On Network Locality in MPI-Based HPC Applications

Felix Zahn
felix.zahn@ziti.uni-heidelberg.de
Heidelberg University
Heidelberg, Germany

Holger Fröning
holger.froening@ziti.uni-heidelberg.de
Heidelberg University
Heidelberg, Germany

ABSTRACT

Data movements through interconnection networks exceed local memory accesses in terms of latency as well as energy by multiple orders of magnitude. While many optimizations make great effort to improve memory accesses, large distances in the network can easily dash these improvements, resulting in increased overall costs. Therefore, a deep understanding of network locality is key for further optimizations, such as improved mapping of ranks to physical entities.

In this work, we are looking at locality in the hardware-independent application level and at locality aspects of common network structures. In order to quantize the former, two new metrics are introduced, namely *rank locality* and *selectivity*. Our studies are performed on a selection of 16 exascale proxy mini apps, with a scale ranging from eight to 1152 ranks. These traces are statically analyzed regarding their spatial communication pattern at the MPI level. The resulting practice in actual hardware is evaluated with a network model, which implements topologies such as tori, fat tree, and dragonfly, and an according minimal routing. As a result, this work is founded on a large set of experimental configurations, based on different applications, scales, and topologies.

While in most traces single ranks have a wide range of commutation partners, 90% of the communication is exchanged only with a small set of ten or fewer other ranks. Results suggest the 3D torus as the most favorable topology for a small number of ranks, while for larger configurations the fat tree is preferable. Furthermore, we show that in general, the network is highly underutilized and that in 93% of all configurations less than 1% of network resources are actually used. Overall, this indicates that static analyses could assist to select an advanced mapping, which assigns groups of heavily communicating ranks to nearby physical entities. This could help to minimize the total number of packet hops and, thereby, improve latency and reduce the probability of congestions.

CCS CONCEPTS

• **Computing methodologies** → **Parallel programming languages**; • **Networks** → **Network performance analysis**.

KEYWORDS

locality, characterization, MPI, interconnection networks

1 INTRODUCTION

At the end of further frequency scaling, trends in high-performance computing (HPC) shift to an increase of parallelism to maintain Moore's Law. More parallelism enables the integration of specialized hardware and new architectural approaches in order to speed up particular operations. Along with these benefits, parallelism also increases the demand for more communication, since applications need to spread their data across the system and keep them synchronized.

However, data movements are costly regarding both time and energy. While inside the memory hierarchy data movement costs increase with different levels, they are even several magnitudes higher when data passes the interconnection network. Reducing this network overhead requires a deep understanding of communication and topological characteristics and their correlation.

Particularly, if the communication pattern exhibits sufficient locality, tuning the mapping of ranks to physical nodes for a certain application and topology can reduce network-level data movements substantially. Thus, on the application side, probably the most important characteristic is the application's communication pattern. Especially locality, in terms of communication distance and volume between distinguished ranks, affects performance and data movement costs. The distance between two ranks, respectively the number of hops, as well as the injected data, are the measures for the actual amount of traffic propagating on the network.

When locality and communication patterns are studied so far, these explorations focus either on memory locality or abstract density plots, which provide good human readability, but are not suitable for further analyses and abstract comparisons. Closest to this purpose is the work of Kim et al. [6], which investigates communication event locality, message destination locality, and message size locality. However, these analyses are more than 20 years old and the authors report that their metrics are oblivious to variations in system configuration and problem size.

In order to gain a better understanding of locality, the presented work introduces objective metrics for locality that allow a better understanding of communication patterns. While these metrics are derived at MPI level, we provide further studies, how this applies to certain hardware configurations. These studies include the impact of multi-core scaling, as well as the effects of different topologies. We provide insights on which application benefits from a low-diameter topology, such as dragonfly, which need high bisection bandwidth, for instance in a fat tree, and which application are exploiting the three-dimensional character of a torus.

In particular, we make the following contributions:

- Introduction of *rank locality* and *selectivity* as high-level metrics which describe communication volume and distance between ranks in MPI traces in a hardware-agnostic view.
- Analyses of *topological locality* for different hardware configurations. In particular, how different topologies affect locality regarding average hop distance and packet hops.
- A qualitative comparison of high-level metrics with topological locality as ground truth to assess the fitness of the high-level metrics as an abstract workload characterization.

The remainder of this work is structured as follows: Section II provides short background information about MPI communication and basic network properties, followed by a brief overview of related works. In Section IV, the methodology of this analysis is provided. In particular, we introduce new metrics to describe network locality. The next Sections (V& VI) reports the locality results for MPI level studies and topology level, respectively. Section VII discusses the findings, followed by a summary in Section VIII.

2 BACKGROUND

This section provides a brief overview of the data structures and technologies on which our analyses are based. First, on the software side, more information about the input data is provided, followed up by the hardware properties underlying these studies.

2.1 Traces

Analyses on network locality require input data about the data volume that is transferred on the interconnection network. Since the Message Passing Interface (MPI) is the de-facto standard for data exchange and synchronization in HPC, this input data is acquired by tracing MPI calls when executed on real systems.

2.1.1 Message Passing Interface. MPI is an abstract interface for transmitting data and synchronization purposes in parallel applications. Therefore, the application starts multiple simultaneously running processes (ranks), which are communicating via MPI messages. The data exchange between two ranks in a point-to-point (p2p) fashion is represented by an `MPI_Send()` and an `MPI_Recv()` call, respectively, that contain detailed information about the transmitting data, such as size (e.g. data type and number of elements), source and destination memory addresses, and a communicator. The communicator describes the set of eligible ranks that take part in the communication. In addition to this point-to-point communication, MPI provides also collective operations for communication between a group of ranks. These collective operations are often used for synchronization purposes, but also allow for shared data processing or data distribution. In the context of collective operations, the participating ranks are determined by the communicator.

2.1.2 Dumpi Trace Format. The dumpi trace format was introduced to gather more detailed information about MPI calls than other available trace formats. It was developed as part of the SST/macro simulator by the U.S. Sandia National Laboratories [1], which also provide a rich repository of traces from various applications [3]. Besides CPU and wall time for entering and leaving an MPI routine, dumpi traces contain every MPI function call along with its parameters.

2.2 Interconnection Network

While MPI provides the software interface for data transfer, the actual data transmission is performed by the interconnect.

2.2.1 Data Transfer. The interconnection network is composed of two major components: switches and links, where a link is a connection between two switches or a switch and a node. Usually, when packets travel through the network, they have to pass multiple links and switches. Every time a packet is transmitted on a link it is referred to as hop. Internal switching logic, such as crossbar, routing units, or arbiter, does not consume significant power (~15%) compared to the SerDes, which convert the parallel internal into a serial external data stream (~85%) [19]. Because current interconnection technology consumes power statically at all time, links are a sweet spot for power and energy saving.

