

# Reducing communication in algebraic multigrid with multi-step node aware communication

The International Journal of High Performance Computing Applications 2020, Vol. 34(5) 547–561  
© The Author(s) 2020  
Article reuse guidelines:  
sagepub.com/journals-permissions  
DOI: 10.1177/1094342020925535  
journals.sagepub.com/home/hpc



Amanda Bienz<sup>1</sup>, William D Gropp and Luke N Olson

## Abstract

Algebraic multigrid (AMG) is often viewed as a scalable  $O(n)$  solver for sparse linear systems. Yet, AMG lacks parallel scalability due to increasingly large costs associated with communication, both in the initial construction of a multigrid hierarchy and in the iterative solve phase. This work introduces a parallel implementation of AMG that reduces the cost of communication, yielding improved parallel scalability. It is common in Message Passing Interface (MPI), particularly in the MPI-everywhere approach, to arrange inter-process communication, so that communication is transported regardless of the location of the send and receive processes. Performance tests show notable differences in the cost of intra- and internode communication, motivating a restructuring of communication. In this case, the communication schedule takes advantage of the less costly intra-node communication, reducing both the number and the size of internode messages. Node-centric communication extends to the range of components in both the setup and solve phase of AMG, yielding an increase in the weak and strong scaling of the entire method.

## Keywords

Parallel, multigrid, algebraic multigrid, sparse matrix

## 1. Introduction

Algebraic multigrid (AMG) (Brandt et al., 1984; McCormick and Ruge, 1982; Ruge and Stüben, 1987) is an iterative solver for sparse linear systems, such as those arising from discretized partial differential equations. AMG targets linear cost in the number of unknowns and is dominated in cost by sparse matrix operations such as the sparse matrix-matrix multiplication (SpGEMM) and sparse matrix-vector multiplication (SpMV). As state-of-the-art supercomputers are continuously increasing in performance capabilities, there is pressure on numerical methods to fully exploit the potential of these machines. Due to large costs associated with parallel communication, for example, on the coarse levels (Baker et al., 2011), AMG lacks parallel scalability. This article explores a method of altering the parallel implementation of communication throughout AMG to improve both performance and parallel scaling.

Standard parallel AMG typically exhibits strong scaling to 5 or 10000 degrees of freedom per core before communication costs outweigh local computation. Further extending the core count yields an increase in total solve time due to dominant communication costs. Figure 1 shows the time required to solve a Laplacian system, created with MFEM (Lawrence Livermore National Laboratory (LLNL), 2010),

and described in detail in example 2.1. The timings are partitioned into local computation and inter-process communication costs. As the processor count reaches the strong scaling limit, communication becomes increasingly dominant. The problem scales to 512 processes, after which communication costs outweigh any reductions in local computation.

Most methods for reducing communication costs in AMG focus on a redesign of the method or on the underlying sparse matrix operations. Aggressive coarsening, for example, reduces the dimensions of coarse levels at a faster rate, yielding reduced density and communication requirements (Sterck et al., 2005, 2008; Yang, 2010). Similarly, the smoothed aggregation solver allows large aggregates, coarsening a larger number of fine points into a single coarse point (Treister and Yavneh, 2015; Tuminaro and Tong, 2000). Small nonzero entries resulting from fill-in on coarse levels may be systematically removed, adding sparsity into coarse-grid operators (Bienz et al., 2016;

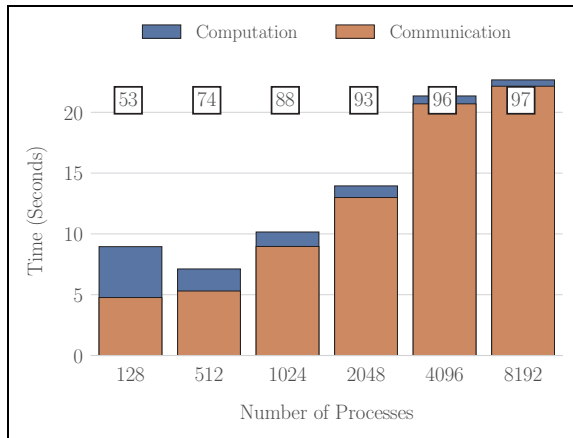
---

Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA

### Corresponding author:

Amanda Bienz, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA.

Email: bienz2@illinois.edu



**Figure 1.** The total time required to solve a three-dimensional Laplacian system with 1884545 degrees of freedom described in example 2.1. The percentage of total time spent in communication is listed at the top of the bars.

Falgout and Schroder, 2014; Treister and Yavneh, 2015). Furthermore, matrix ordering and graph partitioning yield reduced communication costs throughout sparse matrix operations (Hendrickson and Kolda, 2000; Pinar and Aykanat, 2004; Vastenhouw and Bisseling, 2005; Çatalyürek and Aykanat, 1999, 2010), and coarse-level repartitioning has potential to reduce the cost of the solver (Adams et al., 2004). Likewise, coarse-level redistribution and duplication of the coarsest level solves yield large reductions in communication time. MPI + X approaches have shown improvements to communication requirements by reducing the number of processes active in communication as well as overlapping communication and computation (AIONazi et al., 2017; Baker et al., 2011). Similarly, asynchrony can be introduced through additive approaches, allowing for multiple levels to be operated on at once, yielding reduced load imbalance (AIONazi et al., 2017; Vassilevski and Yang, 2014). The approach presented in this article augments these approaches, reducing off-node message counts and sizes through aggregation of data.

Topology-aware methods and message agglomeration are commonly used to reduce communication costs in MPI applications. Topology-aware task mapping minimizes message hop counts, reducing the cost associated with communication (Bhatele and Kale, 2008; Bhatele et al., 2012). Message agglomeration is commonly used to reduce the cost of communication, for example, in MPI collectives (Karonis et al., 2000; Kielmann et al., 1999; Sack and Gropp, 2012; Solomonik et al., 2011). The Tram library (Wesolowski et al., 2014) explores agglomeration of point-to-point messages, by streamlining messages between neighboring processes. Neighborhood collective operations were added to MPI 3, allowing for topology-aware communication patterns (Hoefler and Traff, 2009). However, current implementations of neighborhood collectives do not address topology, consisting only of standard nonblocking sends and receive operations. Recent work has

combined messages in neighborhood collectives based on common neighbors (Mirsadeghi et al., 2017).

This article presents, analyzes, and evaluates a method for reducing communication costs in both the setup and solve phases of parallel AMG through agglomeration of messages among nodes. The novel contributions of this article include applying an existing communication technique to the communication of sparse matrices, introducing a second method for agglomeration of messages, analyzing the performance of the various sparse communication strategies, and automatically selecting a communication scheme based on a performance model. The article is outlined as follows. Section 2 covers AMG and common parallel implementations. Section 3 focuses on the node-aware communication algorithm, with background and the original node-aware communication approach described in section 3.1, while a new variation is presented in section 3.2. Section 3.3 presents performance models that differentiate between communication strategies. Section 4 covers numerical experiments in support of the approach, and section 5 contains concluding remarks and future directions.

## 2. Background

Throughout this article, AMG methods are analyzed with regard to the three-dimensional Laplacian in example 2.1. This system is representative of the types of problems that are often solved with AMG.

*Example 2.1.* Let the system  $Ax = b$  result from a finite element discretization of the Laplace problem  $-u = 1$ , created with MFEM (LLNL, 2010). The linear system is created with MFEM's escher-p3 mesh, a three-dimensional mesh consisting of unstructured elements with structured refinement. Furthermore, this system consists of 1884545 degrees of freedom and 27870337 nonzeros, unless otherwise specified. The associated Ruge–Stüben (Ruge and Stüben, 1987) hierarchies are created with Hybrid Modified Independent Set (HMIS) (Sterck et al., 2005; Yang, 2010) coarsening and extended + i interpolation (Sterck et al., 2008), while the smoothed aggregation solver (Tuminaro and Tong, 2000) forms aggregates based on a distance-2 maximal independent set (MIS-2) of the graph. Both classical and smoothed aggregation hierarchies use a strength tolerance of 0.25. Classical interpolation operators are truncated, as described in (Sterck et al., 2008), with a threshold of 0.3. All tests are performed with RAPtor (Bienz and Olson, 2017), an AMG codebase containing both Ruge–Stüben and smoothed aggregation solvers. The Ruge–Stüben hierarchy created with RAPtor is identical to that created with the same parameters in HYPRE (Lawrence Livermore National Laboratory (LLNL), 2008), allowing for comparisons. All timings are performed on 8192 processes of Blue Waters (Bode et al., 2013; NCSA, 2012), a Cray XK/XE supercomputer at the National Center for Supercomputing Applications, unless stated otherwise.

AMG consists of forming a hierarchy of successively coarser levels, followed by an iterative solve phase.

