Non-collective Scalable Global Network Based on Local Communications

Marco Berghoff, Ivan Kondov
Steinbuch Centre for Computing
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: {marco.berghoff,ivan.kondov}@kit.edu

***Abstract*—To efficiently perform collective communications in current high-performance computing systems is a time-consuming task. With future exascale systems, this communication time will be increased further. However, global information is frequently required in various physical models. By exploiting domain knowledge of the model behaviors globally needed information can be distributed more efficiently, using only peer-to-peer communication which spread the information to all processes asynchronous during multiple communication steps. In this article, we introduce a multi-hop based Manhattan Street Network (MSN) for global information exchange and show the conditions under which a local neighbor exchange is sufficient for exchanging distributed information. Besides the MSN, in various models, global information is only needed in a spatially limited region inside the simulation domain. Therefore, a second network is introduced, the local exchange network, to exploit this spatial assumption.**

Both non-collective global exchange networks are implemented in the massively parallel NAStJA framework. Based on two models, a phase-field model for droplet simulations and the cellular Potts model for biological tissue simulations, we exemplary demonstrate the wide applicability of these networks. Scaling tests of the networks demonstrate a nearly ideal scaling behavior with an efficiency of over $90\,\%$. Theoretical prediction of the communication time on future exascale systems shows an enormous advantage of the presented exchange methods of $O(1)$ by exploiting the domain knowledge.

***Index Terms*—peer-to-peer communication, distributed memory, scalable parallel algorithms, massive-parallel performance, network protocol, stencil code, phase-field method**

I. Introduction

Large-scale computer simulation enables realistic 3D reproductions of various physical phenomena. To efficiently exploit current and future high-performance computing systems a high node-level performance and efficient communication schemes are desired. For various applications, global communication schemes are required for consistency calculations of the models. However, global collective communication is showing non-satisfying scaling behavior. With an increasing number of communication partners, this can become the limiting factor of the simulations. Detailed domain knowledge can help to identify more efficient methods for global information exchange. In this article, we demonstrate this with two applications, the phase-field method and the cellular Potts-model.

The phase-field method is used to investigate the microstructure evolution in many problems of computational material science [1], [2]. Many of these problems require calculations only in small interface regions of the simulated domain. As illustrated in Figure 1 a simulation of a water droplet only needs to be calculated the interface between water and the surrounding air. Here the interface is covered by blocks and only these are calculated. If the droplet moves during the simulation, the blocks are adapted. To be able to assign created blocks uniquely to a process, new blocks must be announced globally.

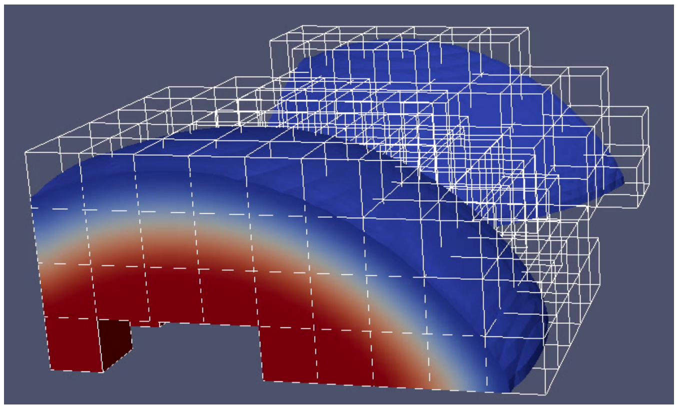
Fig. 1: Blocks denoted by white lines cover the interface between water (red) and air (blue). If the water droplet water condenses on the surface (bottom), the boundary surface and thus the calculating blocks move upwards.

The cellular Potts-model requires the surface and volume of distributed biological cells to calculate an energetically more favorable form of the individual cells. All processes that hold a part of a cell have to exchange the volume and the surface changes to each other. If it is not known in advance where exactly the parts of the cell are, this information must also be spread globally. Figure 2 shows cells in a simulation domain, divided in different blocks.

In the recently developed NAStJA framework [4] we employ a regular block-structured grid to decompose the simulated domain in small blocks, which are distributed over the MPI (Message Passing Interface) processes. Each of this blocks has a regular grid, where stencils can be used to calculate the partial differential equation with the explicit Euler scheme. NAStJA is able to process only the interface regions of the

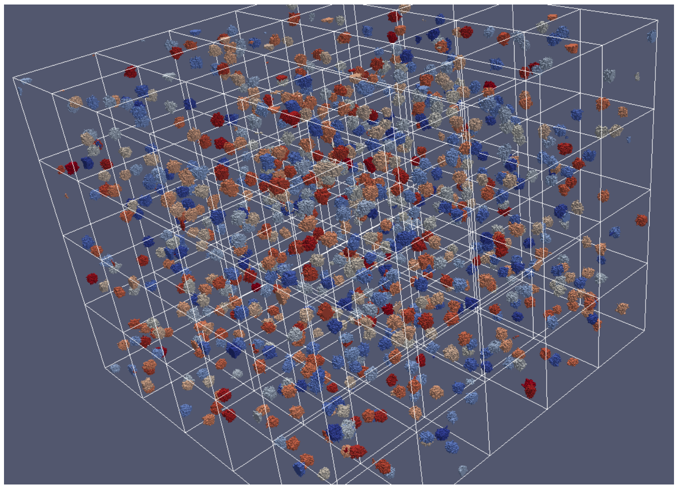
Fig. 2: More than 1000 biological cells placed in 100 blocks of a structured block grid (white lines). [3]