2.2.2 Topology. This work focuses on three of the most common topologies that also represent different topology types, such as direct/indirect and hierarchical/non-hierarchical networks.

3D Torus. A 3D torus is a direct topology in which nodes are arranged similar to a 3D mesh with wrap around links that connect the first and last node of every dimension. Therefore, every dimension can be seen as a ring instead of a chain, which reduces the diameter. The main feature of a direct topology is the integration of a switch inside a node. Since switches are connected directly to the PCIe bus, there is no additional hop to the switch required. While tori are highly modular in terms of scaling, they scale not linearly in bisection bandwidth and diameter.

Fat Tree. The fat tree topology is an indirect, tree-based interconnection pattern, which provides the same bisection bandwidth at every stage [10]. Therefore, the branches become thicker at the same scale as there are leaves connected to a switch. To ensure a constant bisection bandwidth, every stage provides the same amount of switches in which half the links are used for the upward and the other half for the downward direction. The top state is an exception, where only half the switches are used to connect all child switches. Last, all compute nodes are connected as leaves in the last stage.

Dragonfly. The dragonfly topology was first introduced by J.Kim et al. [5] as a hierarchical, indirect topology, which provides a low diameter and minimizes the usage of costly optical links. The concept is based on multiple groups, in which all nodes are connected to according switches via local links and all groups are connected via global links with each other. Although there is no strict rule for the connection pattern, the following rule is recommended to ensure a balanced channel load: there should be twice as many routers in each group (a) as nodes connected to each router (p), and links within each router used to connect to other groups (h). Particularly, in the context of this work, the groups are connected in a palm tree pattern.

3 RELATED WORK

The fundamental problem of using models to represent the communication behavior of parallel applications is summarized by Singh et al. [17]. In particular, it is incomplete to describe communication without regarding its relationship to the parallel computation. Since irregular, dynamically behaving applications are commonplace, simulation tools are mandatory for an in-depth understanding of communication behavior.

Locality characterization in HPC application has up to now mostly been used in the context of memory locality, i.e. which

[1] https://github.com/sstsimulator/sst-dumpi

amount of locality is found and exploited. Murphy et al. [12] characterize the sensitivity of HPC applications to memory latency and bandwidth, which is also a question of locality. In another work, the authors report a similar analysis of working set (size) [14]. Furthermore, they provide a formal definition of temporal and spatial locality and an analysis of those metrics for HPC apps, including SPEC and benchmarks from Sandia [13]. Further work on memory locality is provided by Ibrahim et al. [4]. Manian et al. analyze locality in the context of GPU allocations and MPI [11].

Regarding parallel communication characterization, Chodnekar et. al. [2] introduce a couple of metrics related to communication characterization, including message generation frequency, the spatial distribution of messages, and message length. By using statistical regression analysis of the log of network activity, the message inter-arrival time and spatial distribution are determined. While the methodology is reported as feasible for a couple of workloads, the authors also note some particular limitations if there are no distinct phases of an application, as for multigrid applications. MPI communication locality is furthermore studied by Kim et al. [6], with particular treatment of communication event locality, message destination locality, and message size locality. For a limited set of rather old benchmarks (NAS [1] and MICOM [16]), the authors observe that the proposed metrics are relatively insensitive to system and problem size variations.

Characterizing communication patterns can also shed light on locality. In this context, a variety of related work exists, for instance [8, 9, 15, 18]. However, such analyses are usually limited to rather abstract density plots characterizing communication calls and volume. In the best extend, a visual analysis is required to understand to which amount locality is present. One of the most recent works is in this context is presented by Klenk et al. [7], which also used exascale proxy apps as underlying workload set for detailed characterization. Contrary to locality, this work focuses on abstract MPI behavior instead.

4 METHODOLOGY

In order to gain deep insights into the communication patterns of HPC exascale applications, a wide set of MPI traces is analyzed. The following section describes the selected applications as well as the analyzing procedure. Furthermore, a set of new metrics is introduced to characterize different locality aspects of the communication behavior of these applications.

Locality in MPI-based applications is mostly characterized by communication patterns represented in heat maps so far. While this is well suited for humans to understand patterns of a small number of ranks, heat maps become increasingly unclear with the number of ranks. Additionally, they are not qualified to be interpreted abstractly. As shown in related work, communication analysis still lacks more detailed metrics to describe the locality of communication patterns. We introduce the following metrics to close this gap. In particular, these metrics aim to understand the spatial distribution of messages from a network point of view and how it can be utilized for further improvements.

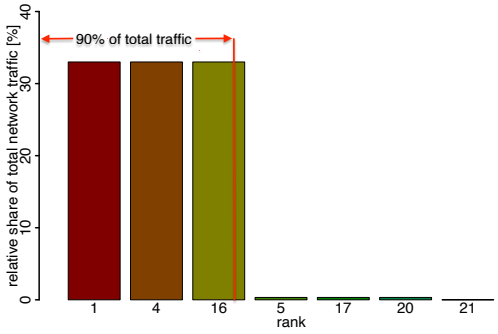

Figure 1: Illustration of selectivity metric. For an exemplary rank (LULESH, rank 0), the communication volume (y axis) to every other communication partner (x axis) is shown.

4.1 MPI Level

At MPI level, there is no information about the actual path a message is taking in the network, its particular length or the network's utilization. But there are still useful insights about locality that can be found in this layer. For example, the identification of certain communication partners or groups could help to improve mapping and reduce the overall number of hops.

4.1.1 Locality. Rank locality represents the spatial distance of communication at the MPI level which means the communication distance between two distinguished MPI ranks. Given a rank sourcing the communication $rank_{source}$, a rank sinking the communication $rank_{dest}$, and based on a linearization of the different rank IDs according to their numerical ID, the *distance* between two ranks and the according locality are defined as

$$dist = |rank_{source} - rank_{dest}| \tag{1}$$

$$locality = \frac{1}{dist} \tag{2}$$

Thus, lower distances describe a higher locality and vice versa, where a distance of one (i.e. communication with direct neighbors) results in a locality of 100%. Note that this is an abstract metric, which neglects topology and mapping, and corresponding effects. Still, it can be evaluated based on traces, with no replay necessary.

To quantize this metric, we define *rank locality* as the maximum spatial distance for which 90% of the overall traffic is covered. Since only applications with global communicators are considered, collective communication can be assumed uniform between all nodes. As a consequence, only point-to-point messages are considered here.

4.1.2 Selectivity. Selectivity can also be derived completely from the MPI level. It describes how many partner ranks are dominating the communication of one rank in a certain application. For a given source rank, this metric is calculated by determining destination ranks and sort them by data volume exchanged between this pair of ranks. Figure 1 depicts the concept for an exemplary rank. The y-axis indicates the communication volume that is sent to every other rank and the x-axis shows the corresponding receiving ranks.