**Algorithm 1.** AMG setup: setup.

```

Input:  $A$  {sparse system matrix}
max_coarse {Maximum dimension of coarsest matrix}
Aggregation {Whether to use Smoothed Aggregation or Ruge–Stüben}

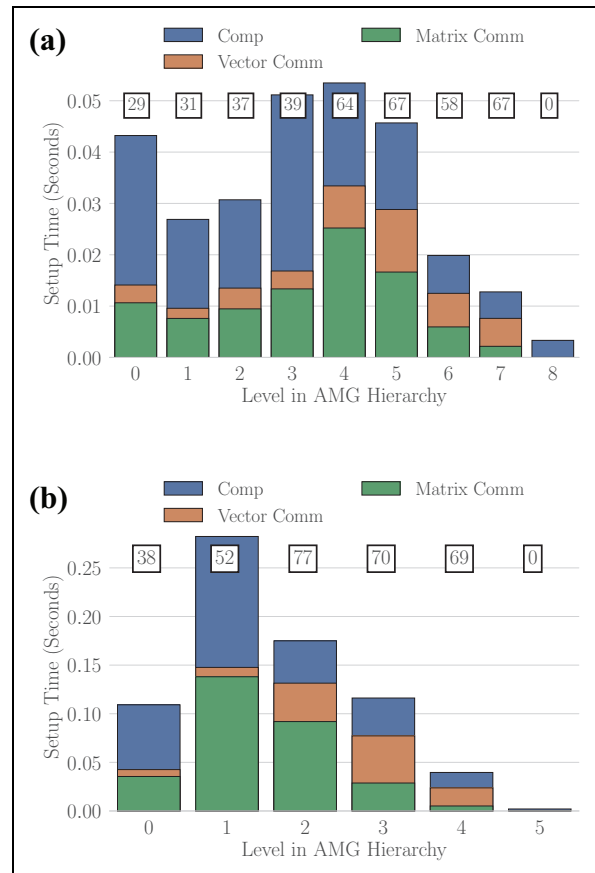
Output:  $A_0, A_1, \dots, A_\ell$  {hierarchy of sparse operators}
 $P_0, P_1, \dots, P_{\ell-1}$ 

 $A_0 \leftarrow A$ 
 $\ell \leftarrow 0$ 
while  $|A_\ell| > \text{max\_coarse}$ 
   $S_\ell \leftarrow \text{strength}(A_\ell)$  {strength-of-connection}
  if Aggregation
     $Agg_\ell \leftarrow \text{splitting}(S_\ell)$  {partition nodes}
     $P_\ell \leftarrow \text{interpolation}(Agg_\ell)$  {form interpolation}
  else
     $C_\ell, F_\ell \leftarrow \text{splitting}(S_\ell)$  {partition nodes}
     $P_\ell \leftarrow \text{interpolation}(C_\ell, F_\ell)$  {form interpolation}
   $A_{\ell+1} \leftarrow P_\ell^T \cdot A_\ell \cdot P_\ell$  {coarse operator, Galerkin product}
   $\ell \leftarrow \ell + 1$ 
  
```

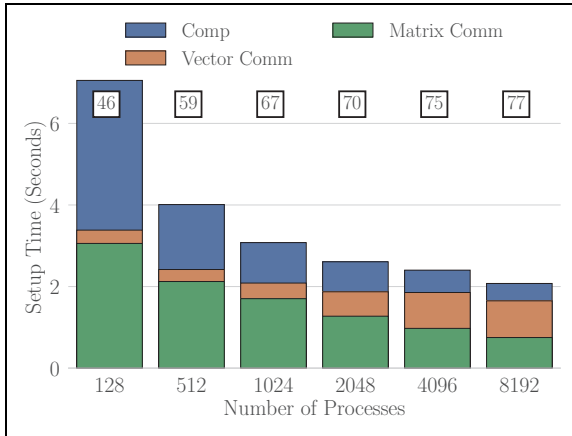
Common algorithms for constructing an AMG hierarchy include the Ruge–Stüben solver (Ruge and Stüben, 1987) and the smoothed aggregation solver (Tuminaro and Tong, 2000). This setup phase, described in Algorithm 1, consists of four methods: strength, splitting, interpolation, and  $P^T \cdot A \cdot P$ , regardless of the solver used. First, the strength function determines nodes that are strongly connected to one another. The resulting strength-of-connection matrix is then partitioned in splitting to determine the nodes influencing each coarse degree of freedom. The interpolation function uses the node partition to form a transfer operator, which projects data between the fine and coarse nodes. Finally, the coarse-grid operator is formed with a Galerkin product,  $P^T \cdot A \cdot P$ .

The underlying algorithms for splitting and interpolation are solver dependent. The Ruge–Stüben solver partitions nodes into coarse (C) and fine (F) points. The interpolation operator projects C-points directly between fine and coarse levels, while F-points influence neighboring coarse nodes. Alternatively, the smoothed aggregation solver partitions the nodes into groups of aggregates, each corresponding to a single coarse degree of freedom. The transfer operator is initially created as a point-wise constant, with a single column holding each aggregate. Near-nullspace candidates are then fit to the operator, and finally, the resulting matrix is smoothed.

The construction of a parallel AMG hierarchy requires both local computation and point-to-point communication, specifically communication of both vectors and sparse matrices. Figure 2 partitions the per level cost of constructing a hierarchy for example 2.1 into local computation, vector communication, and sparse matrix communication. Irrespective of which setup algorithm is used, the cost of hierarchy construction is split into local computation and



**Figure 2.** The setup phase cost for various AMG hierarchies for the system from example 2.1. The total cost is partitioned by level, with the finest level labeled 0, and further split into communication and local computation. The percentage of total time spent communicating is listed at the top of the bars: (a) Ruge–Stüben, (b) smoothed aggregation. AMG: algebraic multigrid.



**Figure 3.** The cost of creating a Ruge–Stüben hierarchy for example 2.1 on various core counts. The percentage of total time spent communicating is listed at the top of the bars.

MPI communication, with communication dominating coarse-level setup cost.

Point-to-point communication dominates the total cost of the setup phase, particularly when a large number of processes are active in construction of the hierarchy. Figure 3 displays the cost of forming a Ruge–Stüben hierarchy for example 2.1, strongly scaled across a variety of core counts. As the number of processes is increased, a larger percentage of time is associated with point-to-point communication.

After the hierarchy is constructed, the solve phase iterates over all levels until convergence. This phase, described in Algorithm 2, consists of `relax`, which relaxes error with a smoother such as Jacobi or Gauss–Seidel, calculating the residual, and restricting this residual to a coarser level where this process is repeated until error can be solved for directly. Finally, error from the coarser level is interpolated up the hierarchy, added to the current solution, and again smoothed with a relaxation method.

Figure 4 displays the cost of iteratively solving the Ruge–Stüben and smoothed aggregation hierarchies for example 2.1. The solve phase costs are partitioned into local computation and MPI vector communication,

associating the large increase in cost on coarse levels with point-to-point communication.

MPI communication dominates the cost of the AMG solve phase, particularly on coarse levels and at large scales, as analyzed in Bienz et al. (2016). Figure 5 shows the full cost of the iterative solve phase of AMG at various scales. As the number of processes increases, the percentage of cost due to communication also increases, even as the problem size stays constant.

**2.1. Parallel matrix operations.** Parallel vector and sparse matrix communication dominates the cost of AMG, particularly at large scales. This point-to-point communication is required for parallel sparse matrix operations in methods of both the setup and solve phases.

Assuming a row-wise partition of a linear system, as displayed in Figure 6, each process holds a contiguous subset of the rows of the matrix, along with corresponding vector values. The local rows of the matrix are further split into an on-process block, associated with local vector values, as well as off-process columns, which correspond to vector entries stored on other processes. As a result, matrix operations such as SpMV communication require communication of vector values corresponding to nonzero off-process columns. Hence, vector communication consists of each process sending vector values to any process with corresponding nonzero columns. This communication pattern is initialized during the construction of the matrix.

Similarly, matrix communication depends on the nonzero off-process columns. Figure 7 shows two matrices,  $A$  and  $B$ , partitioned row-wise across four processes, with the local rows again partitioned into on- and off-process columns. Matrix operations such as SpGEMM multiplication require communication of the rows of  $B$  that correspond to nonzero off-process columns of  $A$ . In essence, matrix communication retains the same communication pattern as that of vectors, but requiring entire rows of the matrix rather than single values.

Communication (Gropp et al., 1996) dominates the total time (both setup and solve), as shown in Figure 1. More

#### Algorithm 2. AMG solve: solve

**Input:**  $A_\ell, P_\ell, x_\ell, b_\ell$

{system, interpolation, initial solution, right-hand side at level  $\ell$ }

**Output:**  $x_\ell$

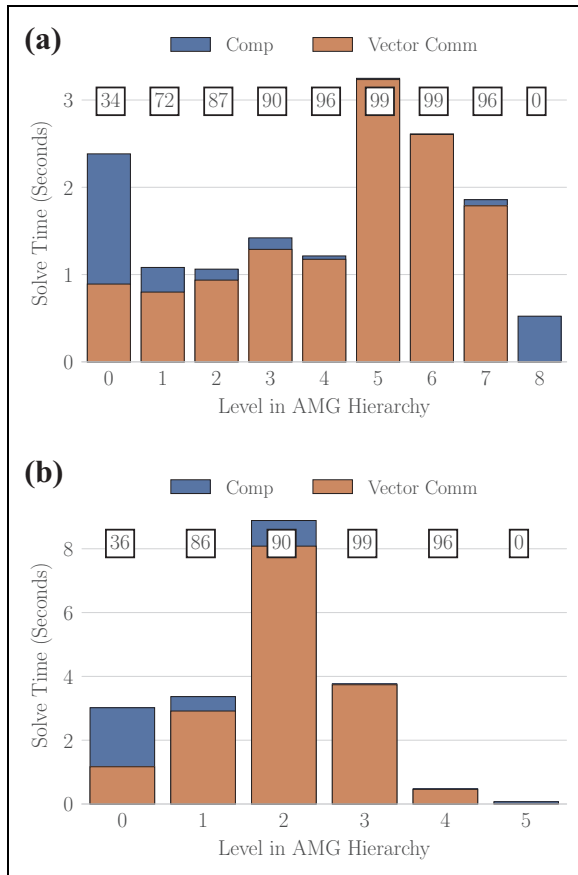
{updated solution vector}

**if**  $\ell = N$

`solve`  $A_\ell x_\ell = b_\ell$

**else**

<code>relax</code> $A_\ell x_\ell = b_\ell$	{pre-relaxation}
$r_\ell \leftarrow b_\ell - A_\ell \cdot x_\ell$	{calculate residual}
$r_{\ell+1} \leftarrow P_\ell^T \cdot r_\ell$	{restrict residual}
$e_{\ell+1} \leftarrow \text{solve}(A_{\ell+1}, P_{\ell+1}, 0, r_{\ell+1})$	{coarse-grid solve}
$e_\ell \leftarrow P_\ell \cdot e_{\ell+1}$	{interpolate error}
$x_\ell \leftarrow x_\ell + e_\ell$	{update solution}
<code>relax</code> $A_\ell x_\ell = b_\ell$	{post-relaxation}



**Figure 4.** The solve phase cost for various AMG hierarchies for the system from example 2.1. The total cost is partitioned by level, with the finest level displayed as 0, and further split into communication and local computation. The percentage of total time spent communicating is listed at the top of the bars: (a) Ruge–Stüben, (b) smoothed aggregation. AMG: algebraic multigrid.

