8 GPU devices demonstrate that our model achieves an accuracy of over 97% within the PCIe network. We demonstrate the model on a weather forecast application to identify the best algorithms for its communication patterns among GPUs.**

***Index Terms*—Multiple GPUs, PCI-Express, performance model;**

I. Introduction

MeteoSwiss is the first national meteorological service which has chosen a computer architecture purely based on GPUs for operational numerical weather prediction. Computer nodes of this new architecture are densely populated with GPU accelerators in order to reduce time to solution of the simulations while increasing the energy efficiency of the system [1]. This new system is made of two cabinets of the Cray CS-Storm supercomputer which are exclusively dedicated to weather forecast prediction. Each cabinet consists of 12 hybrid computing nodes for a total of 96 NVIDIA Tesla K80 GPU accelerators (or 192 GPU processors) and 24 Intel Haswell CPUs. The full system delivers performance up to 360 GPU teraflops or 15 GPU teraflops per node. The climate and weather model COSMO [2] – used by MeteoSwiss – has been ported to GPU accelerators [3]. Combining densely populated accelerator nodes with COSMO allows MeteoSwiss to execute per day higher resolution simulations and a larger number of simulations with more day forecasts [1] [4].

Multiple accelerator devices in a server or compute node increase stress on both the inter and intra node communication networks. To manage concurrent memory transfers, network fabrics use a congestion control mechanism, which may impact application performance. Contemporary accelerator devices, e.g., GPGPU or Intel Xeon Phi, perform intra-node communication using the PCI-Express (PCIe) fabric. A PCIe topology is based on a tree topology with a complex congestion control mechanism performed at each node of the topology. In order to avoid or minimize performance losses in densely-populated servers, this study systematically investigated the causes of congestion specific to the PCIe fabric.

In this paper we present a congestion-aware performance model for the PCIe technology. The model is based on a technical description of the congestion control mechanism used by PCIe together with empirical results. Our performance model captures all the technology-specific phenomenons, yielding the accuracy in timing prediction of over 97%. To the best of our knowledge this is the first time that a complete performance model has been proposed for the PCIe technology. We apply the model to the COSMO weather forecast application to be able to identify the best algorithms for performing halo exchanges among the GPUs. Furthermore, the model can be used for evaluating tuning parameters [5], increasing accuracy of cost predictors used by schedulers [6], helping designing efficient application algorithms and developing densely-populated nodes of future large-scale systems.

The key contributions of our work are:

- a detailed analysis of complex congestion behaviors in PCIe fabric;
- a performance model which is capable of predicting elapsed times of communications subjected to congestion on any PCIe tree topology;
- a performance improvement of halo exchange collective operations for a weather forecast model.

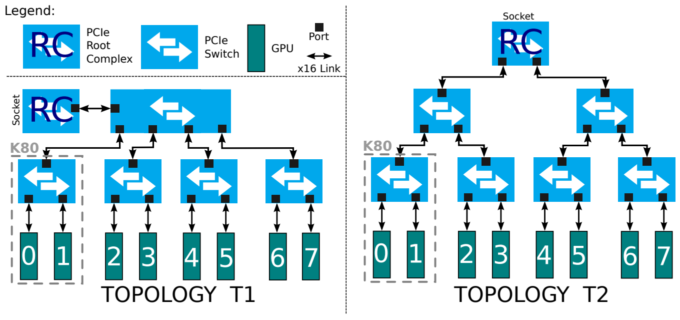


Fig. 1. Topology T1 connects 8 GPUs (4×K80) featuring four 48-lane PCIe switches and one 80-lane switch. Topology T2 connects 8 GPUs (4×K80) featuring six 48-lane PCIe switches and one root complex.

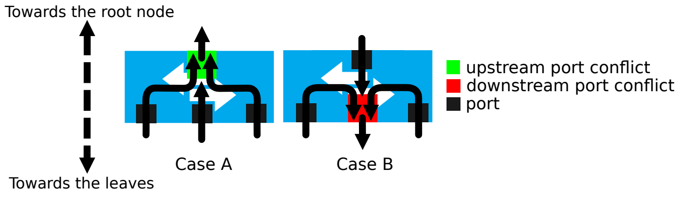


Fig. 2. Two types of conflicts: an upstream port conflict occurs when communications going towards the root node are in conflict on a port, while a downstream port conflict occurs when communications are in conflict on a port when going towards the leaves.

II. Impact of PCI-express on performance

A. *PCIe topology*

One major parameter influencing the available bandwidth is the PCIe topology. PCIe uses a tree topology where the root node or root complex communicates directly with the socket [7]. The endpoint devices are leaf nodes and PCIe switches are the remaining nodes. The depth of the tree depends on the number of ports per switch, which determines the branching factor. To ensure maximal bandwidth each link is using a x16 wide lane.

Our testbeds consist of two systems connecting 4 NVIDIA K80 using two different PCIe topologies. Both testbeds are presented in Fig. 1. An NVIDIA K80 contains two GPU processors GK210 connected by a 3-port 48-lane PCIe switch. Our first testbed, named T1, connects four K80 by using a 5-port 80-lane switch. The second testbed, named T2, uses two 3-port 48-lane PCIe switches to connect the four K80. Both switches are connected together by a root complex node. Topology T2 is the topology connecting 4 K80 GPUs on one socket of a Cray CS-Storm node, it is our reference topology whereas topology T1 is used in Section VII to validate our performance model. All switches (including the one inside the K80) are manufactured by PLX technology, which is a market leader for PCIe switches [8]. A 16-lane link is full-duplex and provides a theoretical bandwidth of 16 GB/s. By using a micro benchmark (*p2pBandwidthTestLatency*) from the CUDA samples list [9] between two GPU processors of a K80 we could assess the effective bandwidth to be 11.6 GB/s. Peer-to-Peer access is enabled for all communications in both topologies, which allows direct memory transfer from one device to another without using host memory.

Fig. 2 presents the conflict types on the ports of a switch. The direction of the communications which is either towards the root node or towards the leaves of the tree defines the type of conflict on the ports. If communications are routed to a PCIe component at a higher level, i.e., towards the root node (Fig. 2 Case A), and are in conflict on a port, the conflict is qualified to be an upstream port conflict. A conflict is qualified as a downstream port conflict if communications are reaching PCIe components at a lower level, i.e., towards the leaves of the tree (Fig. 2 Case B), and are sharing the port bandwidth.

B. *Flow control*

Switches use for each of its ports an arbitration policy to control access to its connecting links. Such arbitration handles the distribution of bandwidth among communications. A credit-based flow control mechanism allows each component to exchange credit tokens to compute the buffer availability of its link partner.

The flow control is initialized between every two PCIe component ports along the path of a transfer. Therefore, a communication can be delayed at every port it crosses depending on the load on the port.

An example of such a delay is Head-Of-Line (HOL) blocking. When multiple flows share the same input port, and one of them targets a congested link, then the bandwidth availability is reduced. In that case, the entire set of flows crossing the input port is congested, including flows that are not directed towards the congested link. In the context of PCIe technology, HOL blocking has been studied by Krishnan et al. [10]. Since the flow control mechanism does not have any information on the congested flow, it schedules all flows coming from the same input port using a fair strategy. Eventually, the congested link will take a longer time to return credits to the port, creating a credit starvation that will affect all flows crossing that port.

C. *Recent studies*

Recent studies showed the performance impact of congestion on the PCIe network. Lutz et al. [5] are interested in implementing an auto-tuning framework for stencils. Their study emphasizes the relevance of PCIe topology as a tuning parameter. Due to the negative impact on the performance of PCIe data transfers, the authors conclude that using all available GPUs is not necessarily optimal. Additionally, they point out that it is necessary to map carefully the application on GPUs in order to minimize the negative impact. Their study is based on three different topologies connecting four GPUs each.

Schaa et al. [11] propose a model to compute the performance benefits gained by increasing the number of GPUs for a scientific application. They model performance of CUDA kernels in terms of number of elements to compute divided by the number of GPUs. The impact of PCIe on performance is considered in the model by using experimental values from a 2-GPU system. The authors only consider a PCIe topology with two-end points. Extrapolation of these values, disregarding the PCIe topology, is given for a 4-GPU system.

Martinasso et al. [12] have developed a methodology to help building congestion-aware performance models. They applied it to an InfiniBand interconnect technology. In our work, we use a similar methodology, however, our performance model for PCIe requires to model the status of every port which leads to a more complex performance model.