simulated domain by allocating and distributing just those blocks that contain a part of the interface, as needed by the droplet simulation. As the interface moves in the simulated domain throughout the simulation, the corresponding blocks are created or deleted. Communicating processes build a local neighborhood and act autonomously in this neighborhood. The locality of neighborhoods limits the number of connections per process and the local communication overhead and leads to high scalability. The NAStJA framework has being already employed for a phase-field method [5], [6] especially for droplets [7], a phase-field crystal model [8]–[10] and for the cellular Potts model, a cellular automaton for biological cells [11]. The framework can be extended with a wide range of algorithms that work on finite difference schema or other regular grid methods.

Nevertheless, besides from the communication within local neighborhoods, global information exchange is required. A use case within the NAStJA framework is the connection of previously disjoint local neighborhoods. This may be necessary while creating new blocks. With an increasing number of processes, global collective communications become a bottleneck and limit the scalability. It is therefore desirable that a global exchange, involving all MPI processes, of information can be performed without the use of collective communication functions. This requires certain prerequisites. Another way to avoid global collective communication is to use domain knowledge to restrain the width of communications. In the NAStJA framework, there are different communication schemes, which are all performed in each time step. Some of these overlap with the calculations to benefit from communication hiding.

For a clear distinction between the different communication schemes within the NAStJA framework, we first give a brief overview of all communication schemes.

(i) **Halo exchange** – In stencil code simulations, one or more halo layers usually extend the blocks. The halo holds a copy of the values of neighboring blocks. After each calculation step, the halo layers have to be updated. Therefore, a halo exchange is performed between each pair of adjacent sides of the blocks.

(ii) **Neighborhood exchange** – In the autonomously creating and deleting block modus, NAStJA must prevent overlaps, i.e., two different processes must not create a particular block. To ensure this, processes build local groups. Within a local group, the processes exchange information about their blocks. The topology of this network, i.e., the local groups, depends on the blocks contained on one process and the neighboring blocks. Each block has up to 125 neighbors so that a process can have a maximum of 125 neighboring processes per block.

(iii) **Global announcement network** – Two subdomains that are moving towards each other but do not have a joint local group must create a local group before they touch. In the autonomous system, however, it is not known precisely where the other subdomain is located or whether there is one at all. When a block is newly created, this is announced globally to all others. A receiver can then decide if the block is close enough and if he has to build a local group with it.

(iv) **Local exchange network** – A particular object, i.e., a connected subset of the domain, can be located in different neighboring blocks. To exchange information for these objects, all blocks that contain a fragment of the object must communicate with each other. A priori these are not known, so all blocks must communicate with each other.

We can distinguish these communication schemes between local exchanges and global exchanges. The local exchanges act between neighboring processes or blocks which have to communicate with each other. They can be implemented by peer-to-peer communications that constantly scale with $O(1)$.

Some information must be distributed globally because it is either required by each process or block or because the receivers are not known. In the further part of this article, we only consider the last two communication methods (iii) and (iv) which need a global exchange of information.

In the next section, we will show different approaches to achieve global exchange with only local communications. We outline the necessary prerequisites and present some use cases. In Section III we compare the introduced global exchange methods with global collective communication. Therefore we use scalability measurements of different test cases. We conclude with a summary and an outlook.

II. Methods

In this section, we present methods that are suitable for global exchange without the use of collective communication due to their prerequisites or specific domain knowledge. Firstly, the necessary prerequisites for each method are discussed.

A. Global Announcement Network

NAStJA implements a global announcement network to distribute information across all processes. This network includes all MPI processes independent of the blocks that each process holds.

In NAStJA, this network is mainly used to announce newly created blocks to all processes. Each process only knows neighboring blocks and only maintains a connection to the processes that host these adjacent blocks. Thus, the process that creates a new block cannot know all processes that have this new block in their neighborhood and have to communicate with the creator process of the new block. A new block does not have to be known directly, only if the absence of a block violates the consistency in the knowledge of the individual processes, i.e., a block is created a second time. Another error occurs if the simulation result differs from a reference full-domain simulation, so the new block must be known before the simulated structure (the interface) touches the border of the block, and a boundary condition is executed instead of the halo exchange. These inconsistencies occur at the earliest when the interface has moved through a block. Details can be found in Ref. [4]. With the used phase-field stencil, the interface can move by a maximum of one grid point per time step, so that the block size specifies how long the global announcement network may take to distribute the information to the entire network. Admittedly, the time is reduced by seven time steps, which are required for the communication setup after the new communication partner is known.

This global announcement network can be used in cases whenever consistency is not necessary at each time step, i.e., when some time steps may pass before the information is used. In particular, this network can be used to distribute system messages that occur only in one process, but to which all other processes must react. If a process notices that it does not make sense to continue the calculation because a previously defined state has been reached, it can report this to every process. Then a snapshot can be created before terminating the simulation. Such events could be premature exits when the simulated object reaches the boundary, and further simulation results are affected by the finite size of the domain.