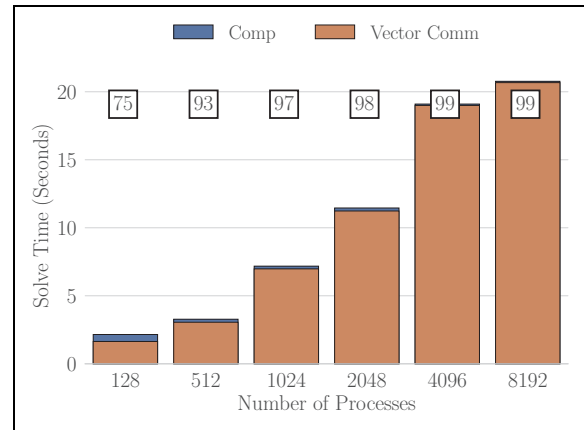
precisely, the coarse-level matrix operations are dominated by *point-to-point* communication of vectors and sparse matrices, as displayed in Figures 2 to 5. The cost associated with communication increases on coarse levels.

### 3. Node-aware communication

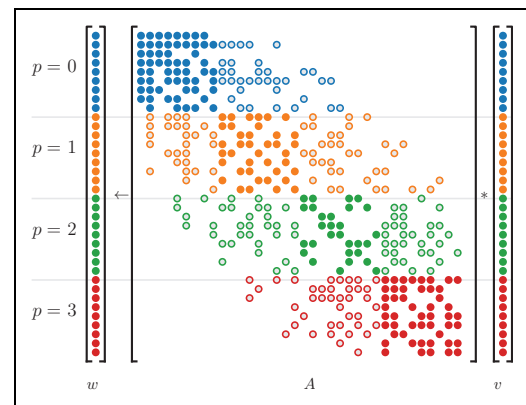
The cost associated with standard point-to-point communication throughout AMG can be reduced through the use of node-aware communication, particularly when a large number of messages are communicated, as is the case on coarse levels of AMG. This concept is introduced in Bienz et al. (2019) for the SpMV and is extended here to all components of the AMG setup and solve phases. In particular, a new two-step communication process is introduced for the finest levels of the AMG hierarchy.

#### 3.1. Background of node-aware communication

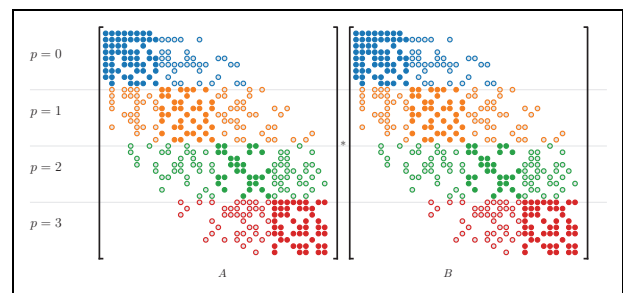
The cost of communication depends on many factors, such as number of messages, size of the messages, and relative locations of the send and receive processes. For instance,



**Figure 5.** The cost of iteratively solving the Ruge–Stüben hierarchies for the system from example 2.1 on various core counts. The percentage of total time spent communicating is listed at the top of the bars.

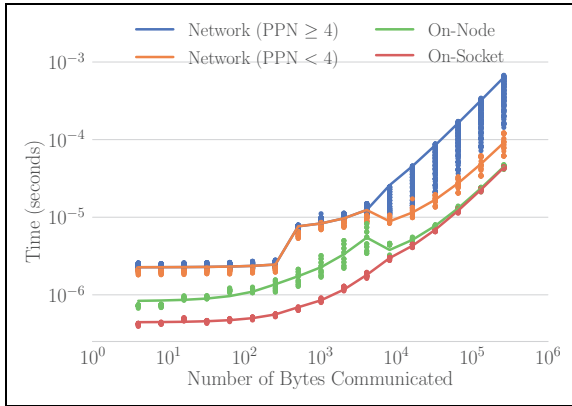


**Figure 6.** Matrix-vector multiplication across four processes, row wise.

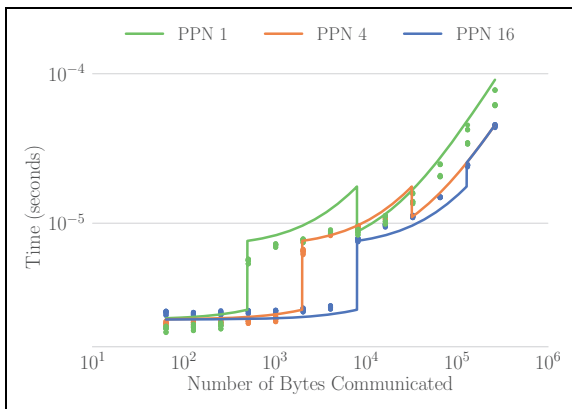


**Figure 7.** Two matrices partitioned across four processes in a row-wise manner.

messages between two processes on the same socket are significantly cheaper than communication between processes located on the same node but different sockets. Communication cost is further increased when the send and receive processes are on different nodes, requiring messages to be injected into the network. Figure 8 shows the difference in the cost of sending a single message relative to the location of participating processes, with measured

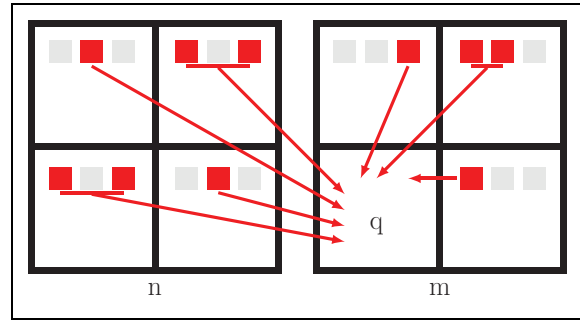


**Figure 8.** The cost of sending a single message between two processes located on the same socket, on different sockets of the same node, or on different nodes. The scattered dots represent timings, while the thick lines are corresponding models.

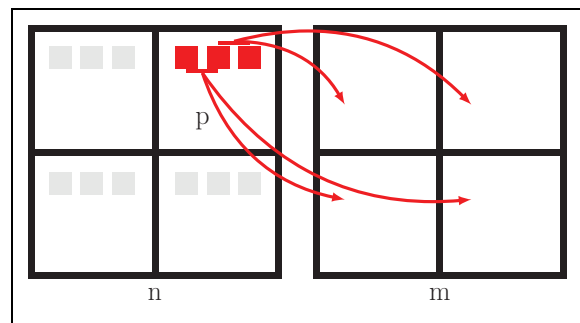


**Figure 9.** The cost of sending data between two nodes, with the data split evenly across all PPN. The dots are measured timings, while the lines are the corresponding max-rate models. PPN: processes per node.

timings represented as scattered dots while the corresponding thick lines display the associated model measurements. The model is calculated with the max-rate model, which adds bandwidth injection limits to the standard postal model (Gropp et al., 2016), and the ping-pong tests are measured with Nodecomm.<sup>1</sup> Furthermore, the cost of communicating data between nodes is dependent on the number of active processes, with the cost minimized as data is distributed across a larger number of processes. Figure 9 displays the cost of sending a single message between two nodes, with various numbers of active processes. Additional parameters not included in the max-rate model add to the cost of communication, such as queue search costs, which result from sending a large number of messages at once, and network contention, which occurs when many processes communicate large amounts of data across multiple links of the network (Bienz et al., 2018). The large costs associated with internode messages motivate replacing them with extra intra-node messages when possible. However, a node’s communication load should remain



**Figure 10.** Standard communication between processes on node  $n$  and a process  $q$  on node  $m$  yields multiple messages to be sent between the two nodes.

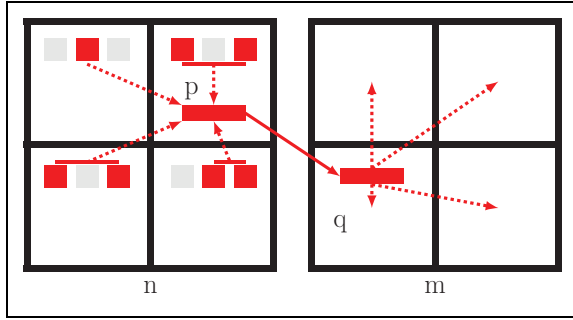


**Figure 11.** Standard communication from a process  $p$  to all processes on node  $m$  yields multiple messages to be communicated between nodes  $n$  and  $m$ , while also sending duplicate data through the network.