Faraji et al. [13] show that GPU-to-GPU communication performance depends on the number of PCIe components they cross on the node topology. Moreover, they show that it is specially true for large messages. They propose a scheme to efficiently map processes to GPU to improve performance of microbenchmarks. They do not explain which features of PCIe are responsible for the performance variation or investigate complex communication patterns.

III. Problem Statement

The PCIe congestion behavior varies significantly depending on the conflicts created by communications. Message sizes, number of shared ports, crossing root complex, and HOL blocking have all significant impact on observable latency and bandwidth. To analyze the problem, we have performed a set of tests on topology T2 which is used inside the Cray CS-Storm machine.

A. Preliminary observations on GPUs

Our first investigation explores the behavior of a single GPU initiating a set of communications. Each communication has a different destination and starts simultaneously by using an asynchronous memory copy function (*cudaMemcpyPeerAsync()*) from the CUDA [14] libraries. By measuring latency of a set of overlapping communications, we observe that even by using an asynchronous memory copy function, they are serialized in the First Come First Serve order.

B. Preliminary observations on switches

The PCIe congestion can be hidden if the overlapping communications have very small message sizes. Timing disparities may cause the first message to arrive before the second congests the shared port. From our experiments, we did not observe congestion for small messages. Therefore, for the model simplification, we can safely assume that congestion of small messages is negligible and we focus only on the bandwidth distribution. To analyze bandwidth distribution behavior and its sensitivity to the transfer size, we evaluate the congestion ratio of two communications depending on their communication path and message size.

In a first conflict, we measure the latency of a single communication 0→3. Then, we add an overlapping communication, 1→2. We analyze the congestion impact on the communication latency depending on the message size. The results are presented in Fig. 3. It can be observed that with increasing message size the performance approaches a half

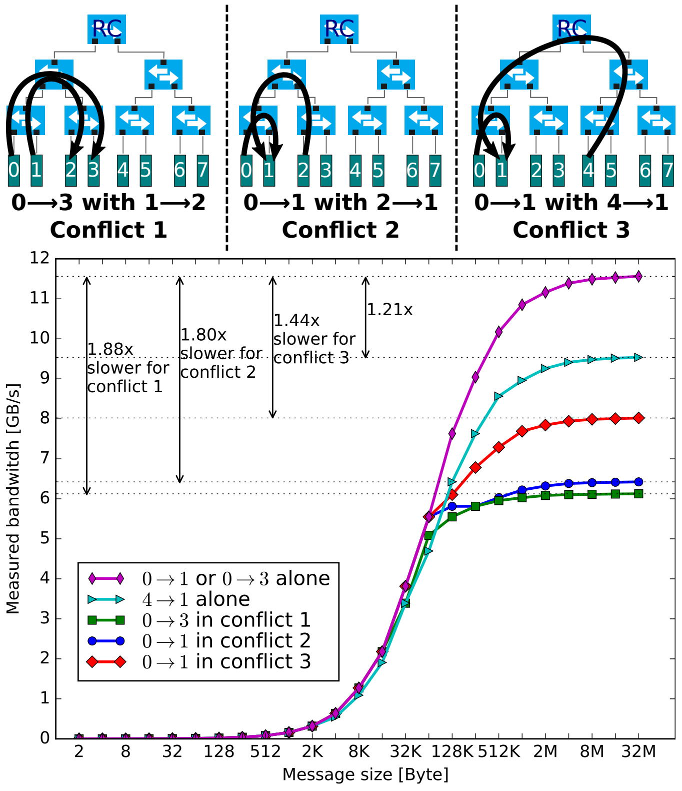


Fig. 3. Observed bandwidth for different conflicts of concurrent communications over topology T2. Presented data are median over a set of 100 tests, standard deviation is below 5% in all cases.

of its initial value for both 0→3 and 1→2 - the startup time disparities become negligible as most of the time the ports are shared equally between the two communications.

In a second conflict, we compare the performance of communication 0→1 sharing the PCIe bandwidth with communication 2→1. For this conflict, we observed a similar behavior as for the previous conflict. Because communication path of 2→1 is longer in term of PCIe ports traversed, the disparities in arrival time at the shared port is increased leading to slower convergence towards a 50% slowdown.

The third conflict introduces a unique property of the root complex. We observed that communication 0→1 is 1.44x (almost 1.5x) slower when sharing bandwidth with communication 4→1 whereas, in the previous conflict, its performance is reduced to a half. A second property of the root complex is displayed when comparing bandwidth measured just for single communications. Comparing communication 4→1 that crosses the root complex with 0→1 we observed a performance decrease of about 1.25x for communication 4→1. One possible explanation for these unique properties is that a communications crossing the root complex suffers from the overhead of using smaller packet sizes than other PCIe components. As discussed in the white paper [15], this effect reduces the theoretical transfer bandwidth efficiency to 76%.

In our test case application, COSMO climate code, the halo exchange message sizes depend on the grid size. The grid

1024×512×80 represents the Alpine region with Switzerland on its center. A 256×256×80 grid block per GPU leads to message sizes in the range of 40 KB to 254 KB. Such range of messages are large enough to create congestion.

C. Head-Of-Line blocking

HOL blocking applies if one communication crosses at least two ports on different switches. One port is responsible for the credit starvation affecting the port on the other switch. All communications that cross a port under credit starvation have their bandwidth availability reduced to the minimum bandwidth that the port can deliver. Therefore, the port reduces all communication bandwidths to the slowest value. When a communication subjected to HOL blocking is part of a conflict on another port, all other communications that are not subjected to HOL blocking, but part of this conflict, benefit proportionally from the available bandwidth originating from decreased bandwidth values.

IV. PCIe performance model

We establish a rigorous performance model that captures all the observations discussed in the previous section. We present a model derived from both the architecture specifications and the results from our experiments.

A. Dynamic congestion graph

We define a *congestion graph* as a directed graph $CG = (V, E)$, where the set of vertices V corresponds to devices and the edges E represent pending communications among them. A communication $c \in E$ in a congestion graph $CG = (V, E)$ is defined as a tuple $(s_c, d_c, tstart_c, tend_c, M_c)$ where $s_c, d_c \in V$ are respectively the source and destination nodes, $tstart_c \in Tstart$ and $tend_c \in Tend$ are the communication start and end times and M_c is the size in bytes of the message.

A *dynamic congestion graph*, is a CG where the set of edges changes in time. We sort all time events $T_i \in Tstart \cup Tend$ in the ascending order, so that $\forall i, j \in S: \; i < j \iff T_i < T_j$, where S is a set of time steps. A time step i is a period between the two consecutive events T_i and T_{i+1}. We can express the duration of time step i by the following equation:

$$t_i = T_{i+1} - T_i \quad \text{for} \quad i \geq 0 \tag{1}$$

A communication may take several time steps to transfer M_c bytes in time $L_c = tend_c - tstart_c$. In each time step different communications may create conflicts on different ports, resulting in sharing network bandwidth. The *congestion factor* $\rho_{c,i} \in [0, 1]$ expresses how much can a communication c utilize the available bandwidth in time step i due to the bandwidth sharing. In each time step, $m_{c,i} = t_i \cdot \rho_{c,i} \cdot B$ bytes are transferred, depending on the time step duration t_i, network bandwidth B and the current congestion factor $\rho_{c,i}$. We can

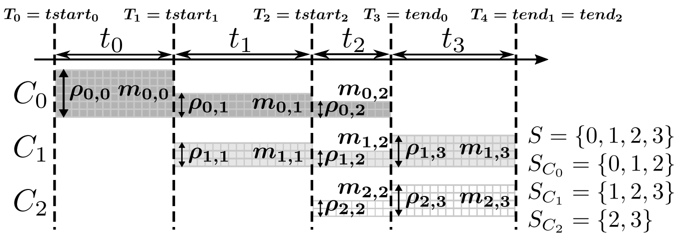


Fig. 4. A time step is defined as the interval of time between two events which are either a communication start or end. A communication belonging to a time step sends a certain amount of data which depends on the ratio of bandwidth it obtains due to arbitration policies. On the right, the set of steps S is shown, together with the steps belonging to the specific communications with $S_{C_0}, S_{C_1}, S_{C_2} \subset S$. For instance, at time step t_1, communication C_0 sends $m_{0,1}$ bytes with a bandwidth ratio of $\rho_{0,1}$ while communication C_1 sends $m_{1,1}$ bytes with a bandwidth ratio of $\rho_{1,1}$.

then express the total elapsed time L_c of a communication c being part of $S_c \subset S$ time steps as:

$$L_c = \sum_{i \in S_c} t_i = \frac{1}{B} \sum_{i \in S_c} \frac{m_{c,i}}{\rho_{c,i}} \tag{2}$$

$$M_c = \sum_{i \in S_c} m_{c,i} \tag{3}$$

Fig. 4 represents the evolution of communications being part of a several time steps in a congestion graph.