The global announcement network works according to a multi-hop system and uses a store-and-forward distribution. The information is provided together with a time to live counter (TTL) which is initialized with the diameter of the network. Both the information combined with its TTL are sent to the peers. Then the TTL is decreased, and the information is forwarded to its peers in the next time step. In each time step, one hop is performed. The information can be processed directly, or if it is essential that all processes execute the ensuing action at the same time step, the peer waits until the TTL is 0 to perform the related action. Depending on the topology of the network a message may be received by some processes for a second time or may return. In such cases, the message is ignored and not further sent. Therefore a buffer containing all information with a valid (positive) TTL so that arriving information can be checked for existence in this buffer. The topology of the global announcement network is initially negligible.

TABLE I: Properties of some selected well-established network topologies.

network topology	nodes	degree	diameter
hypercube	2^n	n	n
butterfly	2^n	4	n
cube-connected cycles (order n)	$n2^n$	3	$2n + \lfloor n/2 \rfloor - 2$
regular nD torus (N edge size)	N^n	$2n$	$n \cdot \lfloor N/2 \rfloor$
Manhattan Street	N^n	n	$n \cdot \lfloor N/2 \rfloor + 1$

The network topology is chosen in a trade-off between the number of connections per process and the diameter, i.e., the minimum number of hops necessary to complete the communication. The optimal network topology has a minimum diameter at minimum degree, i.e., number of connections per process.

Let us map the global communication network onto a graph in which the vertices represent the processes and the edges the connections. In the past, some network topologies have been designed, discussed, analyzed and built. These include hypercubes [12], butterfly networks [12], cube-connected cycles [13], [14] and n-dimensional torus networks [15], [16]. Their properties are listed in Table I with the maximum possible number of vertices, degree, and diameter. If the number of processes deviates from the regular number of vertices, it is for the hypercube network especially difficult to build a modified variant. This is easier for butterfly and cube-connected cycles networks. However, there the number of nodes is limited by the fixed degree and given diameters. The n-dimensional torus network is flexible in these respects, i.e., the nodes can be increased by increasing the dimension for a given diameter. In case of undirected graphs, the received information is sent back directly or alternatively has to be filtered out to avoid sending them back. So, bidirectional edges make little sense. Manhattan Street Networks (MSN) [17]–[19] are similar to torus networks. They consist of directed edges alternating in positive and negative direction, as Figure 3 shows. Compared to the torus network, this halves the number of connections and so the degree. The diameter d remains almost the same, because in the n-dimensional MSN it is

$$d = \sum_{i=0}^{n-1} \frac{N_i}{2} + v, \tag{1}$$

where N_i is the number of vertices in the ith-dimension and $v = 1$ if and only if $N_i \bmod 4 = 0$ for all i otherwise $v = 0$.

To obtain the smallest possible diameter, the network should be as close to cubic as possible. For dimension n and K_0 vertices, the number of vertices per dimension is calculated inductively. Let $N_i := \lceil \sqrt[n-i]{K_i} \rceil$ be the number of vertices in the dimension i. The remaining vertices of the hyperrectangle of dimension $n - i$ are defined as $K_{i+1} := \lceil K_i / N_i \rceil$, for $i = 0, \dots, n-1$. In most cases, this results in a non-regular MSN. As an example, Figure 4 shows the two-dimensional

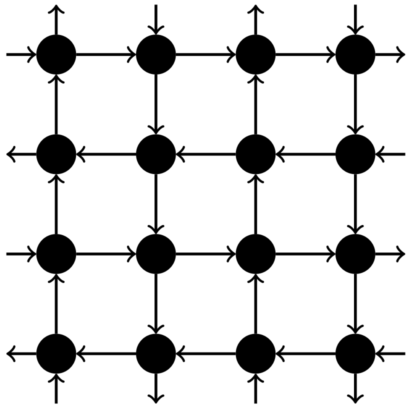

Fig. 3: Two-dimensional regular MSN of 4×4 vertices. The short arrows represent the connection with the opposite side.

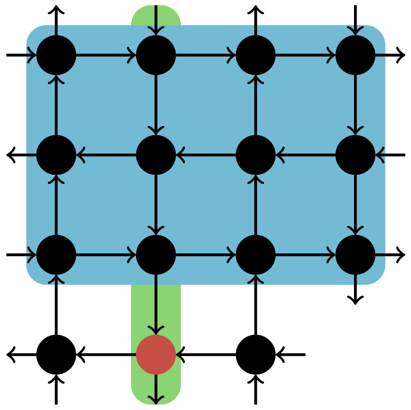

Fig. 4: Two-dimensional non-regular MSN with 15 vertices. The blue rectangle illustrates the regular (cuboid) part of the MSN. For the red vertex outside of the regular part there is a direct connection to the regular part, and from the regular part there is a direct connection to the red vertex, both marked with a green background.