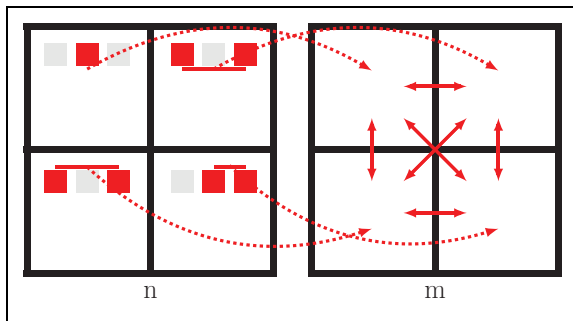
balanced with all  $ppn$  processes sending approximately  $\frac{1}{ppn}$  th of the data to minimize bandwidth costs, where  $ppn$  is the number of processes per node (PPN).

Standard communication requires sending data directly between processes, regardless of their locations within the parallel topology. For example, Figure 10 displays standard communication in which a number of processes on nodes  $n$  and  $m$  send data directly to a process  $q$ . Furthermore, Figure 11 shows the standard process of communicating data from some process  $p$  on node  $n$  to all processes on node  $m$ . In both cases, multiple messages are communicated between the two nodes. Furthermore, in the latter example, duplicate data are sent to multiple processes on node  $m$ , indicating that both the number and size of messages communicated between nodes  $n$  and  $m$  are larger than ideal.

Three-step node-aware parallel (NAP-3) communication reduces the number and size of messages injected into the network while increasing the amount of less-costly on-node communication (Bienz et al., 2019). NAP-3 communication gathers all data to be sent to node  $m$  on some process local to the node  $n$  on which it originates. These data are then sent as a single message through the network, before being distributed to the necessary processes on node  $m$ . Figure 12 displays the steps of sending data from node  $n$  to  $m$ , sending first to process  $p$  on node  $n$ . A single message is then sent from process  $p$  on node  $n$  to process  $q$  on node



**Figure 12.** Three-step node-aware communication: (1) gather data on a process  $p$  local to the node on which data originate, (2) send a single message between process  $p$  on node  $n$  and process  $q$  on node  $m$ , and (3) redistribute across processes on node  $m$ .



**Figure 13.** An alternative, two-step node-aware communication: (1) gather all data to be sent to node  $m$  on process and (2) send directly to the corresponding process on node  $m$ .

$m$ . Finally, process  $q$  distributes the received data to processes on node  $m$  that need it. In addition, all on-node messages, or those for which the process of origin lies on the same node as the destination process, are communicated with the standard approach. Moreover, it is likely that there are other nodes in the network to which processes on node  $n$  must also be send. These data are gathered on some process  $s$  on  $n$  that is not  $p$ . As a result, several PPN are communicating.

NAP-3 communication greatly reduces both the number of messages and the number of bytes injected into the network by any node. When a node  $n$  is communicating similar amounts of data to many other nodes, the per process message size is also greatly reduced. However, in the case that node  $n$  is communicating the majority of data to a single node  $m$ , a large imbalance can occur in communication requirements of the processes local to node  $n$ , reducing bandwidth and increasing message cost.

### 3.2. Two-step node-aware communication

Alternatively, this article introduces a method to allow *all* processes to remain active in internode communication. This two-step approach to node-aware parallel (NAP-2), displayed in Figure 13, consists of gathering all data on process to be sent to a node  $m$ , and sending this directly to the

corresponding process. This is followed by redistribution of values on the receiving node. This alternate node-aware method reduces the number and size of data by eliminating the duplication displayed in Figure 11, but the multiple messages communicated between nodes in Figure 10 remain. Therefore, NAP-2 communication can greatly reduce both the number of messages and bytes communicated over standard communication, while process loads remain equally balanced to standard communication. However, as up to  $ppn$  messages remain between each set of nodes, NAP-2 does not reduce the message count to the extent of NAP-3, and as a result is less beneficial when there is communication between a large number of nodes.

### 3.3. Performance models for communication strategies

The optimal communication strategy varies with problem type, problem scale, level in AMG hierarchy, and operation. Figure 14 displays the cost of performing various dominate matrix operations on each level of the Ruge–Stüben AMG hierarchy for the Laplace system from example 2.1. While node-aware communication often yields large speedups on coarse levels, additional work and load imbalance can significantly slowdown fine-level communication. Therefore, a performance model should be used to determine the ideal communication strategy for the various operations throughout AMG.

The max-rate model (Gropp et al., 2016) describes the cost of sending messages from a symmetric multiprocessing node as:

$$T = \alpha n + \frac{ppn \cdot s}{\min(R_N, ppn \cdot R_b)}, \quad (1)$$

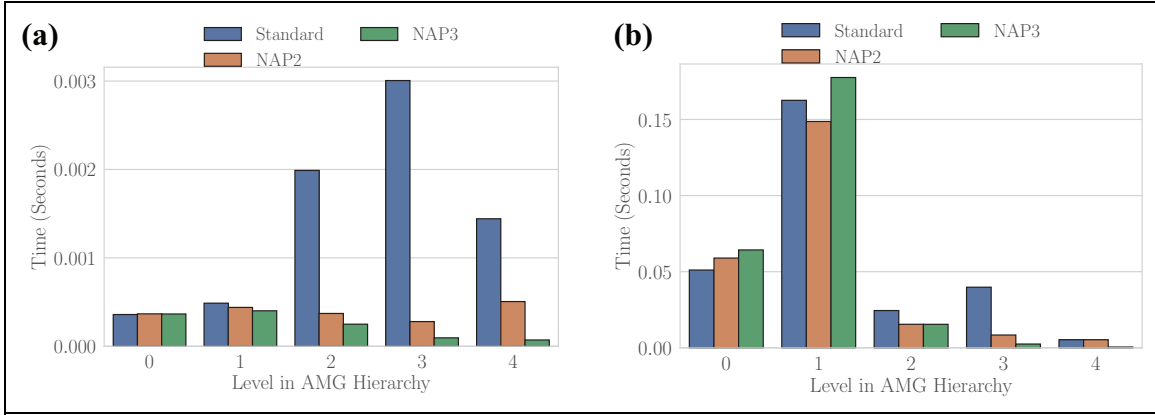
where  $\alpha$  is the latency,  $R_b$  is rate at which a process can transport data, and  $R_N$  is the rate at which a network interface device (NID) can inject data into the network. Furthermore,  $n$  is the maximum number of messages sent and  $s$  is the maximum number of bytes. This model assumes that all processes on a node communicate equal amounts of data. However, as NAP-3 can result in imbalanced communication loads, the max-rate model is altered to be:

$$T = \alpha n + \max\left(\frac{s_{node}}{R_N}, \frac{s_{proc}}{R_b}\right), \quad (2)$$

where  $s_{proc}$  is the maximum number of bytes communicated by any process and  $s_{node}$  is the maximum number of bytes injected by any NID. In the case of perfect load balance,  $s_{node}$  is equal to  $ppn \cdot s_{proc}$ , and equation (2) reduces to the original max-rate model. Furthermore, when modeling the cost of intra-node communication, the max-rate model reduces to the standard postal model:

$$T = \alpha_l n + \frac{s}{R_b}, \quad (3)$$

as data are not injected into the network. In this model,  $\alpha_l$  is the latency required to send a message to another process



**Figure 14.** Cost of standard, two-step node-aware, and three-step node-aware communication for SpGEMM and SpMV operations throughout smoothed aggregation hierarchies from example 2.1: (a)  $A \cdot x$ , (b)  $P^T \cdot AP$ . SpMV: sparse matrix-vector; SpGEMM: sparse matrix-matrix.

on-node, and  $R_b$ , is the rate at which data are transported between two on-node processes. In all models, the latency and bandwidth terms are measured and applied separately to short, eager, and rendezvous protocols.

The cost of each communication strategy can be approximated as the sum of the cost of internode messages, modeled by equation (2), and intra-node message cost as determined by equation (3). Furthermore, as fully intra-node communication is performed equivalently in all strategies, only internode communication and the node-aware approaches' additional intra-node communication requirements are modeled.

Standard communication is modeled as:

$$T = \alpha n_{\text{proc}} + \max\left(\frac{s_{\text{node}}}{R_N}, \frac{s_{\text{proc}}}{R_b}\right), \quad (4)$$

where  $n_{\text{proc}}$  is the maximum number of processes with which any process communicates.

Similarly, NAP-2 communication is modeled by:

$$T = \alpha n_{\text{proc}2\text{node}} + \max\left(\frac{s_{\text{node}}}{R_N}, \frac{s_{\text{proc}}}{R_b}\right) + \alpha_\ell(\text{ppn} - 1) + \frac{s_{\text{proc}}}{R_{b_i}}, \quad (5)$$

where  $n_{\text{proc}2\text{node}}$  is the maximum number of nodes with which any process communicates. Additional intra-node communication is modeled with an upper bound of  $\text{ppn} - 1$  messages, transferring a total of  $s_{\text{proc}}$  bytes. In this worst case, a process sends all received bytes to the  $\text{ppn} - 1$  other processes on the node.