The goal of our model is to find communication end times which are not known at the beginning. As the input, only start times $tstart_c$ and message sizes M_c are known, $c \in E$. Our model first determines the congestion factors $\rho_{c,i}$, $i \in S$. Then, the length of each time step i can be computed and finally we find the unknown communication time L_c, from which we can derive $tend_c = L_c + tstart_c$.

The flow control of the PCIe fabric exchanges buffer credits between every two switch ports along a communication path. Therefore, the status of every port should be modeled. The proposed model for PCIe is based on a multicommodity flow problem using trees as graphs [16]. The objective of our model is not to assign flows to arcs for minimizing a cost, but to compute flow distribution of fixed-path flows as functions of the port or commodity capacities.

To find the required congestion factors ρ we use an algorithm that performs four steps.

(A) We model source arbitration for multiple communications issued by one device (Subsection V-A).

(B-C) We find for each port of every switch the output congestion factors ρ' based on the incoming congestion factors ρ. This step applies the port arbitration to reduce the bandwidth availability of a communication in case of congestion, $\rho' \leq \rho$. Step (B) computes the congestion factor for upstream port conflicts (Subsection V-B), whereas Step (C) computes it for downstream port conflicts (Subsection V-C). During Step (C) we consider the properties of communication crossing the root complex.

(D) Once all port arbitrations are computed for all communications, we evaluate the effects of HOL blocking.

HOL blocking, if applied, reduces output factor ρ' to a new congestion factor ρ''. Reducing already computed ρ' may change the output factors of Step (B) or Step (C) and, therefore, increase these factors to a new value ρ'' of communications not directly subjected to HOL blocking (Subsection V-D).

V. Computing congestion factors

Congestion factors reflect the bandwidth distribution among the communications. Port arbitration and the PCIe flow control are responsible for assigning PCIe component bandwidth. PCIe technical specification [17] allows vendor to select different port arbitration strategies. Therefore, switch technologies are often proprietary and in order to be able to analyze packet flows either proprietary software or specialized hardware [15] attached to the topology is required.

In this section we present the different features that reduce the bandwidth availability along with corresponding models. Finally, we introduce a performance model as an algorithm which combines all the features and computes the congestion factors of any congestion graph by following communication paths in a PCIe tree topology. In our model, we use the following notation (unless otherwise stated):

- $\overline{A}$, resp. $\underline{A}$, to denote that the property A applies to a port in the upstream, resp. downstream, direction;
- A_c to denote that the property A applies to the communication c;
- $A(p)$ to denote that the property A applies to the port p;
- A' to denote that the property A applies after a port arbitration policy;
- A'' to denote that the property A is modified by HOL blocking.

Each communication c enters a switch at port e and leaves from port l. If a communication goes towards the root complex then it is noted as $c(\underline{e}, \overline{l})$. If it goes in the opposite direction, i.e., towards a leaf, then it is noted as $c(\overline{e}, \underline{l})$.

Example: $\rho''_c(\underline{l})$ denotes the congestion factor ρ for the communication c on the port l in the downstream direction that is modified by HOL blocking.

A. Source arbitration

A device initiating multiple communications simultaneously applies an arbitration policy to determine the schedule in which communications access its bandwidth. Our tests show that a GPU device can only send one active communication at a time even when using asynchronous communication. The order in which the communications are scheduled is determined by the First Come First Serve strategy. Pending communications are modeled with a congestion factor $\rho = 0$.

B. Port arbitration for upstream port conflicts

Communications create conflicts by accessing ports of PCIe switches or the root complex (RC) simultaneously. RC is treated as a special switch. For upstream port conflicts, we use an equal sharing of the port bandwidth among the communications. We model a switch as a set of P ports p_n with $1 \leq n \leq P$. Each communication $c(\underline{e}, \overline{l})$ enters the switch at port $\underline{e}$ with a congestion factor $\rho_c(\underline{e})$ and leaves the switch from port $\overline{l}$ with a congestion factor $\rho_c(\overline{l})$ with $0 < \rho_c(\overline{l}) \leq \rho_c(\underline{e}) \leq 1$. The sum of congestion factors per port cannot exceed 1 (which represents maximum achievable bandwidth):

$$\sum_c \rho_c(\overline{l}) \leq 1 \tag{4}$$

We group together communications entering by a port $\underline{e}$ and leaving by a port $\overline{l}$ into a set of communications $C(\underline{e}, \overline{l})$ called a super communication. A super communication $C(\underline{e}, \overline{l})$ enters a port $\underline{e}$ with a congestion factor $\underline{R}_{C(\underline{e},\overline{l})} = \sum_{c \in C(\underline{e},\overline{l})} \rho_c(\underline{e})$ and leaves a port $\overline{l}$ with a congestion factor $\overline{R}_{C(\underline{e},\overline{l})} = \sum_{c \in C(\underline{e},\overline{l})} \rho_c(\overline{l})$. For each port $\overline{l}$, the arbitration mechanism determines the outgoing congestion factors $\overline{R}_{C(\underline{e},\overline{l})}$ depending on the incoming congestion factors $\underline{R}_{C(\underline{e},\overline{l})}$.

For a port $\overline{l}$ we define the set $C(\overline{l})$ as the set of all super communications $C(\underline{e}, \overline{l})$ leaving through that port. Arbitration applies only if super communication incoming congestion factors would overflow a port capacity: $\sum_{C(\underline{e},\overline{l}) \in C(\overline{l})} \underline{R}_{C(\underline{e},\overline{l})} > 1$, which would violate (4).

To model the effect of the arbitration, we use a linear transformation $\overline{R'}_{C(\underline{e},\overline{l})} = \beta \cdot \underline{R}_{C(\underline{e},\overline{l})}$ with $0 < \beta \leq 1$ as an arbitration parameter between the incoming and outgoing congestion factors of a super communication $C(\underline{e}, \overline{l})$. The model is represented by the following equations:

$$\sum_{C(\underline{e},\overline{l}) \in C(\overline{l})} \overline{R'}_{C(\underline{e},\overline{l})} = 1$$

$$\forall_{C(\underline{e},\overline{l}) \in C(\overline{l})} : \overline{R'}_{C(\underline{e},\overline{l})} = \beta \cdot \underline{R}_{C(\underline{e},\overline{l})}$$

with $0 < \beta \leq 1$. By solving this system we obtain the equation for a congestion factor $\overline{R'}$:

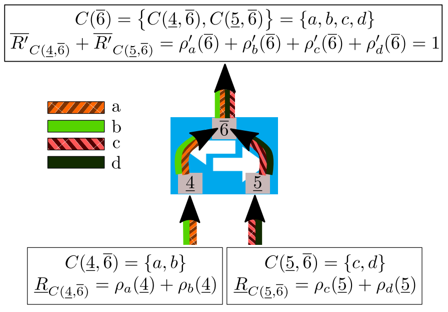
$$\overline{R'}_{C(\underline{e},\overline{l})} = \frac{\underline{R}_{C(\underline{e},\overline{l})}}{\sum\limits_{C(\underline{e},\overline{l}) \in C(\overline{l})} \underline{R}_{C(\underline{e},\overline{l})}} \tag{5}$$

To retrieve the congestion factor $\rho'_c(\overline{l})$ of a communication $c \in C(\underline{e}, \overline{l})$ we normalize the value of all $\rho_c(\underline{e})$ with the ratio of $\overline{R'}_{C(\underline{e},\overline{l})}$ and $\underline{R}_{C(\underline{e},\overline{l})}$:

$$\rho'_c(\overline{l}) = \rho_c(\underline{e}) \cdot \frac{\overline{R'}_{C(\underline{e},\overline{l})}}{\underline{R}_{C(\underline{e},\overline{l})}} = \rho_c(\underline{e}) \cdot \frac{1}{\sum\limits_{C(\underline{e},\overline{l}) \in C(\overline{l})} \underline{R}_{C(\underline{e},\overline{l})}} \tag{6}$$

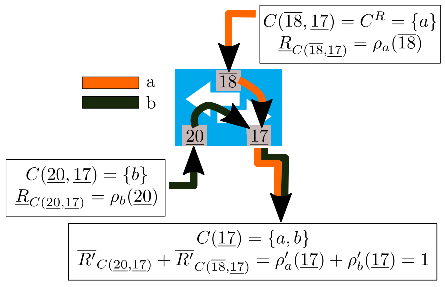
For every port that a communication c crosses in the upstream direction we use (6) to compute its congestion factor ρ'_c using its previously computed ρ_c at an earlier port in its path.