As locality, this metric also refers only to point-to-point communication in the context of this work. However, when including more diverse collective patterns, this type of communication should also be included for both metrics. Again, for quantization reasons, a threshold of 90% of the given rank's total communication volume was selected to describes how many destination ranks participate in the communication set of each rank. While many workloads establish at least little communication across all ranks, there are significantly fewer ranks that contribute to the majority of overall communication.

4.2 Topological Locality

Contrary to the MPI level, the network level allows estimating effects of network parameters, such as mapping, topologies, or routing, on communication patterns. In order to address topology and mapping effects based on rank locality, we define topological locality as the distance in number of network hops between the source rank and the destination rank. Therefore, we provide non-temporal models for different topologies, including shortest-path routing algorithms.

The entire analysis is based on this model and not simulated. This means, without the temporal character of a simulation, the results do not contain any information about the interaction of traffic flows. For all calculations, we assume the full network capacity is available for every particular message. Although this representation is simplified, it allows deriving certain network values that are not affected by the interaction of messages.

4.2.1 Packet Hops. This metric indicates how many hops are performed by packets in the network until all data has reached its final destination. MPI messages are split in the according number of packets, with a maximum payload size of 4kB. The number of hops a particular packet has to traverse is calculated by the respective routing algorithm. Formally, this metric is described as:

$$packethops = \sum_{p \in packets} \#hops(p) \tag{3}$$

Due to the non-temporal character of this model and resulting the abstinence of congestions and load balancing, we selected shortest-path routing for all topologies in order to provide fairness and emphasize the impact of the particular topology. This metric is also highly depending on the injection rate of a given application, respectively its communication intensity. However, it can directly be translated to network latency and energy consumption.

4.2.2 Average Number of Hops per Packet. From the total amount of packet hops, the average number of hops can be derived as follows:

$$\overline{\text{hops}} = \frac{packethops}{\#packets} \tag{4}$$

This metric provides details about the mean distance a message has to travel in the network. It is particularly helpful when comparing different topology/application combinations. Furthermore, it is a measure for the efficiency of exploiting locality effects, for instance by a suitable mapping, as a lower value indicates a better locality for a certain topology/application combination.

4.2.3 Network Utilization. This metric depicts the share of execution time in which the network is actually transmitting data. It is also an indicator of the probability of congestions, including different increments in hierarchical topologies. This can provide useful insights for system architects about how to provision networks for particular applications. Although energy saving is not common in interconnection networks yet, this metric can also help to estimate the minimum energy that is required by the network, when energy-saving mechanisms are applied. We define network utilization as:

$$utilization = \frac{datavolume}{BW \cdot t_{execution} \cdot \#links} \tag{5}$$

To provide fairness for all topologies, only links and switches are considered that are actually transmitting data, in particular when the configuration provides more nodes than the application has ranks. For the fat tree, the total number of links equals $\#nodes \cdot \#stages$ (only half the links for the last stage), the torus has three links per node, which equals one per dimension, assuming that switches are integrated into the NIC. The standard dragonfly is configured with the same amount of global links as nodes are connected to each router and twice as many routers per group. This results in 3.5 to 3.8 links per node in this study, depending on the particular configuration. We assume $BW = 12GB/s$ to be realistic for a representative interconnection network.

4.3 Applications

The selection of HPC applications aims to cover multiple different communication types and patterns at different scale. Exascale mini apps, introduced by the DOE, hit all these criteria. A repository maintained by Sandia National Laboratories provides traces of these mini apps in the dumpi format[2]. While this repository contains a multitude of applications at different scale, we select the traces shown in Table 1 for our analyses. Table 1 also provides a brief overview of the fundamental MPI characteristics, such as communication volume (Vol.) of both collective and point-to-point communication, total execution time, and the according throughput (Vol./t). Applications that are marked with a star (*) make use of MPI Derived Data Types. Since the dumpi repository does not contain any header or other information about the actual size of the used data types, we selected one byte as the according size. Although we assume actual size to be bigger, this enables scaling results easily to the actual size if needed. Furthermore, some applications make use of custom communicators, in particular those representing custom carts of ranks (i.e., as a result of `MPI_Cart_create`), or subtracting certain ranks (`MPI_Cart_sub`). Changes in the communicator affect the identification of particular ranks, due to another lack of information in the traces. Therefore, it cannot be ensured that ranks are still mapped to their particular identifier. In order to remain consistent through the entire execution, traces with these types of collectives are not considered in the context of this work.

4.4 Hardware Parameters

To determine the number of hops a message has to travel in the network, a routing algorithm and formal definition of the particular

[2] https://portal.nersc.gov/project/CAL/doe-miniapps.htm

Application	Ranks	Time [s]	Vol. [MB]	P2P [%]	Coll. [%]	Vol./t [MB/s]
AMG	8	0.03	3.0	100.0	0.00	116.3
	27	0.16	13.6	100.0	0.00	86.98
	216	0.30	136.9	100.0	0.00	461.5
	1728	2.92	1208	100.0	0.00	413.7
ïż£ AMR Miniapp	64	12.93	3106	99.66	0.34	240.3
	1728	42.69	96969	99.45	0.55	2271
ïż£ BigFFT (Medium)	9	0.18	299.2	0.00	100.0	1659
	100	0.50	3169	0.00	100.0	6340
	1024	1.89	32064	0.00	100.0	17003
ïż£ Boxlib CNS large (*)	64	572.19	9292	100.0	0.00	16.24
	256	169.05	15227	100.0	0.00	90.08
	256	150.92	15227	100.0	0.00	100.9
	1024	67.54	34131	100.0	0.00	505.4
ïż£ Boxlib MultiGrid C	64	231.42	23742	99.94	0.06	102.6
	256	62.01	44535	99.95	0.05	718.2
	256	60.28	44535	99.95	0.05	738.8
	1024	20.88	75181	99.94	0.06	3600.9
ïż£ CESAR MOCFE (*)	64	0.38	19.0	5.01	94.99	50.3
	256	1.10	81.6	5.51	94.49	74.11
	1024	3.95	686.2	6.96	93.04	173.9
ïż£ CESAR Nekbone (*)	64	11.83	5307	100.0	0.00	448.8
	256	3.17	1272	50.66	49.34	401.8
	1024	5.15	13232	99.98	0.02	2568.8
ïż£ Crystal Router	10	0.14	133.8	100.0	0.00	930.3
	100	0.71	3439.9	100.0	0.00	4854
	1000	1.28	115521	100.0	0.00	90491
ïż£ EXMATEX CMC 2D ïż£ Multinode	64	842.80	16.0	0.00	100.0	0.0190
	256	208.44	16.1	0.00	100.0	0.077
	1024	58.85	16.4	0.00	100.0	0.279
ïż£ EXMATEX LULESH	64	54.14	3585	100.0	0.00	66.23
	64	44.03	3585	100.0	0.00	81.43
	512	50.24	33548	100.0	0.00	667.8
ïż£ FillBoundary	125	2.32	10209	100.0	0.00	4393
	1000	5.26	92323	100.0	0.00	17549
ïż£ MiniFE	18	59.70	1615	100.0	0.00	27.06
	144	61.06	16586	99.99	0.01	271.63
	1152	84.75	147264	99.96	0.04	1737.7
ïż£ MultiGrid_C	125	0.77	374	100.0	0.00	4889.0
	1000	3.57	2973	100.0	0.00	832.83
PARTISN (*)	168	2.2E+6	42123	99.96	0.04	0.02
SNAP (*)	168	1.2E+6	128561	100.0	0.00	0.11