Finally, NAP-3 communication is modeled as:

$$T = \alpha \frac{n_{\text{node}2\text{node}}}{\text{ppn}} + \max\left(\frac{s_{\text{node}}}{R_N}, \frac{s_{\text{node}2\text{node}}}{R_b}\right) + 2 \cdot \left(\alpha_\ell(\text{ppn} - 1) + \frac{s_{\text{node}2\text{node}}}{R_{b_i}}\right), \quad (6)$$

where  $n_{\text{node}2\text{node}}$  and  $s_{\text{node}2\text{node}}$  are the number and size of messages, respectively, communicated between any two nodes.

These models do not take into account reductions in the size of node-aware messages that result from removing duplicate data. Furthermore, the additional intra-node communication in the NAP-2 and NAP-3 models is a rough estimate of an upper bound. However, the models are accurate enough to distinguish between the various strategies.

Figure 15 shows the cost of communication in the SpGEMM  $A \cdot P$  and the SpMV  $A \cdot x$  on each level of the AMG hierarchy from example 2.1 for each of the communication strategies. Each operation is tested multiple times, and all timings are plotted. The timings corresponding to the communication strategy with minimal modeled measurement are displayed as larger dots. While the measurements vary among the runs, the model accurately chooses a communication strategy that performs at least as well as standard communication.

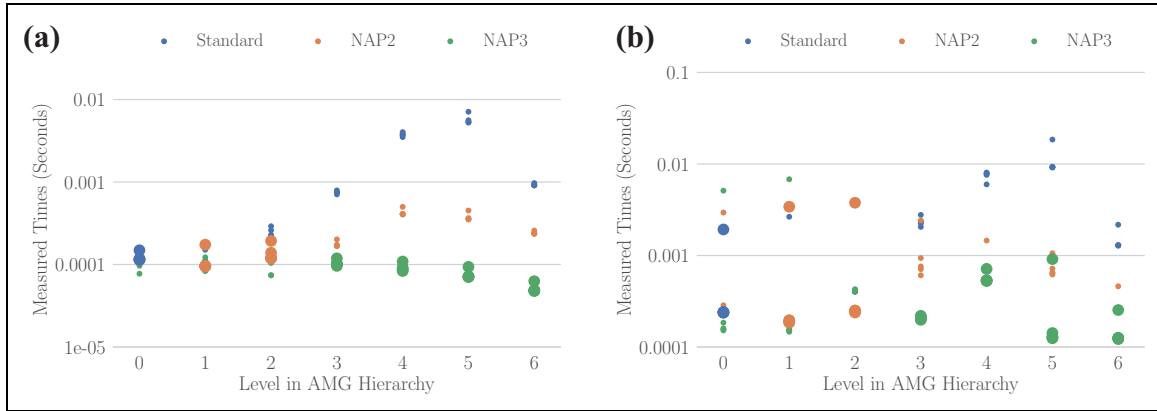
## 4. Results

Node-aware communication can be used throughout the dominant methods of the setup and solve phases. This section presents performance and scaling results of AMG with node-aware communication. Optimal strategies for vector and matrix communication are determined during the formation of each matrix in the AMG hierarchy. After a matrix is created, the performance models in equations (4) to (6) are calculated and the strategy with minimum modeled cost is chosen. Separate models are calculated for vector and matrix communication, allowing for different strategies for each type of communication.

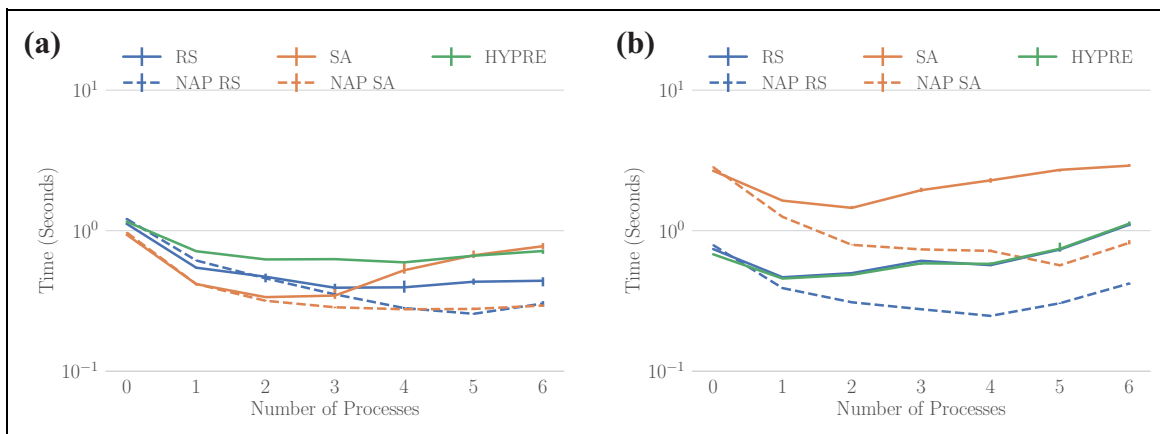
Throughout this section, both Ruge–Stüben and Smoothed Aggregation solvers are analyzed for the problems that follow.

- MFEM Laplace: The system from example 2.1.
- MFEM Grad-Div: The finite element discretization of  $-\nabla(\alpha \nabla \cdot (F)) + \beta F = f$ , created with MFEM,





**Figure 15.** The cost of various AMG operations throughout the Ruge–Stüben hierarchy for example 2.1. Each plot contains the operation cost associated with standard, NAP-2, and NAP-3 communication. The larger dots represent the method chosen as minimal by the performance models. Each timing was tested on five separate runs. Note that interpolation filtering is not included in the formation of these hierarchies. (a)  $A \cdot x$ , (b)  $A \cdot P$ . AMG: algebraic multigrid; NAP-2: two-step approach to node-aware communication; NAP-3: three-step node-aware parallel.



**Figure 16.** (MFEM Grad-Div) Setup and solve times for both the Ruge–Stüben and smoothed aggregation hierarchies for the MFEM Grad-Div system. The system has 2801664 degrees of freedom and 117107712 nonzeros. Each test was performed five times, with a variety of partitions of Blue Waters. The error bars, which represent the run-to-run variation in timings, line up closely to average run times but were included to show speedup is significantly greater than run-to-run variation: (a) setup cost, (b) solve cost.

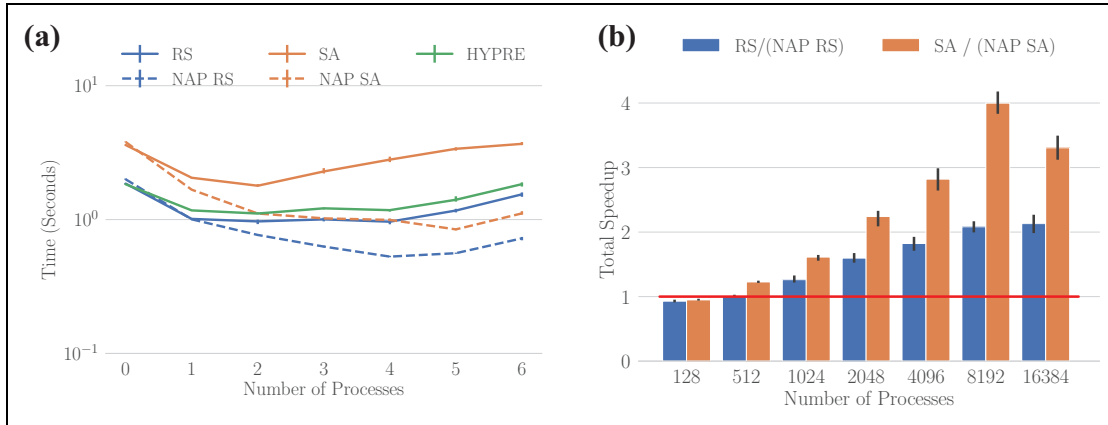
on the three-dimensional fichera-q3 mesh. The system has 2801664 degrees of freedom and 117107712 nonzeros, unless otherwise specified.

- MFEM discontinuous Petrov–Galerkin (DPG) Laplace: The DPG discretization of the Laplace system  $-\Delta u = 1$ , created with MFEM, on the three-dimensional star-q3 mesh. This system contains 131720 rows and 104529920 nonzeros, unless otherwise specified.

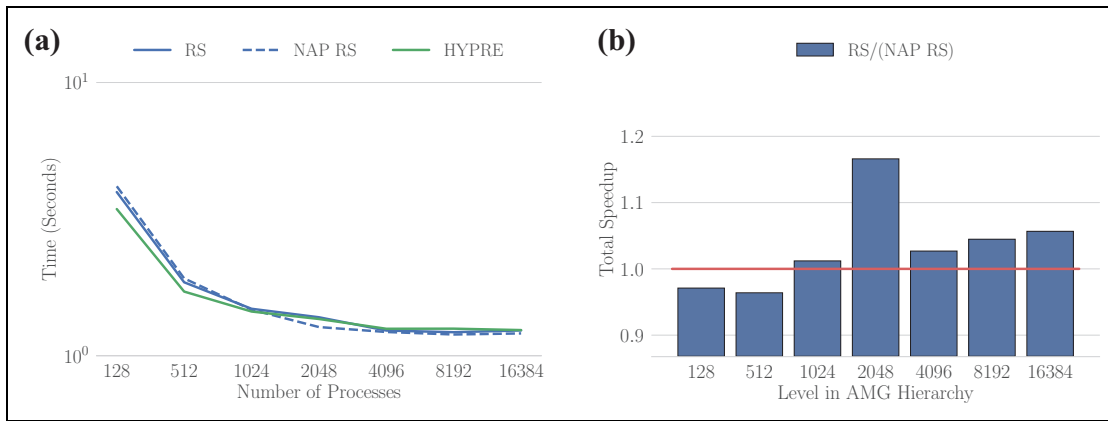
The Ruge–Stüben hierarchies for these systems are aggressively coarsened with HMIS and extended + i interpolation with an interpolation truncation threshold of 0.3. The smoothed aggregation hierarchies are created with aggregates based on an MIS-2 of the graph. All systems are solved to a relative residual of  $10^{-7}$ . All tests are performed with RAPtor (Bienz and Olson, 2017) and compared against

an identical Ruge–Stüben hierarchy that is created and solved with a state-of-the-field solver, Hypre’s Boomer AMG (Henson and Yang, 2002; LLNL, 2008). Performance tests are run on both Blue Waters, a Cray supercomputer at the National Center for Supercomputing Applications (Bode et al., 2013; NCSA, 2012), and Quartz, an Intel Xeon E5 machine at Lawrence Livermore National Laboratory. All Blue Waters tests are performed with 16 PPN, while Quartz timings are acquired with 32 PPN.