MSN for 15 vertices. The resulting network consists of a cuboid MSN plus an incomplete hyperplane. Each vertex in the hyperplane can be reached with one hop from the cuboid, and the cuboid can be reached with one hop from each vertex of the hyperplane, so that the degree Eq. (1) increases by a maximum of one. In Table II the number of processor cores for a selection of supercomputer is given. For simulation using all processes of these supercomputers the diameters for MSNs of different dimensions are shown in Table III. The diameters that are needed to handle blocks with 35 grid points per dimension are highlighted. This shows that for moderate simulation sizes with relatively small blocks, three up to four connections are sufficient to build a global announcement network with an MSN. Even for extensive simulations with small blocks, nine connections are sufficient. If the blocks can be made a little larger, the diameter increased and such the degree will decrease. Note that there is an exponential progression so that for high dimensions, the number of hops changes only slightly.

TABLE II: Number of processor cores for selected supercomputers from the Top500 list, June 2018 [20].

cores	name	top 500 rank	
10 649 600	Sunway TaihuLight	2	most cores in the world
2 282 544	Summit	1	number one in the world
361 760	Piz Daint	6	number one in Europe
185 088	Hazel Hen	27	most cores in Germany
114 480	JUWELS Module 1	23	number one in Germany
22 960	ForHLR II	442	whole system at KIT
10 240	ForHLR II		productive mode

TABLE III: Diameter for given processes and different degrees, that is equal to the dimension of the MSN. The values in bold font highlight the dimension that is needed for diameter limited to 28, which corresponds to block sizes of 35.

number of	degree								
processes	2	3	4	5	6	7	8	9	10
10 240	102	33	**21**	16	15	14	13	13	13
22 960	152	43	**25**	19	17	15	15	14	14
114 486	339	74	37	**26**	21	19	18	17	17
185 088	431	86	42	29	**23**	20	19	18	17
361 760	602	107	50	33	**26**	22	20	19	19
2 282 544	1511	198	78	48	35	29	**25**	23	22
10 649 600	3264	331	115	64	45	36	31	**28**	26

B. Local Exchange Network

The MSN described above manages the global exchange of information using an arbitrary network which is spanned over the individual processes. This network does not take into account the topology of the blocks, i.e., the distribution of the computing domain. In addition, huge buffers, with a size of at least the message size per time step multiplied by diameter, are necessary for exchanges of significant size. For exchanging information that is restricted in a local but distributed region of the simulated domain, a further method of exchange is more reasonable.

In many applications, subregions of the simulation domain build small objects with a special meaning. These objects can be droplets, cells, grains, and so on. They occur in multi-droplet [7], [21], multi-cell [11] or multi-grain [22], [23] simulations. An entire object can be distributed over several subdomains (blocks). Furthermore, objects can move freely and change their size during the simulation, so it is unknown in which blocks a part of an individual object is located. This means that to calculate the volume of a particular object, the partial volumes of that object from each block is required. To distribute changing properties, e.g., the change in volume or surface area, all blocks holding a part of the object must be known, or all blocks have to exchange these properties with all other blocks. A priori these blocks are unknown and following the objects over blocks and hold an up-to-date list of all blocks which hold a part of the objects is communication intensive. On the other hand, for objects that only occur in a few blocks, many zero-length messages of the volume and surface area are exchanged unnecessarily.

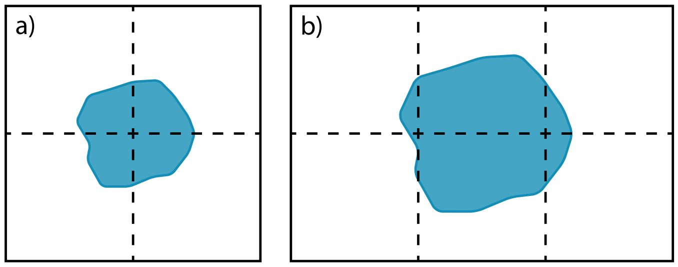

Fig. 5: (a) Valid and (b) invalid distribution of an object (blue) over blocks (dashed rectangles). The invalid object distribution overlaps three blocks in x-dimension. The size of the object is larger than the size of one box.

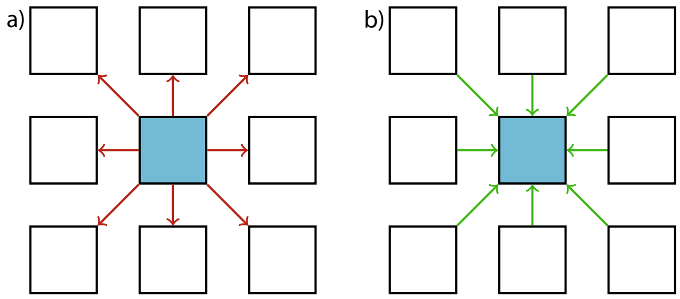

Fig. 6: Local neighbor exchange in two dimensions. The center block (blue) (a) sends to eight neighboring blocks and (b) receives from the same eight neighbors.

Moreover, a collective communication is not desirable for scalability reasons.