Table 1: Overview of MPI-based exascale proxy applications

topology are required. We assume six-port NICs that are directly integrated into nodes for the 3D torus and 48-port switches as components for the fat tree. The dragonfly configurations are selected accordingly to suggestions by J.Kim et al. [5], with $a = 2h = 2p$. An overview of all different configurations is shown in Table 2 (*rad* being radix, *st* being number of stages). Note that, especially for fat tree and dragonfly, not all topology configurations can be chosen accordingly to the number of ranks. To provide fairness in the utilization comparisons, only actual used links in these configurations are taken into account. Additionally, shortest path routing is used for all three topologies, since traffic flows are not considered and, therefore, no load balancing is required.

Furthermore, the execution of collective operations in HPC systems can differ between multiple network technologies or vendors, due to custom implementations, such as special broad- and multicast support. However, the model in this work aims to be technology independent and follows a simpler and more robust approach: collectives are translated to point-to-point messages, which are sent in the pattern of the particular operation. For example, a gather call is performed by all ranks sending a p2p message to the root node. This means, there is no tree structure or similar to spread collectives over the network. Although this implementation often differs from today's hardware, our approach ensures that the network is maximally utilized to give a stable estimate. Notably, data in vector-based collectives is split evenly across all ranks.

	Torus		Fat Tree		Dragonfly	
Size	(x,y,z)	Nodes	(rad, st)	Nodes	(a,h,p)	Nodes
8	(2,2,2)	8	(48,1)	48	(4,2,2)	72
9	(3,2,2)	12	(48,1)	48	(4,2,2)	72
10	(3,2,2)	12	(48,1)	48	(4,2,2)	72
18	(3,3,2)	18	(48,1)	48	(4,2,2)	72
27	(3,3,3)	27	(48,1)	48	(4,2,2)	72
64	(4,4,4)	64	(48,2)	576	(4,2,2)	72
100	(5,5,4)	100	(48,2)	576	(6,3,3)	342
125	(5,5,5)	125	(48,2)	576	(6,3,3)	342
144	(6,6,4)	144	(48,2)	576	(6,3,3)	342
168	(7,6,4)	168	(48,2)	576	(6,3,3)	342
216	(6,6,6)	216	(48,2)	576	(6,3,3)	342
256	(8,8,4)	256	(48,2)	576	(6,3,3)	342
512	(8,8,8)	512	(48,2)	576	(8,4,4)	1056
1000	(10,10,10)	1000	(48,3)	13824	(8,4,4)	1056
1024	(16,8,8)	1024	(48,3)	13824	(8,4,4)	1056
1152	(12,12,8)	1152	(48,3)	13824	(10,5,5)	2550
1728	(12,12,12)	1728	(48,3)	13824	(10,5,5)	2550

Table 2: Configurations for different topologies at scale

5 APPLICATION-LEVEL LOCALITY

This analysis focuses on the application level and includes all metrics that can be derived directly at MPI level without further understanding of the underlying network parameters, such as topology, routing, or mapping. Since only global communicators are considered, collectives can be assumed as a constant bias on the network, while communication variations occur from point-to-point messages. Therefore, only the latter are considered for these analyses. Also note that during this work, all analyses are done statically without considering any timing aspects, including load balancing or possible congestion.

Table 3 provides an overview of all applications and results. Regarding point-to-point messages, many workloads tend to communicate only with a small subset of the global communicator. Therefore, the third column shows the number of *peers* for every workload. This metric, introduced by Klenk et al. [7], indicates the peak number of peer ranks any rank in this application is addressing during point-to-point communication. Particularly, *peers* is significantly smaller than the number of ranks (*ranks*) for most

workloads. This shows that communication happens only with a smaller subset of all ranks, however, this metric does not provide any insights about the distribution between peers.

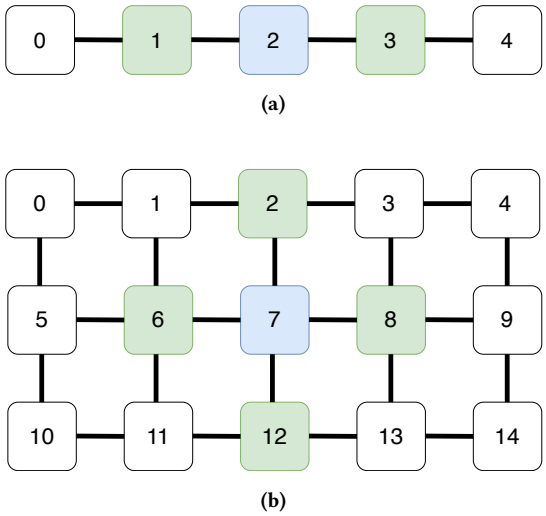

Figure 2: Nearest neighbors (green) of a particular node (blue) for one dimensional problem (a) and two dimensional problem (b)

5.1 Rank Locality

The first analyzed metric is an application's locality at the rank level. This metric indicates the average communication distance between ranks weighted by the transmitted data volume. The results for rank distances at all scales are provided in Table 3. Note, that rank locality is the reciprocal of the rank distance. One can already observe that actual communication is much more selective than the number of communication *peers* suggests. In particular, for applications for which *peers* is close to *ranks*, such as CNS or PARTISN, *rank locality* differs substantially, indicating a considerable amount of locality.

Nearest neighbor communication is a frequently used pattern that also provides good locality properties. This type of communication would translate to a *rank locality* of more than 50% (or a distance of less than 2) since the distances between the adjacent nodes is constant at all scale. Note that, scattered messages to other ranks are not considered since only the nearest 90% of data volume are considered here. However, this scheme is only true for one-dimensional workloads, as spatial distances in further dimensions are not covered by this linear metric. Figure 2 depicts the different neighbor schemes for one (Figure 2a) and two (Figure 2b) dimensions. While in the former scheme the linear numbering matches the spatial setting of neighbors, the latter shows further distances between ranks in the y dimension, for instance from rank two to seven. In particular, this results in a constant offset depending on the number of nodes per dimension.