Figure 16 displays the costs of both setting up and solving Ruge–Stüben and smoothed aggregation hierarchies for the MFEM Grad-Div system on various core counts of Blue Waters. Timings are plotted both with and without node-aware communication. Each test was performed five times with various partitions, and the plots show minimal variation between runs. While minimal improvements are obtained at small core counts, node-aware communication



**Figure 17.** (MFEM Grad-Div) Total AMG times for both the Ruge–Stüben and smoothed aggregation hierarchies for MFEM Grad-Div on Blue Waters. The system has 2801664 degrees of freedom and 117107712 nonzeros. Each timing was test over five separate runs, with a variety of partitions of Blue Waters. The error bars, which represent the run-to-run variation in timings, line up closely to average run times but were included to show speedup is significantly greater than run-to-run variation: (a) total cost, (b) node-aware speedup. AMG: algebraic multigrid.



**Figure 18.** (MFEM DPG) Total AMG times for both the Ruge–Stüben hierarchies for MFEM DPG Laplace on Blue Waters. This system contains 131,720 rows and 104529920 nonzeros. Smoothed aggregation results are not included due to poor convergence at strong scales: (a) total cost; (b) node-aware speedup. AMG: algebraic multigrid.

yields increased improvements as the problem is strongly scaled across processes, particularly in the solve phase of AMG. The cost of both determining the appropriate communication strategy through models and forming node-aware communicators is included in the setup phase costs.

The total cost of solving the MFEM Grad-Div system with AMG on Blue Waters is displayed in Figure 17 alongside speedups achieved with node-aware communication. The system was solved five times on various partitions of Blue Waters, with the plots displaying minimal variation between runs. Large improvements are obtained over AMG with standard communication as the core count is increased, with a nearly 4× speedup near the strong scaling limits of each solver.

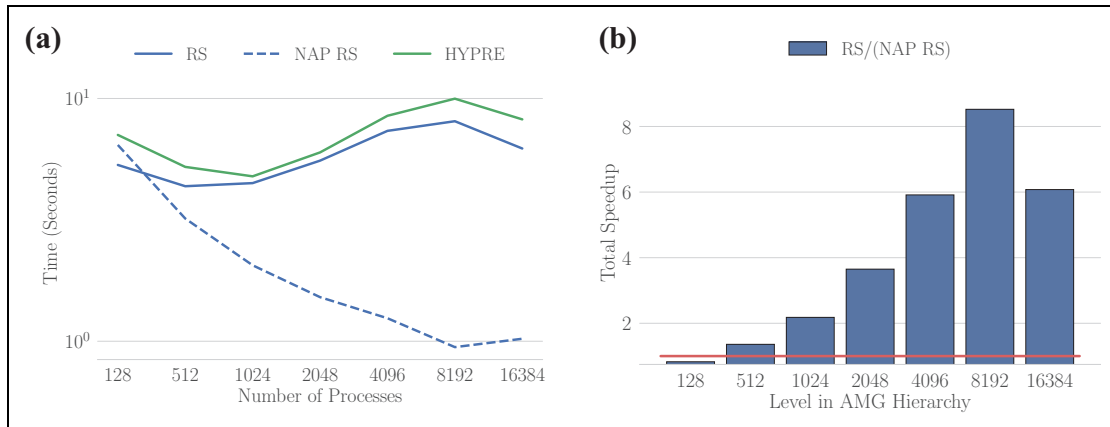
The total AMG costs and node-aware speedups associated with solving the MFEM DPG Laplace system on Blue Waters are displayed in Figure 18. While performance improvements are less drastic than seen in the MFEM

Grad-Div system, strong scalability is extended to at least 16,384 processes for both solvers. Furthermore, the speedups increase with process count.

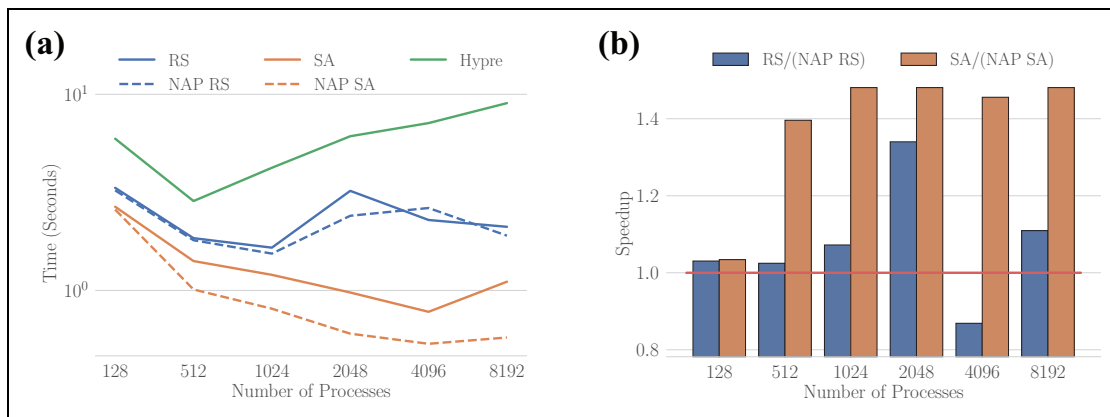
Similarly, Figure 19 displays the costs and speedups associated with node-aware Ruge–Stüben AMG when solving the MFEM Laplace system on Blue Waters.

The node-aware communication strategies can be naturally extended to other architectures with different numbers of cores per node, such as Quartz. The total AMG costs and node-aware speedups associated with solving the MFEM Grad-Div system on Quartz are displayed in Figure 20. As with the Blue Waters results, the total cost of AMG is improved most drastically as the problem is strongly scaled across the processors.

Similar performance is obtained when weakly scaling the system, as shown in Figure 21. Node-aware communication improves performance at the various weak scales, with more drastic improvements as the scale increases. In



**Figure 19.** (MFEM Laplace) Total AMG times for both the Ruge–Stüben hierarchies for MFEM Laplace from example 2.1 on Blue Waters. This system consists of 1884545 degrees of freedom and 27870337 nonzeros. Smoothed aggregation results are not included due to poor convergence at strong scales: (a) total cost; (b) node-aware speedup. AMG: algebraic multigrid.



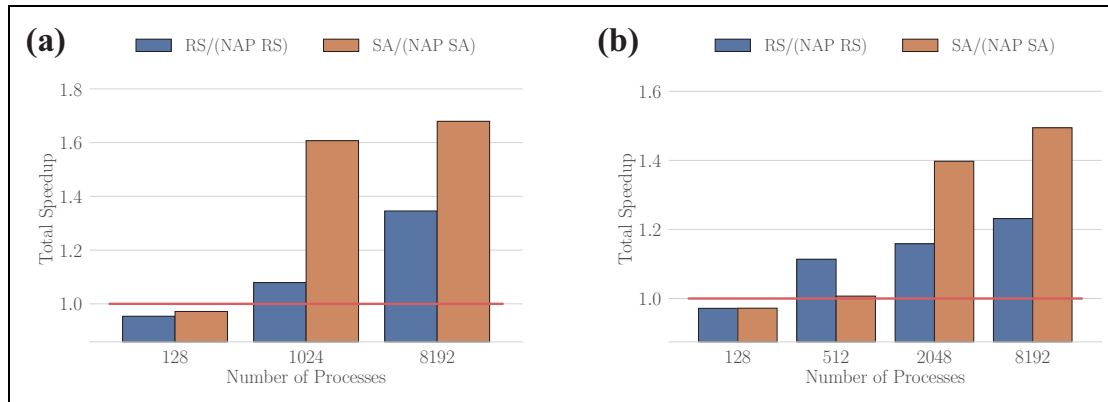
**Figure 20.** (MFEM Grad-Div, Quartz) Total AMG times for both the Ruge–Stüben and smoothed aggregation hierarchies for MFEM Grad-Div on Quartz. The system has 2801664 degrees of freedom and 117107712 nonzeros. Note that these hierarchies do not include interpolation filtering, and as a result, timings could be improved: (a) total cost; (b) node-aware speedup. AMG: algebraic multigrid.

general, node-aware communication yields the most dramatic improvements when communication dominates, such as at strong scales and large process counts.