Example 1: Consider a scenario presented in Fig. 5a. Assume that $\rho_a(\underline{4}) = 0.6$, $\rho_b(\underline{4}) = 0.4$, $\rho_c(\underline{5}) = 0.3$ and $\rho_d(\underline{5}) = 0.5$. Note that $\rho_a(\underline{4}) + \rho_b(\underline{4}) = \underline{R}_{C(\underline{4},\overline{6})} = 1$ and $\rho_c(\underline{5}) + \rho_d(\underline{5}) = \underline{R}_{C(\underline{5},\overline{6})} = 0.8$. Then, we can see that $\underline{R}_{C(\underline{4},\overline{6})} + \underline{R}_{C(\underline{5},\overline{6})} > 1$. Therefore, an arbitration policy must



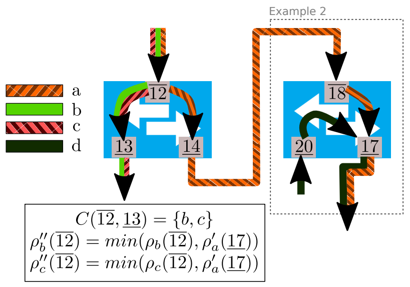
$C(\overline{6}) = \{C(\underline{4},\overline{6}), C(\underline{5},\overline{6})\} = \{a,b,c,d\}$
$\overline{R'}_{C(\underline{4},\overline{6})} + \overline{R'}_{C(\underline{5},\overline{6})} = \rho'_a(\overline{6}) + \rho'_b(\overline{6}) + \rho'_c(\overline{6}) + \rho'_d(\overline{6}) = 1$
a
b
c
d
$\overline{6}$
$\underline{4}$
$\underline{5}$
$C(\underline{4},\overline{6}) = \{a,b\}$
$\underline{R}_{C(\underline{4},\overline{6})} = \rho_a(\underline{4}) + \rho_b(\underline{4})$
$C(\underline{5},\overline{6}) = \{c,d\}$
$\underline{R}_{C(\underline{5},\overline{6})} = \rho_c(\underline{5}) + \rho_d(\underline{5})$

(a) Example 1: upstream port conflict.



$C(\overline{18},\underline{17}) = C^R = \{a\}$
$\underline{R}_{C(\overline{18},\underline{17})} = \rho_a(\overline{18})$
a
b
$\overline{18}$
$\underline{20}$
$\underline{17}$
$C(\underline{20},\underline{17}) = \{b\}$
$\underline{R}_{C(\underline{20},\underline{17})} = \rho_b(\underline{20})$
$C(\underline{17}) = \{a,b\}$
$\overline{R'}_{C(\underline{20},\underline{17})} + \overline{R'}_{C(\overline{18},\underline{17})} = \rho'_a(\underline{17}) + \rho'_b(\underline{17}) = 1$

(b) Example 2: downstream port conflict.



Example 2
a
b
c
d
$\overline{12}$
$\underline{13}$
$\underline{14}$
$\overline{18}$
$\underline{20}$
$\underline{17}$
$C(\overline{12},\underline{13}) = \{b,c\}$
$\rho''_b(\overline{12}) = min(\rho_b(\overline{12}), \rho'_a(\underline{17}))$
$\rho''_c(\overline{12}) = min(\rho_c(\overline{12}), \rho'_a(\underline{17}))$

(c) Example 3: HOL blocking conflict.

Fig. 5. Examples of each conflict type.

reduce the congestion factors $\rho'_i(\overline{6})$ with $i \in \{a,b,c,d\}$. From (5), we derive

$$\overline{R}_{C(\underline{4},\overline{6})} = \frac{\underline{R}_{C(\underline{4},\overline{6})}}{\underline{R}_{C(\underline{4},\overline{6})} + \underline{R}_{C(\underline{5},\overline{6})}} = \frac{1}{1.8} = \frac{5}{9}$$

Similarly, we obtain $\overline{R}_{C(\underline{5},\overline{6})} = \frac{4}{9}$. Finally, from (6) we obtain $\rho'_a(\overline{6}) = 0.33$, $\rho'_b(\overline{6}) = 0.22$, $\rho'_c(\overline{6}) = 0.17$ and $\rho'_d(\overline{6}) = 0.28$.

C. Port arbitration for downstream port conflicts

To allocate the bandwidth, a downstream port arbitration applies a weighted round-robin method among super communications, which is modeled by considering a fair split of the port bandwidth. If $n > 0$ super communications are part of a downstream port conflict at port $\underline{l}$, then for each $C(\overline{e},\underline{l})$ we limit its congestion factor $\underline{R}'_{C(\overline{e},\underline{l})} \leq {}^1/_n$.

Moreover, we distinguish the case where a super communication includes a communication that had crossed the root complex. As shown in Section III-B we observed a different behavior for communications crossing the root complex. Such communications suffer from a bandwidth loss, which results in a lower impact on other communications. We denote the set of such super communications that cross the root complex as C^R and by n the total number of super communications passing through a port $\underline{l}$. To model the port arbitration on a downstream port conflict we use the following formula:

- if $C^R = \emptyset$, then

$$\underline{R}'_{C(\overline{e},\underline{l})} = \frac{1}{n}$$

- if $C^R \neq \emptyset$ then

$$\underline{R}'_{C(\overline{e},\underline{l})} = \begin{cases} \min(\max(\frac{1}{n} - \tau, 0), \underline{R}_{C(\overline{e},\underline{l})}) & \text{if } C^R \cap C(\overline{e},\underline{l}) \neq \emptyset \\ \min(\frac{1}{n} + \tau, \underline{R}_{C(\overline{e},\underline{l})}) & \text{otherwise} \end{cases}$$

Here, τ is a factor that represents the performance loss of communication crossing a root complex. Finally, we compute the resulting congestion factor as:

$$\rho'_c(\underline{l}) = \rho_c(\underline{l}) \cdot \frac{\underline{R}'_{C(\overline{e},\underline{l})}}{\underline{R}_{C(\overline{e},\underline{l})}} \tag{7}$$

A single communication c crossing the root complex has its bandwidth reduced by (7) and obtains $\rho'_c = 1 - \tau$.

Example 2: Consider the example of Fig. 5b. Communications $a(\overline{18},\underline{17})$ and $b(\underline{20},\underline{17})$ generate a downstream port conflict on port $\underline{17}$. Assume that $\underline{R}_{C(\overline{18},\underline{17})} = \rho_a(\overline{18},\underline{17}) = 0.7$ and $\underline{R}_{C(\underline{20},\underline{17})} = \rho_b(\underline{20},\underline{17}) = 0.9$. Furthermore, communication a crosses the root complex, therefore $C^R \neq \emptyset$. Assume that $\tau = 0.2$. Therefore, $\min(\max(\frac{1}{2} - \tau, 0), \underline{R}_{C(\overline{18},\underline{17})}) = 0.3$ and $\min(\frac{1}{2} + \tau, \underline{R}_{C(\underline{20},\underline{17})}) = 0.7$. Using (7), we obtain $\rho'_a(\underline{17}) = 0.3$ and $\rho'_b(\underline{17}) = 0.7$.

D. Head-Of-Line blocking

HOL blocking is computed at the last step of the algorithm where all communications have obtained congestion factors ρ'. We first identify communications subjected to HOL blocking. Then, we increase the congestion factors of communications which are in conflict with them.