Regarding the analyzed applications, the distance increases for all workloads with the number of ranks, which indicates that no application communicates in a one dimensional nearest neighbor pattern. Table 4 shows some exemplary results for *rank locality* in different dimensions. Most workloads show no special correlation to a particular dimension, as represented by CNS and MultiGrid_C here. Although locality improves for all applications with the number of dimensions, this is rather caused by a decreasing diameter than a particular correlation. The only workload that has a two-dimensional structure is PARTISN, as *rank locality*peaks when mapped to a 2D grid. The class of three-dimensional workloads includes LULESH, AMG and Neckbone (64 ranks), which results in a *rank locality* of 100%.

5.2 Selectivity

The *selectivity* metric indicates the number of ranks that make up the largest part (here set to 90%) of the overall point-to-point communication. For this metric positions and layouts are neglected and single rank pairs are sorted by data volume exchanged. Figure 3 depicts the trends for each application, with several ranks sorted according to exchanged data volume on the x-axis, and y-axis reporting the relative amount of communication volume. In particular, *selectivity* is the point in which a curve cuts the 90% traffic share. Although most applications differ in their communication patterns there is a clear trend that 90% of the communication originates from only six or even fewer ranks. Only AMR (1728 ranks), CNS (1024 ranks), MOCFE (256 and 1024 ranks), and Nekbone (1024 ranks) have a *selectivity* larger than ten. Even in the largest configuration of 1728 ranks, 90% of the total point-to-point communication of each rank is still limited to a maximum of 13 ranks (AMR).

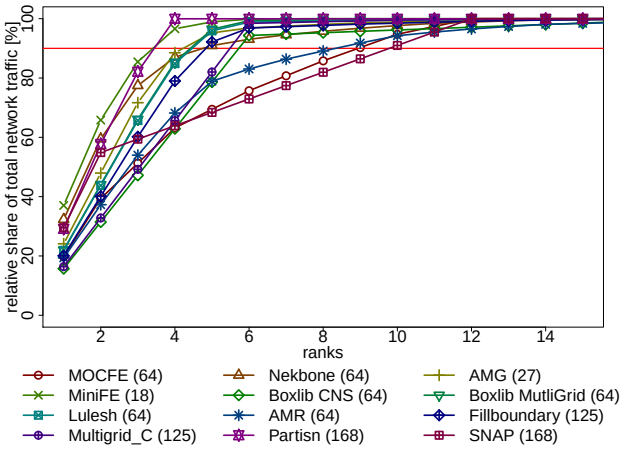

Figure 3: Selectivity trends for all workloads

Additionally, the overall trend is a slightly increasing *selectivity* with the number of ranks. However, while scaling ranks, the increase in *selectivity* is also slowing down, indicating a saturation. Figure 4 exemplarily depicts this trend. The curve of the four AMG application's sample sizes is shifting to the right with an increasing number of ranks. Although other applications vary in slope at a different scale, they show similar trends. In particular, there are three workloads (CNS, MOCFE, and MultiGrid_C) that slightly decrease *selectivity* when increasing the number of ranks.

		MPI Level Metrics			3D Torus			Fat Tree			Dragonfly		
Workload	Ranks	Peers	Rank Distance (90%)	Selectivity (90%)	Packet Hops	$\overline{hops}$	Utilization [%]	Packet Hops	$\overline{hops}$	Utilization [%]	Packet Hops	$\overline{hops}$	Utilization [%]
AMG	8	7	3.7	2.8	4.2E+03	1.57	0.0052	5.7E+03	2.00	0.0303	8E+03	2.83	0.0116
	27	26	8.7	4.2	2.9E+04	1.74	0.0012	3.5E+04	2.00	0.0034	7E+04	4.01	0.0034
	216	127	35.8	5.2	5.5E+05	2.36	0.0008	8.2E+05	3.41	0.0032	1E+06	4.14	0.0021
	1728	293	143.8	5.6	6.0E+06	2.62	0.0001	8.5E+06	3.62	0.0004	1E+07	4.28	0.0002
ïż£AMR_Miniapp	64	39	27.1	8.3	5.9E+06	2.93	0.0034	6.6E+06	3.20	0.0058	9E+06	4.19	0.0048
	1728	490	348.3	13.0	8.9E+09	8.97	0.0278	4.9E+09	4.86	0.0229	5E+09	4.74	0.0119
ïż£ BigFFT (Medium)	9	N/A	N/A	N/A	1.0E+06	1.56	0.6721	1.2E+06	1.78	3.0725	2E+06	2.91	1.2943
	100	N/A	N/A	N/A	7.7E+07	3.40	7.4849	2.7E+08	3.52	10.5544	3E+08	4.36	7.6985
	1024	N/A	N/A	N/A	6.4E+10	8.00	47.2317	3.5E+10	4.35	38.4346	4E+10	4.69	22.1491
ïż£ Boxlib CNS large (*)	64	63	35.1	5.7	5.7E+06	2.99	0.0002	6.5E+06	3.23	0.0003	9E+06	4.23	0.0003
	256	255	109.2	5.4	1.5E+07	4.93	0.0004	1.2E+07	3.75	0.0005	2E+07	4.49	0.0004
	256	255	109.2	5.4	1.5E+07	4.93	0.0005	1.2E+07	3.75	0.0006	2E+07	4.49	0.0004
	1024	1023	661.5	20.8	1.1E+08	7.97	0.0012	6.4E+07	4.35	0.0010	7E+07	4.68	0.0006
ïż£ Boxlib MultiGrid C	64	26	27.1	4.4	2.6E+07	2.92	0.0011	3.0E+07	3.19	0.0020	4E+07	4.19	0.0017
	256	26	54.3	4.4	3.9E+08	4.94	0.0035	3.0E+08	3.76	0.0045	4E+08	4.50	0.0032
	256	26	54.3	4.4	3.9E+08	4.94	0.0036	3.0E+08	3.76	0.0046	4E+08	4.50	0.0033
	1024	26	109.1	4.9	8.9E+09	7.96	0.0106	4.9E+09	4.33	0.0092	5E+09	4.67	0.0054
ïż£ CESAR MOCFE (*)	64	12	51.3	8.9	2.4E+06	2.96	0.0498	2.7E+06	3.28	0.0769	3E+06	4.24	0.0605
	256	20	195.3	14.0	6.2E+07	4.96	0.1216	4.7E+07	3.80	0.1368	6E+07	4.53	0.0895
	1024	20	771.8	13.3	3.2E+09	7.98	0.4495	1.7E+09	4.36	0.3656	2E+09	4.69	0.2108
ïż£ CESAR Nekbone (*)	64	27	15.8	4.8	4.0E+07	2.92	0.0027	4.6E+07	3.25	0.0090	6E+07	4.24	0.0081
	256	15	28.4	5.4	1.2E+09	4.99	0.3447	9.0E+08	3.80	0.3882	1E+09	4.53	0.2541
	1024	36	127.9	10.2	2.5E+10	7.96	0.0029	1.4E+10	4.35	0.0057	1E+10	4.69	0.0035
ïż£Crystal Router	10	4	6.4	3.0	2.4E+05	1.74	0.0469	2.7E+05	2.00	0.1938	4E+05	3.18	0.0882
	100	8	44.3	5.8	1.4E+06	2.41	0.0408	7.4E+06	2.76	0.0637	1E+07	3.61	0.0490
	1000	11	334.3	8.9	2.8E+08	4.69	0.1475	1.9E+08	3.26	0.1531	2E+08	3.82	0.0959
ïż£ EXMATEX CMC 2D ïż£ Multinode	64	N/A	N/A	N/A	7.9E+05	3.00	2.0E-05	8.4E+05	3.28	3.0E-05	1E+06	4.25	2.4E-05
	256	N/A	N/A	N/A	5.2E+06	5.00	0.0001	4.0E+06	3.81	0.0001	5E+06	4.54	0.0001
	1024	N/A	N/A	N/A	3.4E+07	8.00	0.0008	2.0E+07	4.36	0.0007	2E+07	4.69	0.0004
ïż£EXMATEX LULESH	64	26	15.7	4.5	2.3E+06	2.70	0.0004	3.8E+06	3.17	0.0013	5E+06	4.18	0.0011
	64	26	15.7	4.5	2.3E+06	2.70	0.0004	3.8E+06	3.17	0.0016	5E+06	4.18	0.0013
	512	26	63.7	5.0	1.7E+08	5.80	0.0005	1.3E+08	3.88	0.0020	2E+08	4.60	0.0012
ïż£FillBoundary	125	26	42.3	4.8	6.6E+06	3.27	0.0319	6.9E+06	3.32	0.0466	9E+06	4.13	0.0351
	1000	26	219.1	5.3	9.9E+07	7.13	0.0245	6.6E+07	4.15	0.0248	8E+07	4.55	0.0160
ïż£ MiniFE	18	8	7.4	3.4	8.9E+05	1.82	0.0008	1.1E+06	1.90	0.0031	2E+06	3.69	0.0015
	144	22	31.5	4.6	4.5E+07	3.97	0.0017	4.2E+07	3.62	0.0025	5E+07	4.40	0.0017
	1152	22	91.8	5.1	4.6E+09	7.98	0.0039	2.6E+09	4.47	0.0037	3E+09	4.71	0.0022
ïż£MultiGrid_C	125	22	59.7	5.5	1.2E+06	3.52	0.0038	1.3E+06	3.57	0.0056	2E+06	4.33	0.0041
	1000	22	392.0	5.4	1.0E+08	7.43	0.0013	6.0E+07	4.31	0.0013	7E+07	4.66	0.0008
PARTISN (*)	168	167	13.8	3.4	8.0E+07	2.70	7.4E-08	1.0E+08	3.04	1.6E-07	1E+08	3.88	1.2E-07
SNAP (*)	168	48	139.1	9.8	1.6E+08	3.85	4.2E-07	1.5E+08	3.74	6.2E-07	2E+08	4.41	4.0E-07