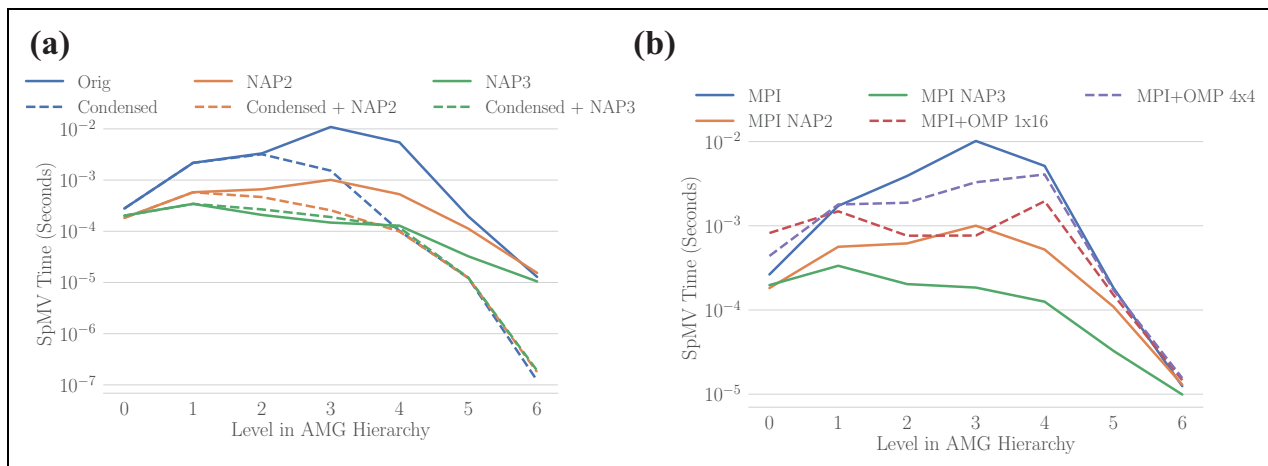
Node-aware communication can be used in combination with other optimizations, such as repartitioning coarse levels. Figure 22(a) displays the cost of performing a SpMV on each level of an AMG hierarchy with the natural partitions resulting from the fine-level Galerkin product compared to a hierarchy with coarse levels repartitioned such that the number of nonzeros per process is constant on all levels, when possible. The repartitioned coarse-level matrices are condensed onto the processes with lowest ranks such that each process with sufficiently low rank holds an even row-wise partition of the matrix, while extra processes sit idle. Graph partitioning could further improve communication requirements of condensed matrices but at an additional cost, as discussed in Bienz et al. (2019). The cost of reordering the matrix is not included in this study, and the trade-off between additional setup cost and improved SpMV times should be further investigated.

Furthermore, while condensing the matrix onto fewer processes yields significant improvements for the coarsest levels, node-aware communication complements this technique, yielding improved performance on costly levels near the middle of the hierarchy.

Similarly, MPI + X strategies reduce communication costs by limiting the number of processes active in communication and overlapping with local computation. Figure 22(b) displays the cost of performing a SpMV on every level of the hierarchy, comparing the pure MPI strategies presented in this article with MPI + X approaches, using OpenMP for on-node computations with MPI used for internode communication. The MPI + X approach yields improvement over standard communication on coarse levels as fewer processes are active in communication, reducing duplicate internode messages. However, the node-aware approaches further improve communication costs by distributing these messages across all 16 PPN. Node-aware communication can be used in combination with MPI + X task-based approaches to further reduce



**Figure 21.** Node-aware speedups achieved for both the Ruge–Stüben and smoothed aggregation hierarchies for weakly scaled MFEM systems. All systems have approximately 10000 degrees of freedom per core: (a) MFEM Grad-Div, (b) MFEM DPG Laplace.



**Figure 22.** Comparisons of node-aware SpMVs with other communication optimizations. Note that interpolation filtering was not included in the formation of these hierarchies. (a) The cost of performing a SpMV on each level of the Ruge–Stüben hierarchy for the MFEM Grad-Div system with original partitions compared to condensed partitions where the number of nonzeros per process stays constant, when possible. Levels 1 and 2 are not condensed as the total number of nonzeros is larger than the original system. Note that the additional cost required to condense the matrices across processes is not included in this figure. (b) The cost of performing a SpMV on each level of the Ruge–Stüben hierarchy for the MFEM Grad-Div system with pure MPI, compared to an MPI + X approach. The MPI + X approaches consist of one MPI process per node with 16 OpenMP (OMP) threads per process, labeled MPI + OMP 1 × 16, and four MPI PPN with four OMP threads per process, labeled MPI + OMP 4 × 4. SpMV: sparse matrix-vector; PPN: processes per node.

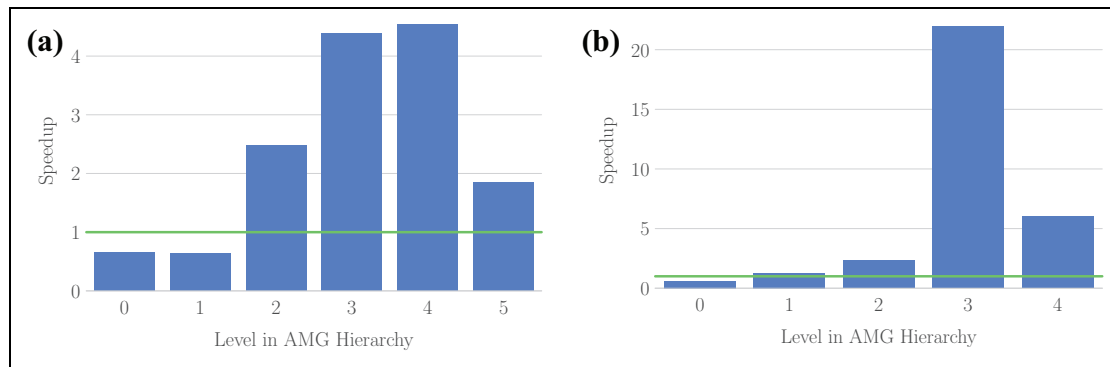
these costs by overlapping these reduced internode communication costs with local computation.

Finally, further improvements to node-aware AMG are possible, as many operations yield little to no speedup, particularly in the setup phase. While the majority of operations throughout the setup phase are dependent on the sparsity pattern of  $A$ , the transpose multiplication step  $P^T \cdot AP$  is dependent on the sparsity pattern of  $P$ . Therefore, node-aware communication will have larger improvements for transpose multiplication with denser  $P$  matrices. This motivates increasing the density of  $P$  to increase the accuracy of projecting data between methods, such as using multiple sweeps of Jacobi smoothing during the smoothed aggregation AMG setup. Figure 23 shows the speedups obtained with node-aware communication during  $P^T \cdot AP$  on each level for both the standard hierarchy and

when using multiple smoothing sweeps. There is significant speedup for the denser  $P$  resulting from the latter.

## 5. Conclusions

This article has introduced a method of using node awareness to reduce the amount of internode communication at a trade-off of less costly intra-node communication and applied this communication method to the various parts of AMG. Node-aware communication yields improvements in the performance and scalability of both the setup and solve phases for a variety of three-dimensional matrices created with MFEM. Node-aware communication yields improvements over the state-of-the-field solver, Hypre. Furthermore, performance and parallel scaling improvements are obtained on both Blue Waters and



**Figure 23.** Speedup from node-aware communication in  $P^T \cdot (AP)$  throughout the smoothed aggregation hierarchy for a stenciled two-dimensional rotated anisotropic diffusion system when using a single sweep of Jacobi prolongation, as typical, or increasing to two smoothing sweeps: (a) one sweep, (b) two sweeps.

Quartz. Future work will be done to further improve the performance of node-aware communication by improving the underlying performance models and allowing for different communication strategies for each node. Node-aware communication can be combined with other existing strategies of reducing communication such as repartitioning coarse levels and MPI + X approaches to further reduce costs in AMG. Furthermore, this strategy can be extended to a variety of other methods as well as emerging architectures.


#### Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

#### Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research is part of the Blue Waters sustained petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the State of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. This material is based in part upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant Number DGE-1144245. This material is based in part upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number DE-NA0002374.

#### ORCID iD

Amanda Bienz  <https://orcid.org/0000-0002-8891-934X>

#### Note

1. See <http://wgropp.cs.illinois.edu/projects/software/baseenv.htm>.

#### References

- Adams MF, Bayraktar HH, Keaveny TM, et al. (2004) Ultrascaleable implicit finite element analyses in solid mechanics with over a half a billion degrees of freedom. In: *SC '04: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*, 6–12 November 2004, pp. 34–34. IEEE. DOI:10.1109/SC.2004.62.
- AlOnazi A, Markomanolis GS and Keyes D (2017) Asynchronous task-based parallelization of algebraic multigrid. In: *Proceedings of the platform for advanced scientific computing conference, PASC '17*. ISBN 978-1-4503-5062-4, 26–28 June 2017, pp. 5: 1–5:11. New York, NY, USA: ACM. DOI:10.1145/3093172.3093230.
- Baker AH, Gamblin T, Schulz M, et al. (2011) Challenges of scaling algebraic multigrid across modern multicore architectures. In: *Proceedings of the 2011 IEEE international parallel & distributed processing symposium, IPDPS '11*. ISBN 978-0-7695-4385-7, 16–20 May 2011, pp. 275–286. Washington, DC, USA: IEEE Computer Society. DOI:10.1109/IPDPS.2011.35.
- Baker AH, Schulz M and Yang UM (2011) On the performance of an algebraic multigrid solver on multicore clusters. In: Palma JMLM, Daydé M, Marques O and Lopes JC (eds) *High Performance Computing for Computational Science—VECPAR 2010*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-19328-6, pp. 102–115.
- Bhatele A and Kale LV (2008) Application-specific topology-aware mapping for three dimensional topologies. In: *2008 IEEE international symposium on parallel and distributed processing*, 14–18 April 2008, pp. 1–8. IEEE. DOI:10.1109/IPDPS.2008.4536348.
- Bhatele A, Gamblin T, Langer SH, et al. (2012) Mapping applications with collectives over sub-communicators on torus networks. In: *Proceedings of the International conference on high performance computing, networking, storage and analysis, SC '12*. ISBN 978-1-4673-0804-5, pp. 97:1–97:11. Los Alamitos, CA, USA: IEEE Computer Society Press. DOI:10.1109/SC.2012.75.
- Bienz A and Olson LN (2017) RAPtor: parallel algebraic multigrid v0.1. Available at: <https://github.com/lukeolson/raptor>. Release 0.1 (accessed February 2019).