A possible option can be given under the prerequisite that only the direct neighboring blocks have to exchange information, this can be done in every time step without the usage of collective communication. If the spatial extension of objects does not exceed the extension of one block, this exchange method can be used. The distribution of the object across the block boundaries is not affected by this, as shown in Figure 5(a). An object occurs in a maximum of four blocks in two dimensions or eight blocks in three dimensions. For this purpose, individual objects can only be distributed to blocks if they do not leave a block on two opposite sides. However, this can be guaranteed if the size of the objects is limited. The objects can have a maximum size of the block extension, see Figure 5(b). Because the information is only sent to the neighbors, the middle part of the object receives all the information, but the outer parts will not receive it. Information from the right does not reach the left and vice versa, so the object information would no longer be consistent in all parts. The exchange is performed after each calculation step as shown in Figure 6. Initially, all the information, which can consist of different amounts of data per block, is stored in a message package. For example, for each object in a block two values for the volume and surface area plus an index of the object are stored in the message package. This message package is then sent to all direct neighbors, i.e., to eight neighbors in two dimensions and 26 in three dimensions. On the receiving side, the message packages of eight in two dimensions or 26 neighbors in three dimensions are received. The message packages are unpacked. If the data have been received from all neighbors, these can be processed. To ensure that the data is consistent, the local changes of the values are exchanged. From these, the absolute values are calculated, so the sum of the local changes is equal to the change of the entire object. Since objects move, they can enter blocks in which they were not before. In addition to the changes of volume and surface, their absolute values must be transferred, such that the newly entered blocks can calculate the current values of volume and surface from the changes. The amount of transferred data depends on the number of different objects and the number of different types of values. For typical simulation setups with a block size of 80^3 and an object size of 10^3, information of up to 512 different objects has to be exchanged. For an exchange of volume and surface, each with absolute value and a value for the change, as well as four values for a center of mass calculation, these are summed up to 13 values including keys and thus in total almost 7000 values per exchange.

C. *Data-aware Global Multi-hop Network*

The most recently introduced method allows to distribute data locally to neighbors so that these data can be considered as global for all partners specified under the given requirements, i.e., all other partners do not need this data. The former method (see Section II-A) performs a global exchange independent of the structure of the simulation domain. A combination of both methods is the extension of the local neighbor exchange with an additional TTL counter. This method is useful for events that are fast spreading throughout the whole domain but that are not time-critical. The only difference to the local exchange network is the appending of the TTL counter to the exchanged data which slightly increases the size of the exchanged message package. The number and partner of all connections stay the same. In a second step, these stored data are forwarded to the neighboring blocks of the first receivers. In this way, information can be propagated through the whole domain, or even only through parts of it. For this purpose, the topology of the simulation domain is used and not an arbitrary topology as it is used in the method from Section II-A. This allows restricting the exchange to local areas up to n blocks away from the triggering block.

As an example, refer to Figure 7, in the shown block chain two hops are needed to get the information from A to C, for reasons of clarity only one dimension is illustrated. An arbitrary object is located in A and B. The properties are distributed to the neighbors, so C also holds these properties received from B. If a property is changed starting from A, this property is transferred to B via the neighbor exchange. The object behaves consistently since A and B hold the same properties for this object. However, C has not yet heard of the change, and that is not a problem as long as the object is not located in C. Through the multi-hop exchange, C will be

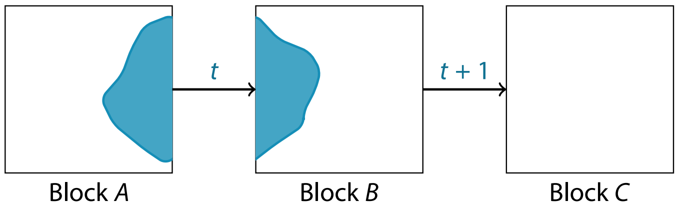

Fig. 7: Multi-hop exchange in one dimension. Block A generates a message and send this to block B in the time step t. In the next time step $t+1$, block B sends this message to block C.

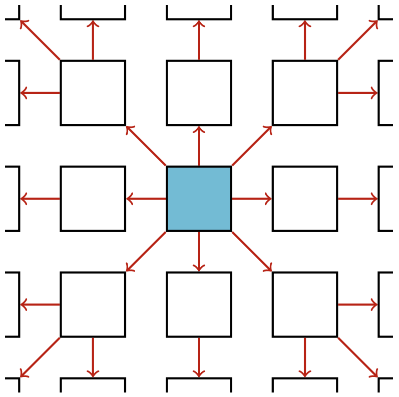

Fig. 8: Distribution of information in the multi-hop local exchange, for the sake of clarity two steps in 2D. Incoming messages from sides only have to be forwarded to the opposite side. For edges in 3D or corners in 2D, these must be forwarded to three blocks on the opposite side. For corners in 3D, forwarding to seven neighbors is necessary. This results in $6\cdot 1+12\cdot 3+8\cdot 7=97$ forwardings in 3D and $4\cdot 1+4\cdot 3=16$ in 2D.

informed about the change one time step later, if the object will move into C after that, it is also consistent here. This procedure is useful and necessary if, for example, degenerate objects that occur in both A and C must be deleted. So every block in the extended neighborhood will be informed in the next n time steps, where n is equal to the distance to the original creator block of that message.

D. Theoretical Evaluation of the Different Network Topologies

A significant disadvantage of all-to-all communications is that the size of the message size must be known in advance so that it must be generously increased to the largest message size that is possible. Our presented networks can all work with MPI_Probe and thus can handle the receiving of data of different sizes. Furthermore, it is sufficient to send a tiny message if no data has to be exchanged, the size of this is neglected in the calculation below. The number of messages can, therefore, vary significantly. Nevertheless, we want to give estimates for the message rate (M), and the amount of exchanged data (E) for the different networks.