Table 3: Workload characteristics in different locality-describing metrics.

Furthermore, it appears that there are two classes of workloads. The first one, including Boxlib MultiGrid C, Crystal Router, LULESH and Multigrid_C, have a constant ratio between their *selectivity* and number of peers at all scale. The other group includes the remaining applications, which have more variance when scaling.

6 SYSTEM-LEVEL LOCALITY

After looking at the application level, system level effects are analyzed in this section. In particular, this includes the impact of different hardware configurations on locality. Contrary to the abstract metrics in the previous section, mapping on a particular network topology provides precise results about the distances in terms of

Workload	Ranks	Rank Locality 1D	2D	3D
AMG	216	3%	17%	100%
	1728	1%	8%	100%
ïż£Boxlib CNS large	64	3%	13%	21%
	256	1%	8%	13%
	1024	0%	3%	7%
ïż£EXMATEX LULESH	64	6%	24%	100%
	512	2%	6%	100%
ïż£MultiGrid_C	125	2%	6%	17%
	1000	0%	3%	9%
PARTISN	168	7%	100%	22%

Table 4: Exemplary workloads for different dimensionalities in rank locality

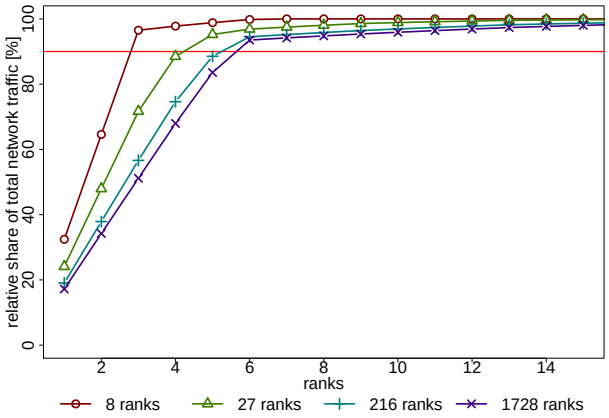

Figure 4: Scalability of selectivity (example: AMG)

hops and traffic volume in the network. These metrics can, for instance, directly be translated to latencies or energy consumption and, therefore, provide useful insights for further improvements in the system's design.

6.1 Multi-Core Effects

First, the impact of multi-core systems, respectively the ratio of cores to network endpoints, on network traffic is studied. Recent trends include an increasing number of cores per socket, allowing to execute more ranks on the same node and, thereby, reducing the amount of data that has to be exchanged on the interconnection network.

Figure 5 depicts this scaling behavior for all applications that are available with a configuration of at least 512 ranks. Smaller configurations are not considered here since a problem size in the same magnitude as the number of cores would sophisticate scaling effects. The x-axis indicates the number of cores per socket, where one core executes one rank. The y-axis shows the amount of inter-node traffic relative to the one-rank-per-node configuration. Here,

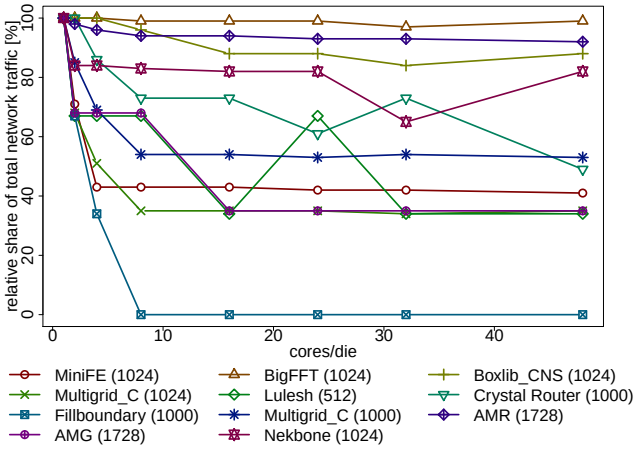

Figure 5: Network traffic for different cores-per-socket configurations

traffic includes both point-to-point and collective messages. Note that this study is topology-independent and only indicates how much traffic is transmitted on the network, no matter the distance between source and destination. Furthermore, a simple mapping is used in which the number of ranks is consecutively mapped to one node, according to the number of cores.

