A Visual Analytics System for Optimizing Communications in Massively Parallel Applications

Takanori Fujiwara[*]
University of California, Davis

Preeti Malakar[†]
Argonne National Laboratory

Khairi Reda[‡]
Indiana University-Purdue University Indianapolis

Venkatram Vishwanath[§]
Argonne National Laboratory

Michael E. Papka[¶]
Argonne National Laboratory
Northern Illinois University

Kwan-Liu Ma[‖]
University of California, Davis

Abstract

Current and future supercomputers have tens of thousands of compute nodes interconnected with high-dimensional networks and complex network topologies for improved performance. Application developers are required to write scalable parallel programs in order to achieve high throughput on these machines. Application performance is largely determined by efficient inter-process communication. A common way to analyze and optimize performance is through profiling parallel codes to identify communication bottlenecks. However, understanding gigabytes of profile data is not a trivial task. In this paper, we present a visual analytics system for identifying the scalability bottlenecks and improving the communication efficiency of massively parallel applications. Visualization methods used in this system are designed to comprehend large-scale and varied communication patterns on thousands of nodes in complex networks such as the 5D torus and the dragonfly. We also present efficient rerouting and remapping algorithms that can be coupled with our interactive visual analytics design for performance optimization. We demonstrate the utility of our system with several case studies using three benchmark applications on two leading supercomputers. The mapping suggestion from our system led to 38% improvement in hop-bytes for MiniAMR application on 4,096 MPI processes.

Keywords: Supercomputing, parallel communications, performance analysis, visual analytics, communication visualization

Index Terms: I.3.8 [Computer Graphics]: Applications

1 Introduction

Today's fastest supercomputers [73] have hundreds of thousands of multi-core nodes that enable solutions to "grand challenge" problems in science and engineering [4, 5]. Several computational science simulations have scaled to millions of cores in order to model, at high fidelity, the underlying complex physics in domains such as cosmology, climate, and material science [34, 36, 39, 45, 63]. Typically, the application domain is decomposed over several thousand processor cores and involves a large number of communications among the processes. The compute nodes are typically connected via a high-bandwidth low-latency network with a specific topology [25]. Network topologies have evolved from fat-tree [52, 62] and butterfly [40] to high-dimensional torus [3] and high-radix dragonfly [48] to achieve lower latency and higher bandwidth for faster communications. Modern interconnects have network bandwidths in the range of Gigabytes/s whereas we can achieve more than a Teraflop/s computing speed per node [71]. Therefore, fast inter-process communications between the compute nodes are necessary to achieve maximum parallel efficiency.

Several studies [9, 56, 66, 72] show that communication slowdown can lead to poor scalability of parallel applications. This is not only true for communications between the parallel processes, but also for communications involved during input and output of data to/from the storage [7, 51]. Hence application developers spend significant amount of time in optimizing communication performance [9, 28, 80]. This task is challenging in supercomputers because of the scale of the nodes involved, the complex network topologies, and the proprietary routing algorithms used in these interconnects. Additionally, the communication performance is also dependent on the underlying implementation as well as on the requirements of the application [76]. Performance profiling tools [2, 49, 60, 65] are commonly used to analyze the communication behavior. These tools collect various communication profile data, such as the source and destination process information involved in a communication, the message size, and type of communication. However, the profiling data generated from such tools is very large (order of gigabytes [1]) due to the number of nodes and the number of communications in high performance applications.

Visualization can help us better comprehend the complex and massive profiling data. Several tools have been developed to visualize communications in a parallel application [13, 22, 23, 43, 60, 82]. Most of these tools visualize the readily available performance profile data such as communication endpoints and data size, which helps to know the communication pattern. However, such collective information is not useful to isolate the primary communication bottlenecks. These may arise due to several communications on the same network links and/or long-hop communications along congested links [13, 15, 55]. It is important to identify such hot spots to reduce communication costs. Visualizing such communications on supercomputers requires us to 1) show the behavior of a large number of compute nodes, networks, and communications, 2) depict high-dimensional and complex network topologies, and 3) analyze and mitigate communication bottlenecks. However, existing tools do not handle all these requirements.

In this paper, we present a comprehensive visual analytics system for identifying the origin of communication bottlenecks in various applications on diverse supercomputers. We focus on analyzing point-to-point MPI communications [35] and deterministic routing [27] for data movement. To help a user identify congested routes and unused links, our system effectively visualizes all the communication paths and the network simultaneously. Additionally, our system can suggest rerouting of communications for improved load balance and performance based on the visualized results. Our system also recommends improved process-to-processor mapping (placement of MPI ranks on the physical cores) to reduce the communication time among ranks that frequently communicate with each other. These rerouting and remapping algorithms that help improve communication efficiency are integrated into our interactive

[*]e-mail: tfujiwara@ucdavis.edu
[†]e-mail: pmalakar@anl.gov
[‡]e-mail: redak@iu.edu
[§]e-mail: venkat@anl.gov
[¶]e-mail: papka@anl.gov
[‖]e-mail: ma@cs.ucdavis.edu

visual analytics system. Following are our main contributions.