Consider a communication c and assume that its congestion factor on a port l is $\rho'_c(l)$ with $c \in C(e,l)$. Now assume that a communication $b \in C(e,l)$ crosses another switch after port l and is part of an upstream or downstream port conflict at port m for which it obtains $\rho'_b(m)$. We define a set of all ports that the communication b crosses in its path to the destination after crossing the port l as $P_b(l)$. We compute $\rho''_c(l)$ as follows:

$$\rho''_c(l) = \min_{b \in C(e,l)} (\min_{m \in P_b(l)} \rho'_b(m)) \tag{8}$$

Once all communications subjected to HOL blocking are identified, we compute the congestion factor increase of communications having a conflict with communications subjected to HOL blocking. Consider the port l and the set $C(l)$ of all communications that pass through it. Among those communications, some of them are subjected to HOL blocking. We denote this set $C_{HOL}(l) \subset C(l)$. Then, all $c \in C(l) \setminus C_{HOL}(l)$ have their congestion factor raised due to more bandwidth available at the port l. To find the new congestion factors ρ''_c for all $c \in C(l) \setminus C_{HOL}(l)$, we first find $R'(l) = \sum_{b \in C_{HOL}(l)} \rho'_b(l)$ and $R''(l) = \sum_{b \in C_{HOL}(l)} \rho''_b(l)$

with $R''(l) \leq R'(l)$. Then, it allows us to find ρ''_c as follows for $c \in C(l) \setminus C_{HOL}(l)$:

$$\rho''_c(l) = \rho'_c(l) + \frac{R'(l) - R''(l)}{|C(l) \setminus C_{HOL}(l)|} \tag{9}$$

Example 3: Consider the example in Fig. 5c. Communications a, b and c enter the switch by port $\overline{12}$ and d enters by port $\underline{20}$. The communication a is later a part of a downstream port conflict, where its congestion factor ρ'_a was lowered. Then, according to (8), communications b and c are subjected to the HOL blocking with their new congestion factors.

$$\rho''_i(\overline{12}) = \min(\rho_i(\overline{12}), \rho'_a(\underline{17})) \quad \text{for} \quad i \in \{b, c\}$$

E. Algorithm to compute congestion factors

To combine all presented features, we present an algorithm that is capable of computing the congestion factors. The different steps of the algorithm are presented in Algorithm 1.

Algorithm 1 ComputingCongestionFactors

```
Require: Initial graph CG
Ensure: Congestion factors of each edge
  for source $s$ in all leaf nodes do                              ▷ Step (A)
      Consider only the first communication of source $s$ in the next
  steps of the algorithm and apply a value of zero to congestion
  factor of the other communications
  end for
  for $d = maxDepth$, $d \geq 0$, $d = d - 1$ do                   ▷ Step (B)
      for x in all switches at depth $d$ do
          for p in all ports of switch x do
              Arbitration: upstream port conflicts $p$, Equation (6)
          end for
      end for
  end for
  for $d = 0$, $d \leq maxDepth$, $d = d + 1$ do                   ▷ Step (C)
      for x in all switches and root complexes at depth $d$ do
          for p in all ports of switch x do
              Arbitration: downstream port conflicts $p$, Equation (7)
          end for
      end for
  end for
  for x in all switches do                                         ▷ Step (D)
      for p in all ports of switch x do
          Apply HOL blocking on $p$, Equation (8)
      end for
  end for
  for x in all switches do
      for p in all ports of switch x do
          Apply HOL blocking on $p$, Equation (9)
      end for
  end for
```

All congestion factors are initialized to the value 1. The first step, Step (A), is to ensure that all sources send only one communication at a time by setting a congestion factor of zero to communications in a waiting state. Communications in a waiting state are not considered within the remaining steps of the algorithm. In Step (B), we apply the port arbitration model for every upstream port conflict of every switch starting by switches with higher depth (closer to the leaves). $maxDepth$ is the maximum depth of the tree. This step represents the ascending phase of a communication in the tree topology. Once the congestion factors are computed for the ascending phase, the port arbitration model is applied for all downstream port conflicts in Step (C). This phase reflects the descending phase of the communications towards their destination devices. The model is applied starting from the root complex towards the lower switches. Finally, the last step, Step (D), computes the effect of HOL blocking on communications and its consequent impact on all other communications. Algorithm 1 applies to any tree topology.

We have developed python objects to encapsulate the different PCIe components. A Switch object holds any number of Port objects. A root complex is also encapsulated in a Switch object. A Port object holds a set of Communication object references and an Arbiter object that contains the port arbitration and HOL blocking formulas. Communication objects represent the communications: source, destination, starting time, size and congestion factors per port. We represent a congestion graph step by creating a Communication object for every edge in the graph. We associate for every Port object the set of Communication object references that traverses it. By connecting these python objects together, we can create any tree topology and apply the model to any set of communications. The evaluation of one graph by the model requires only few milliseconds on one core.

VI. Applying the model on a full example

To clarify the usage of the model, we present an example on topology T2. The first part of the example applies Algorithm 1 to compute the congestion factors of a specific congestion graph step. The second part of the example computes the evolution of communication elapsed times within a congestion graph. The congestion graph used for the example is composed of 4 communications: $0 \to 2$, $1 \to 4$, $3 \to 2$, $6 \to 4$. Communications start at the same time and send each 300 MB of data.

Computation of congestion factors are presented in Table I. Algorithm 1 initially sets all congestion factors to the value 1. Every communication starts from a different source which implies that congestion factors remain unchanged after Step (A). In Step (B), communications (a) and (b) cross the same upstream port and both have access to only half of the available bandwidth by (6). Downstream port arbitration is computed in Step (C) for two sets of communications: (a) and (c), and (b) and (d). Neither of communication (a) nor (c) crosses the root complex, then by (7) both communication obtains a congestion factor of $1/2$. For the second set of communications, (b) and (d), communication (b) crosses the root complex. By selecting $\tau = 1/5$ and using (7), the algorithm computes $\rho'_b = 3/10$ and $\rho'_d = 7/10$. In the final step, Step (D), HOL blocking is computed. Communications (a) and (b) share a same upstream port with communication (b) obtaining a lower congestion factor at a later port which reduces the congestion factor of (a) due to HOL blocking. Communication (a) becomes subjected to HOL blocking and its congestion factor equals to the congestion factor of (b) from (8). With (a) being subjected to HOL blocking, the conflict between (a) and (c) is updated

TABLE I

FIRST PART OF THE EXAMPLE: EVOLUTION OF CONGESTION FACTORS OF COMMUNICATIONS CROSSING SEVERAL PORTS WITH $\tau = {}^{1}/_{5}$.

congestion graph step: (0→2, 1→4, 3→2, 6→4) on topology T2

comm.	Step (A)	Step (B)	Step (C)	Step (D)	
(a) 0→2	1	$^1/_2$	$^1/_2$	$^3/_{10}$	$^3/_{10}$
(b) 1→4	1	$^1/_2$	$^3/_{10}$	$^3/_{10}$	$^3/_{10}$
(c) 3→2	1	1	$^1/_2$	$^1/_2$	$^7/_{10}$
(d) 6→4	1	1	$^7/_{10}$	$^7/_{10}$	$^7/_{10}$

TABLE II

SECOND PART OF THE EXAMPLE: EVOLUTION OF COMMUNICATION ELAPSED TIMES FOR A CONGESTION GRAPH.

Congestion graph step 1			
comm.	cong. factor	data remaining	elapsed time
(a) 0→2	$^3/_{10}$	128 MB	36 ms
(b) 1→4	$^3/_{10}$	128 MB	36 ms
(c) 3→2	$^7/_{10}$	0 MB	36 ms
(d) 6→4	$^7/_{10}$	0 MB	36 ms
Congestion graph step 2			
comm.	cong. factor	data remaining	elapsed time
(a) 0→2	$^1/_2$	0 MB	65 ms
(b) 1→4	$^1/_2$	0 MB	65 ms

by (9) and the communication (c) obtains a higher congestion factor of $^7/_{10}$.

Evolution of communication elapsed times among the steps is presented in Table II. In the first congestion graph step communications (c) and (d) are the first communications to complete with an elapsed time of 36 ms. During this time step the two remaining communications, (a) and (b), are sending a lower quantity of data due to their lower congestion factors. When communications (c) and (d) complete, the congestion graph moves to a new step. In this new step, the communication (b) is not anymore in conflict with (d) and its congestion factor is increased. Consequently, the negative effect of HOL blocking does not affect communication (a) anymore. Both communications obtain a congestion factor of $^1/_2$. With these new congestion factors, both communications (a) and (b) complete simultaneously sending their remaining data of 172 MB in 29 ms, which leads to an elapsed time of 65 ms for both communications.