Let m be the message size, N the number of nodes and B the number of blocks in the simulation. For non-dynamically adaptive simulations, usually $B=N$ is selected. For the MSN, the diameter results in $d\approx\sqrt[n]{N}$, where n is the dimension of the MSN. For the local exchange networks, the diameter is the largest number of blocks in one dimension, $d\approx\sqrt[3]{B}$.

(i) For all-to-all communications, it is $M=N\cdot N$ and $E=N\cdot N\cdot m$.

(ii) An MSN with one message from one process in one time step results in $M=n\cdot N$ and $E=n\cdot m$, this exchange then occurs in the d subsequent time steps. For one message per process in a time step $M=n\cdot N$ and $E=n\cdot N\cdot m$, this exchange also takes place in the d following time steps. If each process generates a message of size m in each time step, the result is $M=n\cdot N$ and $E=n\cdot N\cdot m\cdot d$. Here, the size of the message is $m\cdot d$, since the messages from the d previous time steps are distributed further. It should be noted that $m\cdot d$ is an upper limit, old messages that have already been sent or forwarded are not sent again. However, as described above, a message that has already been sent can be received again, for example when exactly one four-node-cycle has ended in the MSN, but they are not sent on the journey again.

(iii) For the local single-hop exchange network, $M=26\cdot B$ and $E=26\cdot m$ for one message from one block and $E=26\cdot B\cdot m$ when every block sends a message.

(iv) In the multi-hop method, $M=26\cdot B$ remains to take older messages into account. $E=26\cdot m\cdot d$ is the upper limit. Here, the sending back of messages can be waived, too. In addition, a message does not have to be sent to all neighbors if it is known that another neighbor is sending it there, which reduces the average number of connections per message to less than four, see Figure 8. This implied $E\leq 4\cdot m\cdot d$.

While all-to-all (i) has a scaling of $O(N^2)$ in relation to the number of nodes, the other networks strongly depend on the number of generated messages. The MSN (ii) varies from $O(N)$ to $O(N^{1+\frac{1}{n}})$, depending on the dimension n. The single-hop exchange (iii) is linear $O(1)$, for the multi-hop exchange (iv) it varies from $O(1)$ to $O(N^{\frac{1}{3}})$ depending on the number of generated messages.

III. Results

In this section, we present measurements and results of the two different communication methods for global data exchange in the NAStJA framework.

For the scaling test we perform simulations on the ForHLR II located at Karlsruhe Institute of Technology. The ForHLR II consists of 1152 20-way Intel Xeon compute nodes. Each of these nodes has two deca-core Intel Xeon processors E5-2660 v3 with the Haswell architecture which run at a base clock rate of 2.6 GHz and have 10×256 KB of level 2 cache and 25 MB shared level 3 cache. Each node provides 64 GB of main memory and an FDR adapter to connect to the InfiniBand

4X EDR interconnect. In total 512 nodes can be used that are connected by a quasi-fat tree topology with a bandwidth ratio of 10 to 11 between the switches and leaf switches. NAStJA used the implementation of OpenMPI.

All scaling tests were performed in the setting of a weak scaling. For the basis value (t_1) we use the runtime on one full node with 20 cores. The side length of 100 for a cubic block is chosen, and one block per process is used. During the tests, we omit all disk I/O routines. The parallel efficiency η is defined by

$$\eta = \frac{t_1}{t_p}, \tag{2}$$

where t_p is the parallel run time with p nodes.

The test size increases by power of two up to 512 nodes, corresponding to 10 240 processes. We use for all tests an artificial constant workload of 65 ms per process and use a halo exchange of six sides. The simulations run 1000 time steps, and an average is calculated.

A. Global Announcement Network

Figure 9 shows measurements of the time and efficiency for a constant workload and a global exchange via the five-dimensional MSN compared to collective communication. For global exchange by collective MPI operations, the size of the exchange array must be specified in advance. The actual required size is not known before and has to be estimated generously. So the global operations were measured for 400 B which corresponds to 100 integer values and 8 kB and 800 kB. As expected, the efficiency for many processes and large exchange arrays drops significantly, but even the exchange of hundred values has an efficiency of only 75 % on 10 240 processes. In contrast, the exchange using the MSN has an efficiency of over 90 % and shows excellent scalability.

B. Local Exchange Networks

The both networks described in Section II-B and II-C have the same network topology. The only difference is the amount of data sent. The data-aware global multi-hop network (II-C) uses a store-and-forward mechanism to forward received data to the next neighbors. Measurements in Figure 10 show the time and efficiency for different data sizes. It was measured again with a constant workload of 65 ms per process and a halo exchange of six sides. The data sizes used are 400 B, 8 kB and 800 kB. The efficiency is over 95 % on 256 nodes with 5 120 processes for all measured data sizes. At least for the sizes shown up to 800 kB this exchange is independent of the size of the transmitted data.