- Bienz A, Falgout RD, Gropp W, et al. (2016) Reducing parallel communication in algebraic multigrid through sparsification. *SIAM Journal on Scientific Computing* 38(5): S332–S357.
- Bienz A, Gropp WD and Olson LN (2018) Improving performance models for irregular point-to-point communication. In: *Proceedings of the 25th European MPI Users' Group Meeting*, Barcelona, Spain, 23–26 September 2018, pp. 7:1–7:8. Association for Computing Machinery. DOI:10.1145/3236367.3236368.
- Bienz A, Gropp WD and Olson LN (2019) Node aware sparse matrix-vector multiplication. *Journal of Parallel and Distributed Computing* 130: 166–178. DOI:10.1016/j.jpdc.2019.03.016.
- Bode B, Butler M, Dunning T, et al. (2013) The Blue Waters super-system for super-science. In: Vetter JS (ed) *Contemporary High Performance Computing: From Petascale Toward Exascale*, CRC Computational Science Series, Vol. 1. 1st ed. Boca Raton: Taylor and Francis, pp. 339–366. ISBN 9781466568341.
- Brandt A, McCormick SF and Ruge JW (1984) Algebraic multigrid (AMG) for sparse matrix equations. In: DJ Evans (ed) *Sparsity and Its Applications*. Cambridge: Cambridge Univ. Press, pp. 257–284.
- Çatalyürek UV and Aykanat C (1999) Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on Parallel and Distributed Systems* 10(7): 673–693.
- Çatalyürek UV and Aykanat C (2010) On two-dimensional sparse matrix partitioning: models, methods, and a recipe. *SIAM Journal on Scientific Computing* 32(2): 656–683.
- Falgout RD and Schroder JB (2014) Non-Galerkin coarse grids for algebraic multigrid. *SIAM Journal on Scientific Computing* 36(3): C309–C334.
- Gropp W, Lusk E, Doss N, et al. (1996) A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing* 22(6): 789–828. DOI:10.1016/0167-8191(96)00024-5.
- Gropp W, Olson LN and Samfass P (2016) Modeling MPI communication performance on SMP nodes: Is it time to retire the ping pong test. In: *Proceedings of the 23rd European MPI Users' Group Meeting*, EuroMPI 2016. ISBN 978-1-4503-4234-6, pp. 41–50. New York, NY, USA: ACM. DOI:10.1145/2966884.2966919.
- Hendrickson B and Kolda TG (2000) Graph partitioning models for parallel computing. *Parallel Computing* 26(12): 1519–1534.
- Henson VE and Yang UM (2002) BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics* 41(1): 155–177.
- Hoeffler T and Traff JL (2009) Sparse collective operations for MPI. In: *2009 IEEE international symposium on parallel distributed processing*, Rome, Italy, 23–29 May 2009, pp. 1–8. IEEE. DOI:10.1109/IPDPS.2009.5160935.
- Karonis NT, de Supinski BR, Foster I, et al. (2000) Exploiting hierarchy in parallel computer networks to optimize collective operation performance. In: *Proceedings 14th international parallel and distributed processing symposium. IPDPS 2000*, Cancun, Mexico, 1–5 May 2000, pp. 377–384. IEEE. DOI:10.1109/IPDPS.2000.846009.
- Kielmann T, Hofman RFH, Bal HE, et al. (1999) MagPIe: MPI's collective communication operations for clustered wide area systems. *SIGPLAN Not* 34(8): 131–140.
- Lawrence Livermore National Laboratory (LLNL) (2008) HYPRE: high performance preconditioners. Available at: <http://www.llnl.gov/CASC/hypre/> (accessed February 2019).
- Lawrence Livermore National Laboratory (LLNL) (2010) MFEM: modular finite element methods. Available at: [mfem.org](http://mfem.org) (accessed February 2019).
- McCormick SF and Ruge JW (1982) Multigrid methods for variational problems. *SIAM Journal on Numerical Analysis* 19(5): 924–929.
- Mirsadeghi S, Traff J, Balaji P, et al. (2017) Exploiting common neighborhoods to optimize MPI neighborhood collectives. In: *2017 IEEE 24th international conference on high performance computing (HiPC)*, Jaipur, India, 18–21 December 2017, pp. 348–357. Los Alamitos, CA, USA: IEEE Computer Society. DOI:10.1109/HiPC.2017.00047.
- National Center for Supercomputing Applications (2012) Blue Waters. Available at: <https://bluewaters.ncsa.illinois.edu/>.
- Pinar A and Aykanat C (2004) Fast optimal load balancing algorithms for 1D partitioning. *Journal of Parallel and Distributed Computing* 64(8): 974–996.
- Ruge JW and Stüben K (1987) Algebraic multigrid (AMG). In: SF McCormick (ed) *Multigrid Methods, Frontiers in Applied Mathematics*. Philadelphia: SIAM, pp. 73–130.
- Sack P and Gropp W (2012) Faster topology-aware collective algorithms through non-minimal communication. In: *Proceedings of the 17th ACM SIGPLAN symposium on principles and practice of parallel programming*, PPoPP '12. ISBN 978-1-4503-1160-1, New Orleans, Louisiana, USA, 25–29 February 2012, pp. 45–54. New York, NY, USA: ACM.
- Solomonik E, Bhatlele A and Demmel J (2011) Improving communication performance in dense linear algebra via topology aware collectives. In: *Proceedings of 2011 international conference for high performance computing, networking, storage and analysis*, SC '11. ISBN 978-1-4503-0771-0, Seattle, WA USA, 12–18 November 2011, pp. 77:1–77:11. New York, NY, USA: ACM.
- Sterck HD, Falgout RD, Nolting JW, et al. (2008) Distance-two interpolation for parallel algebraic multigrid. *Numerical Linear Algebra with Applications* 15(2-3): 115–139.
- Sterck HD, Yang UM and Heys JJ (2005) Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM Journal on Matrix Analysis and Applications* 27(4): 1019–1039.
- Treister E and Yavneh I (2015) Non-Galerkin multigrid based on sparsified smoothed aggregation. *SIAM Journal on Scientific Computing* 37(1): A30–A54.
- Tuminaro RS and Tong C (2000) Parallel smoothed aggregation multigrid: aggregation strategies on massively parallel machines. In: *Supercomputing, ACM/IEEE 2000 conference*, Dallas, TX USA, 4–10 November 2000, pp. 5–5. IEEE. DOI: 10.1109/SC.2000.10008.
- Vassilevski PS and Yang UM (2014) Reducing communication in algebraic multigrid using additive variants. *Numerical Linear*

- Algebra with Applications* 21(2): 275–296. DOI: 10.1002/nla.1928.
- Vastenhouw B and Bisseling RH (2005) A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM Review* 47(1): 67–95.
- Wesolowski L, Venkataraman R, Gupta A, et al. (2014) TRAM: Optimizing fine-grained communication with topological routing and aggregation of messages. In: *2014 43rd International Conference on Parallel Processing*, Minneapolis, MN USA, 9–12 September 2014, pp. 211–220. IEEE. DOI:10.1109/ICPP.2014.30.
- Yang UM (2010) On long-range interpolation operators for aggressive coarsening. *Numerical Linear Algebra with Applications* 17(2–3): 453–472.

### Author biographies

*Amanda Bienz* is a postdoc in the Department of Computer Science at the University of Illinois at Urbana–Champaign. She received her PhD in computer science from University of Illinois in 2018. Her research is focused on using performance models to reduce parallel communication costs, with a focus on sparse matrix operations, graph algorithms, iterative methods, and heterogenous architectures.

*William D Gropp* is the director and chief scientist of the National Center for Supercomputing Applications and holds the Thomas M. Siebel Chair in the Department of Computer Science at the University of Illinois in Urbana–Champaign. He received his PhD in computer science from Stanford University in 1982. He was on the faculty of the Computer Science Department of Yale University from 1982 to 1990 and from 1990 to 2007, and he was a member of the Mathematics and Computer Science Division at Argonne National Laboratory. His research interests are in parallel computing, software for scientific computing, and numerical methods for partial differential equations. He is a Fellow of American Association for the Advancement of Science, Association for Computing Machinery, Institute of Electrical and Electronics Engineers, and Society for Industrial and Applied Mathematics (SIAM) and a member of the National Academy of Engineering.

*Luke N Olson* is a professor in the Department of Computer Science at the University of Illinois at Urbana–Champaign. His PhD is in applied mathematics from the University of Colorado at Boulder in 2003 and his research interests include sparse matrix computations, multigrid methods, finite elements methods, and parallel numerical algorithms.