VII. MODEL VALIDATION

To validate the model we have written a simple benchmark. This benchmark creates a dynamic congestion graph by triggering communications among devices at the same time. A communication is an asynchronous memory transfer from the memory of one device to the memory of another device by using *cudaMemcpyPeerAsync()* function from the CUDA [14] libraries. To assess the model validation we use communications with a large size to be able to minimize

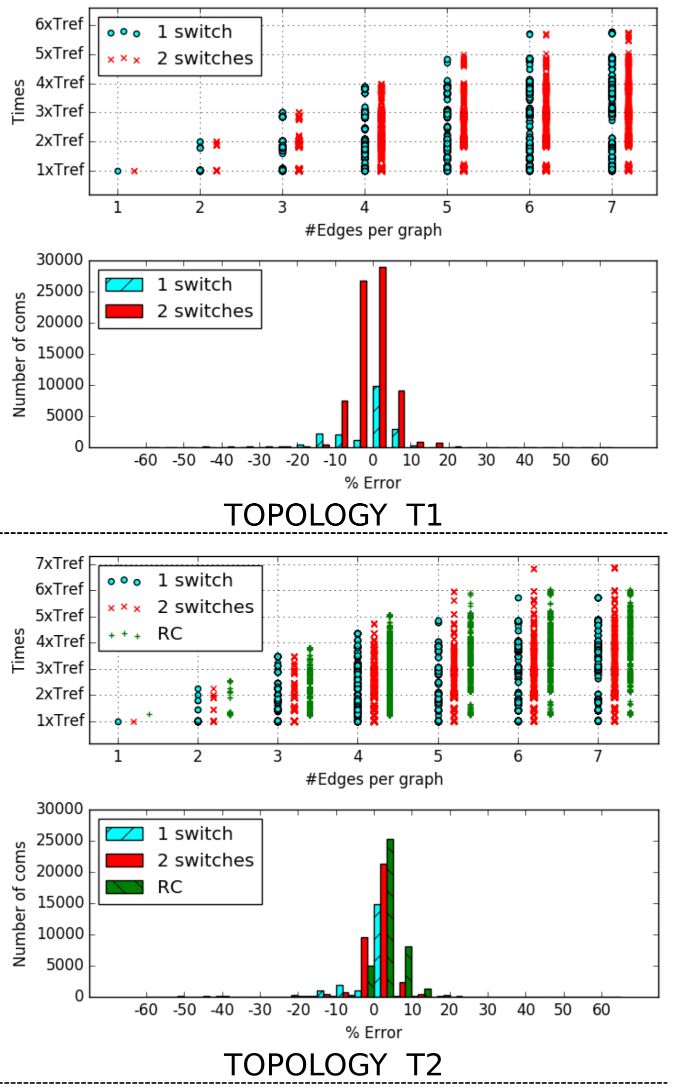


Fig. 6. Validation of the model on several congestion graphs for both topologies T1 (top plots) and T2 (bottom plots).

the disparities of startup time and increase the time of ports being shared among them. We have selected a message size of 300 MB, however, any other large sizes can be used.

Our model requires to identify the value of two parameters, B and τ. The parameter B is the measured bandwidth between two GPU processors of one K80, B = 11.6 GB/s. The parameter τ reflects the bandwidth difference between a single communication crossing the root complex and a single communication which does not cross it. This bandwidth difference is displayed in Fig. 3. Therefore, we evaluate $\tau = 1 - \frac{1}{1.21} = 0.17355$.

Our validation analysis uses a set of dynamic congestion graphs that consists of both arbitrarily selected and random graphs. We select specific graphs that allow us to identify special behaviors of the PCIe network. To this set of graphs we add simple collective operation graphs like scatter, gather, or all-to-all. All graphs in this set are unique and not isomorphic to any other graphs when projected on topology T2. Our validation set also includes all possible and non isomorphic graphs projected on two topologies similar to topology T2 but with only four devices: one topology with four consecutive GPUs of topology T2, and one with all even numbered GPUs of topology T2. To this set we add randomly generated graphs. Our set of graphs is composed of $22,298$ unique directed graphs leading to $94,259$ communications for topology T1 and $22,635$ unique directed graphs, $95,906$ communications, for topology T2. Fig. 6 displays two plots per topology: the measured time of each communication (top) and the number of communications obtaining a specific relative error (bottom). The relative error is computed by comparing the predicted time against the measured time. Measured communication times are shown as multiples of the reference time T_{ref} of one congestion-free communication crossing only one switch (internal to a K80), with $T_{ref} = 25.2829$ ms for a message size of 300 MB. Communication times are arranged by groups of communications which belongs to graphs with the same number of edges with a maximum of seven edges. Communications are also arranged by the type and number of PCIe components that they cross.

The error distribution plots display the number of communications for which times are predicted with a specific error. For topology T1 the error range is -46% to 31% where as for topology T2 the error range is -56% to 29%. For both topologies more than 97% of the predicted communications are in the range -15% to 15%, which represent more than $91,000$ communications predicted with a very low error. The maximum absolute error for both topologies does not exceed 56%. These results validate the accuracy of our model. A tiny group of communications per topology, 408 communications for T2 and 348 communications for T1, shows an error below -30%. These errors are due to features that are incompletely modeled, such as variability on the port capacity due to indirect loads on other ports of the switch.

Measured times are largely spread implying that all communications are sensitive to congestion. Moreover, in almost all cases, measured times cover the full range of values between the fastest and the slowest time. This variability of time values indicates that making accurate predictions is hard.

VIII. Applications of the Model

We apply the model for halo exchange communication patterns among multiple GPUs. We identify the fastest congestion graph for a halo exchange pattern of a 2D domain decomposition as used in weather forecasting application and we generalize it to a halo exchange of a 3D domain decomposition. For both cases, we use domains with non-periodic boundary conditions and we consider only faces exchange among the GPUs. In practice, the GPUs do not always start their initial communications simultaneously, however, the worst case scenario in terms of congestion behavior will occur when they do. Here we study this worst-case scenario of halo exchanges on topology T2 which is the topology of a MeteoSwiss CS-Storm node.

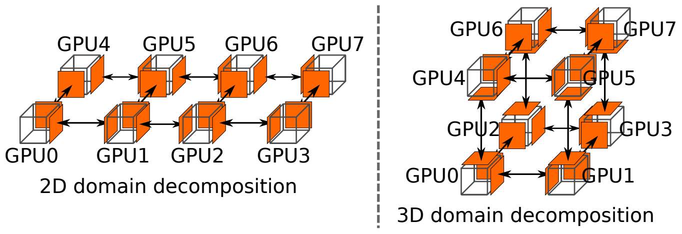


Fig. 7. Typical halo exchange used by 2D and 3D domain decomposition of a 3D input data. The input data are divided into eight sub-domains. Each GPU holds the data of one sub-domain and exchanges faces with its neighbours.

A. Halo exchange of a 2D domain decomposition

The COSMO [2] application is a non-hydrostatic local area atmospheric model used for both operational numerical weather prediction and long-term climate simulation. COSMO is developed by seven national weather forecast institutions. Parallelism strategy decomposes the three-dimensional domain into a two-dimensional Cartesian grid in the North-South and East-West directions. Each sub-domain is assigned to an MPI rank, with MPI communication used to perform halo exchanges among the sub-domains [18]. MeteoSwiss uses an implementation of COSMO GPUs for fine-grained parallelism on each sub-domain [19]. This implementation of COSMO is used on a cluster of nodes with two sockets, where each socket has four K80 GPUs being connected following topology T2. Each socket runs a standalone instance of COSMO.

A halo exchange for eight sub-domains is displayed in Fig. 7. Each GPU sends multiple communications creating a set of possible contention graphs. For this domain decomposition, four GPUs have a choice among three communications, the other four GPUs can choose between two communications. Therefore, we compute $(3!)^4 \cdot (2!)^4 = 20,736$ different congestion graphs and we apply the model on each of them. The fastest graph is displayed in Fig. 8. For clarity we split it into three steps. The model indicates that a set of specific communication rings limits congestion. These rings are carefully selected to minimize the number of communications crossing the root complex in the same direction. Congestion still occurs as communications are not synchronized among step transitions, but the impact is negligible. For instance, when GPU1 starts communication $1 \rightarrow 5$, communication $0 \rightarrow 4$ is not completed leading to a small overlap of time where both communications concurrently access several ports. Elapsed time of all congestion graphs are displayed in Fig. 9. The curve indicates (sharp slope at the beginning) that only few graphs are faster than the vast majority of graphs. The fastest graph is 1.9 times faster than the slowest one and 1.6 times faster than the currently implemented graph in COSMO. Comparing the values obtained by the model with the graph running on topology T2, the model overestimates the fastest graph time by 11%.