The variations in the course of the scaling are caused by mapping effects. Since there are no optimizations done, a larger number of cores per socket can split up a pair of ranks, which are heavily communicating, while the same pair is located at the same node at another configuration. However, the smaller values indicate the optimum scaling here. Surprisingly, although the level of inter-node traffic differs significantly between all applications, they all reach their saturation at 8-16 cores per socket. This indicates that from a network point of view, the optimum for minimizing network traffic is reached at 16 cores per socket and there are no evident improvements when further scaling, as long as the application's scale is much larger than the number of cores.

6.2 Topological Locality

The second part of the system-level studies focuses on how a topology affects locality in terms of *packethops* and average number of hops ($\overline{hops}$). These analyses are performed with a simple mapping of one rank per node. Additionally, collective communication is translated into point-to-point messages, as explained in section 4. The results for all configurations are provided in Table 3.

3D Torus. The 3D torus provides a fitting configuration for every problem size due to its high modularity. Except for the SNAP application, a torus provides the lowest average number of hops for all small problem sizes (<256 *ranks*). Contrary, for a torus the $\overline{hops}$ scales stronger with *ranks* than for other topologies, most likely due to an accordingly increasing diameter. AMG is the only application not following this trend. Although there are more applications with a *ranklocality* of 100% in a 3D grid, the constant overhead caused by collective operations increase the $\overline{hops}$. Throughout all workloads,

both $\overline{hops}$ and *packethops* scale with *ranks*, as distances and overall communication increases.

Fat Tree. Three different configurations suffice to map all configurations onto this topology. The deliberately high switch radix of 48 allows to set up large systems with only a few stages. The mapping here is performed consecutively, which allows ignoring unused nodes without affecting the results due to the tree-based character of this topology. Generally, the fat tree provides good scalability and $\overline{hops}$ only slightly increases with *ranks*. The maximum average hop distance is 4.47 for MiniFE at 1152 ranks. Surprisingly, except for AMR (1728 ranks), the fat tree provides always smaller $\overline{hops}$ and a smaller number of packet hops than the low-diameter dragonfly.

Dragonfly. Similar to a fat tree, a dragonfly is limited to only four different configurations. Although the mapping is again done incrementally, it seems to have a higher impact here. The number of hops can vary from two in the best case to five in the worst case, for all configurations. A dragonfly is known as a low diameter topology, however, the $\overline{hops}$ remains only one time below the value of three for a small configuration of only eight links, while it is close to the maximum of five hops for most other cases. This indicates that, because of the relatively small group size, locality effects cannot be exploited here and most messages are exchanged between different groups. In particular, on average 95% of all messages overall applications use a global inter-group link.

6.3 Network utilization

Network utilization is a useful parameter to estimate the dimensioning of the network and minimal energy demand. The best topology/application combination, as well as a smart mapping, are the most suitable parameter to reach a reasonable utilization.

The first observation is that overall topologies, there is only one application (BigFFT) that actually utilizes the network more than 1%, or in other words: for all but one application, 99% of the total execution time, links are idling. Comparing the different topologies, the general trend is that for most applications, a fat tree shows the highest utilization, while especially for a large number of ranks, a torus' utilization exceeds the one of both fat tree and dragonfly.

7 DISCUSSION

At the hardware-agnostic application level, we introduced *rank locality* and *selectivity* as new metrics to describe locality. While *selectivity* for most workloads is pretty low, even compared to the number of peers, *rank locality* decreases significantly with the number of ranks. This indicates that although there are just few distinct communication partners, the data distribution in exascale apps is not limited to neighbor ranks. One important factor for the communication pattern seems to be the dimensionality of the underlying problem. This observation is also backed up by the multi-core study, which shows that there is still a lot of inter-node network traffic, even when using 48 cores/socket. Consequently, mapping ranks consecutively to a given topology does not exploit the benefits of a small *selectivity*, since communication partners are likely spatially separated. In order to further reduce network traffic and profit from faster on-chip communication, a deeper understanding of communication pairs is necessary. The low *selectivity* of most applications suggests that a significant traffic reduction is possible only by using an optimized mapping. Traffic can further be reduced by tailoring the network topology to a given application, as the dimensionality analyses suggest. However, the best approach and the degree of possible optimization is highly depending on the application.

The dragonfly is often considered as a low-diameter topology. However, its design prevents it from exploiting locality that is present at the application level. This results in the highest average number of hops for most configurations. As the maximum distance for a dragonfly is bound to five hops, larger configurations might be beneficial for dragonflies. The reason for the poor performance of small configurations is probably the links/node ratio in the standard configuration: with as many global links as nodes and twice as many switches per group, the ratio of links/node varies from 3.5 to 3.8 for the used configurations, while a torus has a constant ratio of 3 and a fat tree one of below three. Also note, that this study considers shortest-path routing, while in practice usually adaptive routing is used in dragonfly networks, which often results in even longer paths. Contrary, a 3D torus shows for many configurations the best locality properties, despite a rather large diameter. This hints that the three-dimensional character of many applications is a good fit for a torus. However, at larger scales of 256 ranks or more, the increasing diameter becomes a dominant factor.

Although it would be preferable to derive topology effects directly from the MPI layer, there is no explicit absolute correlation between application-level and system-level locality. A low *selectivity* and *rank distance* often indicate a 3D torus to be the best fit, but this does not hold true for all applications. Generally, although there are many indications, it is hardly feasible to drive absolute findings for all cases. However, the mini apps are selected to cover all kinds of potential exascale workloads with different communication and computation behaviors. Therefore, it is reasonable to assume that they differ significantly regarding their locality properties. Further studies with different subsets of similar applications could be useful to identify correlations between these metrics.