IV. Conclusion and Outlock

Efficient scalability is of enormous importance for software on HPC systems. Global information exchange mostly uses collective communications, which are known to scale unfavorably, especially for large amounts of data and many communicating processes. This behavior could also be reproduced in this work. However, with detailed knowledge of the model system, it is possible to identify prerequisites that enable

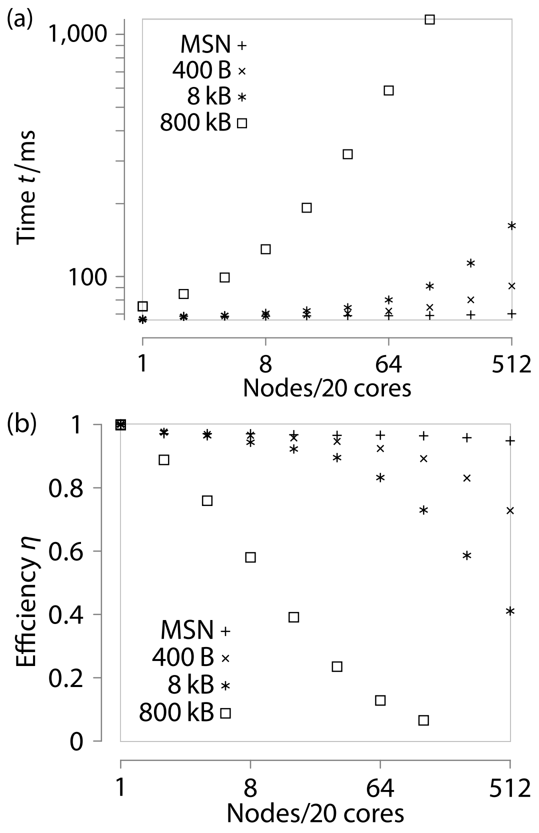

Fig. 9: Scaling at constant workload (65 ms) per process with halo exchange of six sides and a global exchange. Comparison of the MSN to global collective communication with 400 B, 8 kB and 800 kB. (a) Average time per time step and (b) efficiency.

distributing this information globally without collective communications. It has been shown that it is possible to use multi-hop networks to spread time uncritical information through the entire simulation domain using only local peer-to-peer communications. The presented scaling results show that in simulations with more than 10 000 processes it is still possible to achieve an efficiency of more than 90 %. The MSN used here has a larger degree than other network topologies, but the MSN is easy to understand and to implement for non-regular numbers of vertices. With the excellent scalability measured as shown in the results, it is also completely sufficient. The diameter can be controlled via the dimension of the MSN, so that a suitable n-dimensional MSN can be found depending on the specific application.

For other information, which is distributed over blocks but limited locally, local exchanges are sufficient, and standard MPI collective communications can be dispensed. In such cases, an efficiency of more than 95 % is possible if the size of the calculated objects is limited to a sufficiently small size. The method from Section II-B can also be extended in the way that the local exchange goes to the nearest neighbors and their neighbors. This increases the number of exchanges

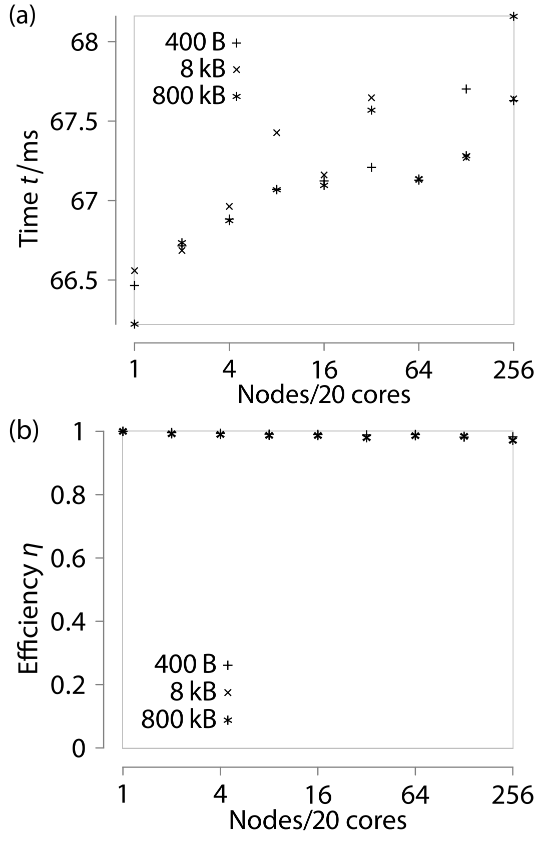

Fig. 10: Scaling at constant workload of 65 ms per process with halo exchange of six sides and an additional exchange. (a) Average time per time step and (b) efficiency.

to 24 in two dimensions and 124 in three dimensions. The absolute time is expected to get slightly worse, but this time stays constant with the number of processes. This would allow treating objects that have an extension of twice the length of the block edge in each dimension. Whenever it is worthwhile to use small blocks, which could fit into the cache and cause more exchange instead, is to be considered on a case-by-case basis. Furthermore, the communication methods presented here show a constant scaling behavior, so that a good scaling behavior can also be projected onto future exascale systems.

Acknowledgment

This work was performed on the computational resource ForHLR II funded by the Ministry of Science, Research and the Arts Baden-Württemberg and DFG ("Deutsche Forschungsgemeinschaft").