We trace the entire set of halo exchanges of a run of COSMO simulating one time step, i.e., 20 seconds of simulated weather forecast for a grid of $1024 \times 512 \times 80$ or

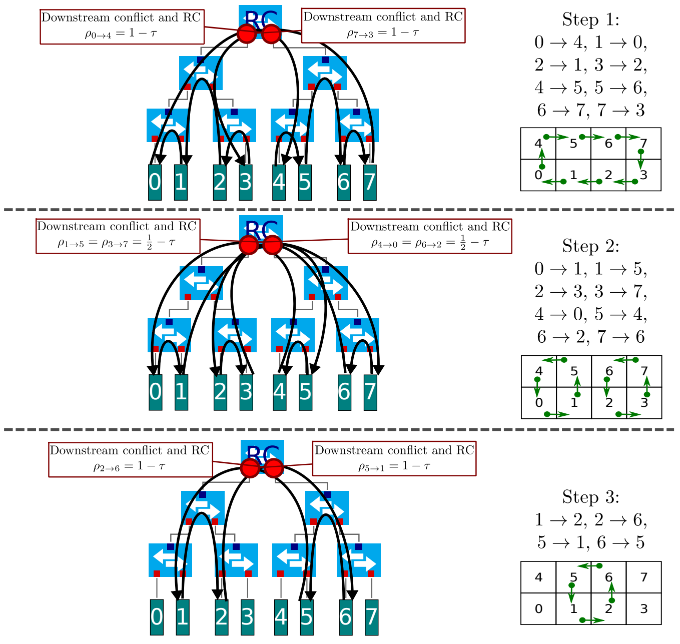

Downstream conflict and RC $\rho_{0\to4} = 1-\tau$
RC
Downstream conflict and RC $\rho_{7\to3} = 1-\tau$
Step 1:
$0\to4$, $1\to0$,
$2\to1$, $3\to2$,
$4\to5$, $5\to6$,
$6\to7$, $7\to3$
Downstream conflict and RC $\rho_{1\to5} = \rho_{3\to7} = \frac{1}{2}-\tau$
Downstream conflict and RC $\rho_{4\to0} = \rho_{6\to2} = \frac{1}{2}-\tau$
Step 2:
$0\to1$, $1\to5$,
$2\to3$, $3\to7$,
$4\to0$, $5\to4$,
$6\to2$, $7\to6$
Downstream conflict and RC $\rho_{2\to6} = 1-\tau$
Downstream conflict and RC $\rho_{5\to1} = 1-\tau$
Step 3:
$1\to2$, $2\to6$,
$5\to1$, $6\to5$
0 1 2 3 4 5 6 7

Fig. 8. Fastest order of communications for a halo exchange from a 2D domain decomposition on topology T2. For clarity, we split the graph into three steps. At every step each GPU triggers a communication. A set of selected rings is the fastest pattern.

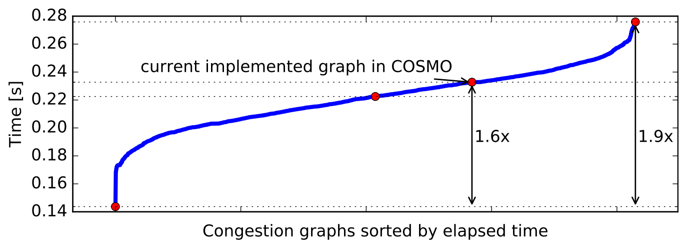


Fig. 9. Times of all possible congestion graphs of a 2D domain.

256×256×80 block per GPU. One time step triggers 312 halo exchanges for which message sizes are in the range of 40 KB to 254 KB. These halo exchanges account for 16% of the time needed to execute one time step. We present, in Fig. 10, the ratio of the measured and modeled times of the fastest congestion graph against the currently implemented congestion graph in COSMO. We repeat 50 times one time step and take the fastest measured time. Standard deviation of this set of 50 tests is below 5%. The large difference in the range of 1.75x to 5x between the modeled and measured times are attributable to MPI overhead in comparison to direct CUDA function calls. This overhead limits the benefit of using the fastest graph inside COSMO. Halo exchanges of small message sizes (below 100 KB), such as halo exchanges occurring at positions 30 to 90 and 180 to 250, do not contribute to the performance improvement. For larger halo exchange message sizes, such as halo exchange from position 0 to 30, 90 to 180 and 250 to 312, the fastest graph gives about 5–10% better performance. The fastest graph with MPI improves the sum of the halo exchange times of one time step

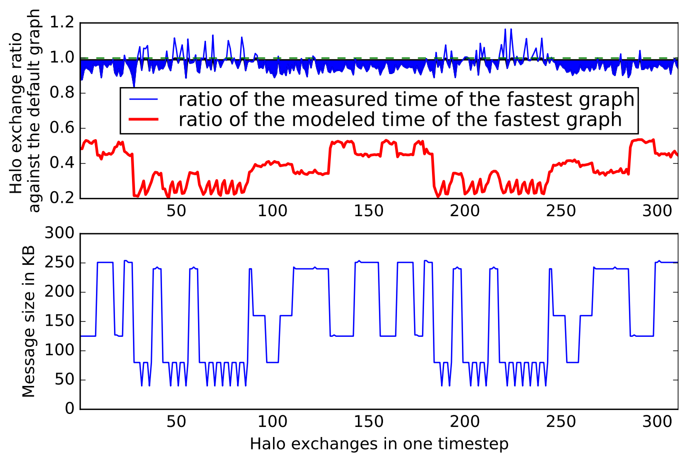


Fig. 10. Measured and modeled times of the fastest congestion graph compared with the time of the currently implemented congestion graph in COSMO. The X-axis is the list of 312 halo exchanges in their order of occurrence during one time step. The top plot shows the ratio of measured and modeled times and the bottom plot the message size used by each halo.

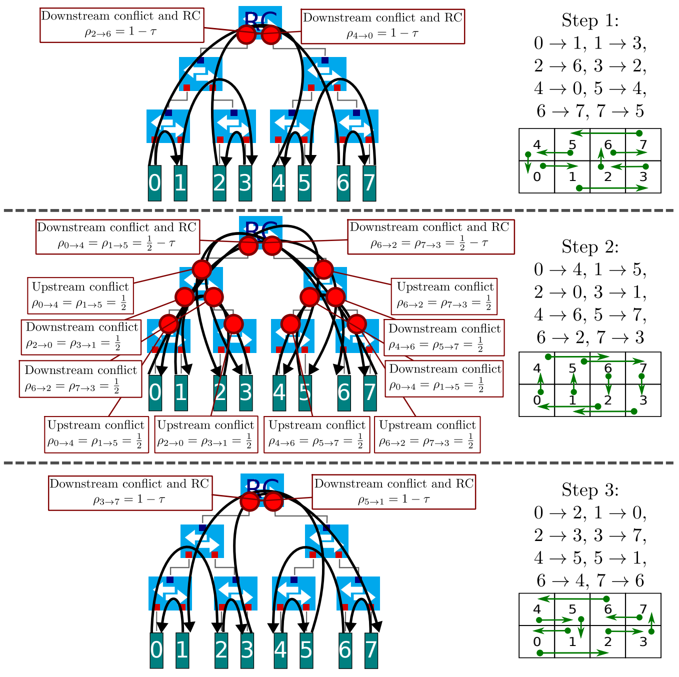

Downstream conflict and RC $\rho_{2\to6} = 1-\tau$
RC
Downstream conflict and RC $\rho_{4\to0} = 1-\tau$
Step 1:
$0\to1$, $1\to3$,
$2\to6$, $3\to2$,
$4\to0$, $5\to4$,
$6\to7$, $7\to5$
Downstream conflict and RC $\rho_{0\to4} = \rho_{1\to5} = \frac{1}{2}-\tau$
Downstream conflict and RC $\rho_{6\to2} = \rho_{7\to3} = \frac{1}{2}-\tau$
Upstream conflict $\rho_{0\to4} = \rho_{1\to5} = \frac{1}{2}$
Upstream conflict $\rho_{6\to2} = \rho_{7\to3} = \frac{1}{2}$
Downstream conflict $\rho_{2\to0} = \rho_{3\to1} = \frac{1}{2}$
Downstream conflict $\rho_{4\to6} = \rho_{5\to7} = \frac{1}{2}$
Downstream conflict $\rho_{6\to2} = \rho_{7\to3} = \frac{1}{2}$
Downstream conflict $\rho_{0\to4} = \rho_{1\to5} = \frac{1}{2}$
Upstream conflict $\rho_{0\to4} = \rho_{1\to5} = \frac{1}{2}$
Upstream conflict $\rho_{2\to0} = \rho_{3\to1} = \frac{1}{2}$
Upstream conflict $\rho_{4\to6} = \rho_{5\to7} = \frac{1}{2}$
Upstream conflict $\rho_{6\to2} = \rho_{7\to3} = \frac{1}{2}$
Step 2:
$0\to4$, $1\to5$,
$2\to0$, $3\to1$,
$4\to6$, $5\to7$,
$6\to2$, $7\to3$
Downstream conflict and RC $\rho_{3\to7} = 1-\tau$
Downstream conflict and RC $\rho_{5\to1} = 1-\tau$
Step 3:
$0\to2$, $1\to0$,
$2\to3$, $3\to7$,
$4\to5$, $5\to1$,
$6\to4$, $7\to6$
0 1 2 3 4 5 6 7