Generally, the network utilization is very low for all configurations, suggesting that a lot of energy is wasted in the interconnection network. As a reminder, most interconnection networks still operate with constant power consumption, independent their utilization. While advanced energy-saving mechanisms could probably optimize energy efficiency, most current hardware does not support them. Another approach could be operating at lower throughput, as reducing the operating frequency should super-linearly decrease power consumption. In this regard, further studies about the slackness in MPI applications could be useful, representing how much leeway a message has before the corresponding receive becomes blocking. This would allow operating links with higher utilization, such as global links in dragonflies, at a higher bandwidth than the seldomly used local links.

8 SUMMARY

Transferring data on interconnection networks is neither efficient in terms of execution time nor energy. Hence, reduced network usage is desirable and can be achieved at many levels.

Looking at the application level, there are two main observations. First, in all applications the majority of p2p communication happens

only between a small set of ranks, i.e. they have a significantly smaller *selectivity* than their number of ranks and even their peers. In 89% of all configurations, these sets include less than ten ranks. This suggests, that a smart mapping could dramatically reduce network traffic, which would improve performance and also enable power saving by scaling down network capacities. Second, the low *rank locality* indicates that these sets of heavily communicating ranks are not spatially grouped but spread overall ranks.

Independent of the low *rank locality*, the topological locality increases significantly. This is caused by the multidimensional character of all studied applications, which is substantiated by the dimensionality studies in *rank locality*. A 3D torus seems to be the most suitable topology for small workloads with $ranks \leq 100$, while for larger configurations the advantage of having a lower diameter protrudes. Although the dragonfly is well-known as low-diameter topology, the average numbers of hops are mostly higher than for the other topologies. This indicates that the dragonfly cannot exploit the inherent workload locality properly, due to the comparable low group size in the standard-setting. As a disclaimer, we want to note that this study is solely based on a static analysis of traffic patterns but not dynamic data of network monitoring. We believe the introduced metrics to be of value for the community in spite of being solely statically analyzed, it seems very promising to address dynamic effects in future work.

Our studies show, that the maximum network utilization for all topologies is noticeable low, where all but one application utilize their links not more than 1% of the execution time. Although static analyses are not eligible to provide accurate performance results they present and upper limit for the maximum utilization on a given topology. A higher utilization would be a better use of resources, presumably despite an increased risk of interferences between packets. Overall, these results suggest that exploiting locality in combination with a network of reduced bandwidth could be a suitable approach to reduce energy consumption and provide a higher utilization without affecting performance.

ACKNOWLEDGMENTS

We highly acknowledge the funding by the Carl-Zeiss-Foundation in form of a scholarship.

REFERENCES

[1] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, et al. 1991. The NAS parallel benchmarks summary and preliminary results. In *Supercomputing'91: Proceedings of the ACM/IEEE conference on Supercomputing*. 158–165.

[2] S. Chodnekar, V. Srinivasan, A. S. Vaidya, A. Sivasubramaniam, and C. R. Das. 1997. Towards a communication characterization methodology for parallel applications. In *Proceedings Third International Symposium on High-Performance Computer Architecture*. 310–319.

[3] D. Dechev and G. Hendry. 2013. A Macroscale Simulator for Exascale Software/Hardware Co-Design.

[4] K. Z. Ibrahim, S. Hofmeyr, and C. Iancu. 2011. Characterizing the Performance of Parallel Applications on Multi-socket Virtual Machines. In *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. 1–12.

[5] J. Kim, W. J. Dally, S. Scott, and D. Abts. 2008. Technology-Driven, Highly-Scalable Dragonfly Topology. In *International Symposium on Computer Architecture*. 77–88.

[6] J. Kim and D. Lilja. 1998. Characterization of communication patterns in message-passing parallel scientific application programs. In *Network-Based Parallel Computing Communication, Architecture, and Applications*. Springer, Berlin, Heidelberg, 202–216.

[7] B. Klenk and H. Fröning. 2017. An Overview of MPI Characteristics of Exascale Proxy Applications. In *High Performance Computing - 32nd International Conference, ISC High Performance, Frankfurt, Germany, June 18-22, 2017*. 217–236.

[8] S. Lammel, F. Zahn, and H. Fröning. 2016. SONAR: Automated Communication Characterization for HPC Applications. In *High Performance Computing*. Springer International Publishing, Cham, 98–114.

[9] I. Lee. 2009. Characterizing communication patterns of NAS-MPI benchmark programs. In *IEEE Southeastcon 2009*. 158–163.

[10] C. E. Leiserson. 1985. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.* 10 (10 1985), 892–901.

[11] K. V. Manian, A. A. Ammar, A. Ruhela, C.-H. Chu, H. Subramoni, and D. K. Panda. 2019. Characterizing CUDA Unified Memory (UM)-Aware MPI Designs on Modern GPU Architectures. In *12th Workshop on General Purpose Processing Using GPUs* (Providence, RI, USA) *(GPGPU '19)*. ACM, New York, NY, USA, 43–52.

[12] R. Murphy. 2007. On the Effects of Memory Latency and Bandwidth on Supercomputer Application Performance. In *10th IEEE International Symposium on Workload Characterization*. IEEE Computer Society, Washington, DC, USA, 35–43.

[13] R. Murphy and P. Kogge. 2007. On the Memory Access Patterns of Supercomputer Applications: Benchmark Selection and Its Implications. *IEEE Trans. Comput.* 56, 7 (July 2007), 937–945.

[14] R. Murphy, A. Rodrigues, P. Kogge, and K. Underwood. 2005. The Implications of Working Set Analysis on Supercomputing Memory Hierarchy Design. In *19th International Conf. on Supercomputing* (Cambridge, MA) *(ICS '05)*. ACM, New York, NY, USA, 332–340.

[15] R. Riesen. 2006. Communication patterns [message-passing patterns]. In *20th IEEE International Parallel Distributed Processing Symposium.*

[16] A. Sawdey. 1995. Using the Parallel MICOM on SGI Multiprocessors and the Cray T3D. In *The MICOM User's Group Meeting.(February 1995)*.

[17] J. Singh, E. Rothberg, and A. Gupta. 1994. Modeling Communication in Parallel Algorithms: A Fruitful Interaction Between Theory and Systems?. In *6th Annual ACM Symposium on Parallel Algorithms and Architectures* (Cape May, New Jersey, USA) *(SPAA '94)*. ACM, New York, NY, USA, 189–199.

[18] S. Sreepathi, E. D'Azevedoand B. Philip, and P. Worley. 2016. Communication Characterization and Optimization of Applications Using Topology-Aware Task Mapping on Large Supercomputers. In *7th ACM/SPEC on Int. Conf. on Performance Engineering* (Delft, Netherlands) *(ICPE '16)*. ACM, New York, NY, USA, 225–236.

[19] F. Zahn, P. Yebenes, S. Lammel, P. J. Garcia, and H. Fröning. 2016. Analyzing the Energy (Dis-) Proportionality of Scalable Interconnection Networks. In *2nd IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*. 25–32.