We thank J. Rosenbauer and J. Hötzer for the fruitful discussions.

References

[1] M. Bauer, J. Hötzer, M. Jainta, P. Steinmetz, M. Berghoff, F. Schornbaum, C. Godenschwager, H. Köstler, B. Nestler, and U. Rüde, "Massively parallel phase-field simulations for ternary eutectic directional solidification," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* ACM, 2015, p. 8.

[2] J. Hötzer, M. Jainta, P. Steinmetz, B. Nestler, A. Dennstedt, A. Genau, M. Bauer, H. Köstler, and U. Rüde, "Large scale phase-field simulations of directional ternary eutectic solidification," *Acta Materialia*, vol. 93, no. 0, pp. 194 – 204, 2015.

[3] J. Rosenbauer, personal communication.

[4] M. Berghoff, I. Kondov, and J. Hötzer, "Massively parallel stencil code solver with autonomous adaptive block distribution," *IEEE Transactions on Parallel and Distributed Systems*, 2018. [Online]. Available: http://doi.acm.org/10.1109/TPDS.2018.2819672

[5] B. Nestler, H. Garcke, and B. Stinner, "Multicomponent alloy solidification: phase-field modeling and simulations," *Physical Review E*, vol. 71, no. 4, p. 041609, 2005.

[6] M. Berghoff, M. Selzer, and B. Nestler, "Phase-field simulations at the atomic scale in comparison to molecular dynamics," *The Scientific World Journal*, vol. 2013, 2013.

[7] M. Ben Said, M. Selzer, B. Nestler, D. Braun, C. Greiner, and H. Garcke, "A phase-field approach for wetting phenomena of multiphase droplets on solid surfaces," *Langmuir*, vol. 30, no. 14, pp. 4033–4039, 2014.

[8] K. Elder, N. Provatas, J. Berry, P. Stefanovic, and M. Grant, "Phase-field crystal modeling and classical density functional theory of freezing," *Physical Review B*, vol. 75, no. 6, p. 064107, 2007.

[9] M. Berghoff and B. Nestler, "Phase field crystal modeling of ternary solidification microstructures," *Computational Condensed Matter*, vol. 4, pp. 46–58, 2015.

[10] M. Guerdane and M. Berghoff, "Crystal-melt interface mobility in bcc fe: Linking molecular dynamics to phase-field and phase-field crystal modeling," *Physical Review B*, vol. 97, no. 14, p. 144105, 2018.

[11] F. Graner and J. A. Glazier, "Simulation of biological cell sorting using a two-dimensional extended Potts model," *Physical Review Letters*, vol. 69, no. 13, p. 2013, 1992.

[12] Y. Solihin, *Fundamentals of Parallel Computer Architecture.* Solihin Publishing and Consulting LLC, 2009.

[13] F. P. Preparata and J. Vuillemin, "The cube-connected cycles: A versatile network for parallel computation," *Commun. ACM*, vol. 24, no. 5, pp. 300–309, May 1981. [Online]. Available: http://doi.acm.org/10.1145/358645.358660

[14] I. Friš, I. Havel, and P. Liebl, "The diameter of the cube-connected cycles," *Information processing letters*, vol. 61, no. 3, pp. 157–160, 1997.

[15] D. Banerjee, B. Mukherjee, and S. Ramamurthy, "The multidimensional torus: analysis of average hop distance and application as a multihop lightwave network," in *Communications, 1994. ICC'94, SUPERCOMM/ICC'94, Conference Record,'Serving Humanity Through Communications.'IEEE International Conference on.* IEEE, 1994, pp. 1675–1680.

[16] P. Fragopoulou and S. G. Akl, "Efficient algorithms for global data communication on the multidimensional torus network," in *Parallel Processing Symposium, 1995. Proceedings., 9th International.* IEEE, 1995, pp. 324–330.

[17] B. Khasnabish, "Topological properties of Manhattan street networks," *Electronics Letters*, vol. 25, no. 20, pp. 1388–1389, 1989.

[18] T.-Y. Chung and D. P. Agrawal, "Design and analysis of multidimensional Manhattan Street Networks," *IEEE transactions on communications*, vol. 41, no. 2, pp. 295–298, 1993.

[19] F. Comellas, C. Dalfó, and M. A. Fiol, "Multidimensional Manhattan street networks," *SIAM Journal on Discrete Mathematics*, vol. 22, no. 4, pp. 1428–1447, 2008.

[20] TOP500.org, "Top500 List - June 2018," 2018. [Online]. Available: https://www.top500.org/list/2018/06/

[21] F. Weyer, M. Ben Said, J. Hötzer, M. Berghoff, L. Dreesen, B. Nestler, and N. Vandewalle, "Compound droplets on fibers," *Langmuir*, vol. 31, no. 28, pp. 7799–7805, 2015.

[22] L. Gránásy, T. Pusztai, and J. A. Warren, "Modelling polycrystalline solidification using phase field theory," *Journal of Physics: Condensed Matter*, vol. 16, no. 41, p. R1205, 2004.

[23] N. Moelans, F. Wendler, and B. Nestler, "Comparative study of two phase-field models for grain growth," *Computational Materials Science*, vol. 46, no. 2, pp. 479–490, 2009.