Fig. 11. Fastest order of communications for a halo exchange from a 3D domain decomposition on topology T2. At every step each GPU triggers a communication. Step 1 and Step 3 use a ring pattern which limits conflicts, but as a result, Step 2 has a large number of conflicts. Nevertheless, such graph is the fastest graph among the entire set of possible graphs.

by 5.7% (filled zone in the plot). This effect is expected as congestion is more visible for large message sizes (see Fig. 3 in Section III).

B. Halo exchange of a 3D domain decomposition

3D domain decomposition are used by scientific applications [20]. A halo exchange of 3D domain decomposition for eight sub-domains is displayed in Fig. 7. With eight active GPUs and three communications per GPU, we compute

$(3!)^8 = 1,679,616$ orders of communications, and, therefore, we evaluate the same amount of congestion graphs. The fastest graph is presented in Fig. 11. It consists of three steps where Step 1 and Step 3 use a ring pattern that limits conflicts to the RC (as seen for the 2D domain decomposition example). However, communications during Step 2 create conflicts at every port. Without the model, it is difficult to identify the most efficient set of communications for Step 2, for which conflicts occur, and for the steps Step 1 and Step 3 which create only RC conflicts.

In the entire set of graphs, the fastest graph is 2.57 times faster than the slowest graph, whereas the median graph is 1.44 times faster than the slowest graph. This large difference in performance shows the significant impact of congestion on PCIe topology. Comparing the predicted values with real values obtained by running this graph on topology T2, the model overestimates the time of the fastest graph by 8%.

IX. Conclusion

MeteoSwiss has chosen a computer architecture with densely populated accelerator nodes, which provide high number of teraflops for a relatively low power consumption. We presented a performance model for the PCIe fabric used to connect GPUs on such nodes. The model is proven to be accurate with almost 97% of communications time predicted with an error in the range of -15% to 15%. We apply the model on the application COSMO to identify graphs with higher efficiency for executing halo exchanges among GPUs.

Based on this model it will be possible to investigate the cost of collective operations and neighborhood communications [21] among GPUs to define algorithms that maximize performance. This study will prove to be useful for incorporating such algorithms in the communication layer of COSMO, or, more generally, into an MPI library.

References

[1] HPCwire. Today's outlook: GPU-accelerated weather forecasting. Accessed on 03/2016. [Online]. Available: http://www.hpcwire.com/2015/09/15/todays-outlook-gpu-accelerated-weather-forecasting

[2] Consortium for small-scale modeling. Accessed on 03/2016. [Online]. Available: http://www.cosmo-model.org

[3] T. Gysi, C. Osuna, O. Fuhrer, M. Bianco, and T. C. Schulthess, "STELLA: A domain-specific tool for structured grid methods in weather and climate models," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC'15. New York, NY, USA: ACM, 2015, pp. 41:1–41:12. [Online]. Available: http://doi.acm.org/10.1145/2807591.2807627

[4] O. Fuhrer, C. Osuna, X. Lapillonne, T. Gysi, B. Cumming, M. Bianco, A. Arteaga, and T. Schulthess, "Towards a performance portable, architecture agnostic implementation strategy for weather and climate models," *Supercomputing frontiers and innovations*, vol. 1, no. 1, 2014. [Online]. Available: http://superfri.org/superfri/article/view/17

[5] T. Lutz, C. Fensch, and M. Cole, "PARTANS: An autotuning framework for stencil computation on multi-GPU systems," *ACM Trans. Archit. Code Optim.*, vol. 9, no. 4, pp. 59:1–59:24, Jan. 2013. [Online]. Available: http://doi.acm.org/10.1145/2400682.2400718

[6] G. A. Elliott, B. C. Ward, and J. H. Anderson, "GPUSync: A framework for real-time GPU management," in *2013 IEEE 34th Real-Time Systems Symposium*. Institute of Electrical & Electronics Engineers (IEEE), Dec. 2013. [Online]. Available: http://dx.doi.org/10.1109/rtss.2013.12

[7] R. Solomon, *PCI Express Basics*. PCI-SIG, Oct. 2011.

[8] Marketwired. PLX achieves industry-first compliance of PCI Express 3.0 switches on exclusive PCI-SIG integrators list. Accessed on 03/2016. [Online]. Available: http://www.marketwired.com/press-release/plx-achieves-industry-first-compliance-pci-express-30-switches-on-exclusive-pci-sig-nasdaq-plxt-1875400.htm

[9] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, 1st ed. Addison-Wesley Professional, 2010.

[10] V. Krishnan and D. Mayhew, "A Localized Congestion Control Mechanism for PCI Express Advanced Switching Fabrics," in *In Proc. 12th IEEE Symp. on Hot Interconnects*, 2004.

[11] D. Schaa and D. Kaeli, "Exploring the multiple-GPU design space," in *2009 IEEE International Symposium on Parallel & Distributed Processing*. Institute of Electrical & Electronics Engineers (IEEE), May 2009. [Online]. Available: http://dx.doi.org/10.1109/ipdps.2009.5161068

[12] M. Martinasso and J.-F. Méhaut, "A contention-aware performance model for HPC-based networks: A case study of the InfiniBand network," in *Euro-Par 2011 Parallel Processing*. Springer Berlin Heidelberg, 2011, pp. 91–102. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23400-2_10

[13] I. Faraji, S. H. Mirsadeghi, and A. Afsahi, "Topology-aware GPU selection on multi-GPU nodes," in *Sixth International Workshop on Accelerators and Hybrid Exascale Systems (AsHES)*. To be published in Proceedings of the 30th IEEE International Parallel & Distributed Processing Symposium Workshops, May 2016.

[14] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA," *Queue*, vol. 6, no. 2, p. 40, Mar. 2008. [Online]. Available: http://dx.doi.org/10.1145/1365490.1365500

[15] "Understanding performance of PCI Express systems," White paper, Xilinx, 2008.

[16] A. A. Assad, "Multicommodity network flows - a survey." *Networks*, vol. 8, no. 1, pp. 37–91, 1978. [Online]. Available: http://dblp.uni-trier.de/db/journals/networks/networks8.html#Assad78

[17] PCI-SIG 2010, *PCI Express Base Specification 3.0*, Std.

[18] T. Gysi, C. Osuna, O. Fuhrer, M. Bianco, and T. C. Schulthess, "STELLA," in *Proceedings of the International Conference for High Performance Computing Networking, Storage and Analysis on - SC'15*. Association for Computing Machinery (ACM), 2015. [Online]. Available: http://dx.doi.org/10.1145/2807591.2807627

[19] O. Fuhrer, C. Osuna, X. Lapillonne, T. Gysi, B. Cumming, M. Bianco, A. Arteaga, and T. Schulthess, "Towards a performance portable architecture agnostic implementation strategy for weather and climate models," *SuperFRI*, vol. 1, no. 1, Sep. 2014. [Online]. Available: http://dx.doi.org/10.14529/jsfi140103

[20] J. H. Chen, A. Choudhary, B. De Supinski, M. DeVries, E. Hawkes, S. Klasky, W. Liao, K. Ma, J. Mellor-Crummey, N. Podhorszki *et al.*, "Terascale direct numerical simulations of turbulent combustion using S3D," *Computational Science & Discovery*, vol. 2, no. 1, p. 015001, 2009.

[21] T. Hoefler and T. Schneider, "Optimization principles for collective neighborhood communications," in *2012 International Conference for High Performance Computing Networking, Storage and Analysis*. Institute of Electrical & Electronics Engineers (IEEE), Nov. 2012. [Online]. Available: http://dx.doi.org/10.1109/sc.2012.86