- A visual analytics system designed for identifying communication bottlenecks and improving communication performance.
- Scalable and flexible visual representations for communications between large numbers of compute nodes on complex networks.
- Algorithms to suggest better routes and mapping through user interaction with the visual analytics process.
- Several case studies to demonstrate improvement in communications of benchmark applications on two leading supercomputers. Communication bottlenecks are identified and performance improvement has been verified experimentally.

2 Related Work

Several performance analysis and visualization tools have been developed for parallel applications. Isaacs et al. [42, 43] and Gao et al. [29] provide a comprehensive survey of performance visualizations. General-purpose performance analysis tools such as HPCToolkit [2], Scalasca [30], Pajé [21], Vampir [60], CrayPat [22], and TAU [67] provide graphical results of performance profiles. These tools apply conventional visualizations that are usually not scalable.

State-of-the-art network interconnects enabling fast communications are quite complex, for example, the IBM Blue Gene/Q (BG/Q) and the K computer have 5D [16] and 6D torus [6] respectively. Several studies focus on visualizing the physical node locations in these complex networks. *Boxfish* [44, 50] visualizes performance data on 3D torus networks by using a 3D mesh representation or by projecting the result onto a 2D plane. McCathy et al. [57] extended this 3D to 2D projection method for visualizing the 5D torus. Theisen et al. [77] also proposed a projection method for the high-dimensional torus network. Chen et al. [18] developed *TorusVis*$^{\mathrm{ND}}$ which can be applied on any high-dimensional torus. They use space-filling curve [64] and a radial layout [24] to preserve distances and continuities in the torus network as much as possible. These studies focus more on node/network layout. Our goal is to visualize the communication paths and detect bottlenecks.

Bhatele et al. [13] analyzed network performance in dragonfly-based supercomputers and developed *DragonView* to visualize the results. *DragonView* applies a radial layout and a matrix view together to show inter-group and intra-group links between the compute nodes separately. Zhou et al. [83] used an adjacency matrix to visualize communication between neighboring nodes on fat-tree network [52, 62] based systems. We developed a generic adjacency matrix representation to visualize the physical locations of compute nodes on diverse networks. Sigovan et al. visualize MPI traces using animated scatterplot [70] and I/O traces on Blue Gene/P using a radial layout [69]. However, they do not show the communications between processes explicitly. In contrast to the above tools, we introduce an interactive analytics system to visualize communication routes, identify communication bottlenecks, and recommend solutions to improve performance. Also, our system is designed to concurrently show communication routes as well as network congestion between links, while other systems only provide the latter. Our system can also handle large-scale communications involving several thousands of compute nodes.

Strategies to improve communication performance include improved routing [8, 15, 55, 76] and topology-aware mapping [11, 74]. Bui et al. [15] developed algorithms to find better routes among multiple shortest paths to re-balance load on BG/Q while Hoefler et al. [38] developed heuristics to remap MPI ranks based on the communications patterns. Bhatele et al. [12, 14] proposed a framework that can find regular and irregular communication patterns in applications and map them to 2D and 3D torus automatically. We provide mapping suggestions to address potential communication bottlenecks as identified by our visual analysis. This is a scalable approach to improve performance. It helps system designers to detect

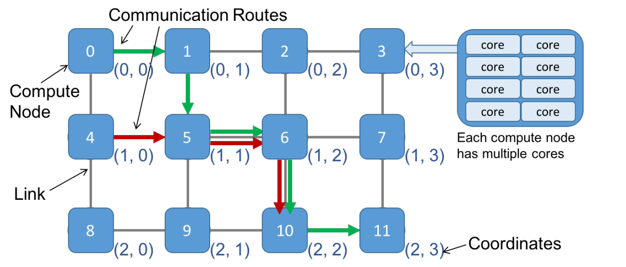


Figure 1: An example of communications in a 2D mesh network. Compute node 0 (0,0) communicates with 11 (2,3) along the green route. Compute node 4 (1,0) communicates with 10 (2,2) along the red route. We can reduce load on links between compute nodes 5, 6, and 10 by rerouting the red route.

inefficient mappings and design efficient routing policies. Tuncer et al. [79] developed a graph-based mapping algorithm for unstructured communication patterns on non-contiguous allocations. Sudheer et al. [74] model the mapping problem as a Quadratic Assignment Problem (QAP) with the hop-byte metric. They use graph partitioning heuristic to solve large-scale problems. We adopt a similar approach, however, we use adjacency matrix to partition the distance matrix. This gave us huge improvement as compared to no improvement with their approach of using distance matrix. Furthermore, our algorithms for rerouting and remapping can be applied on user-selected communication routes and solutions are suggested interactively.

3 Visual Analytics Process

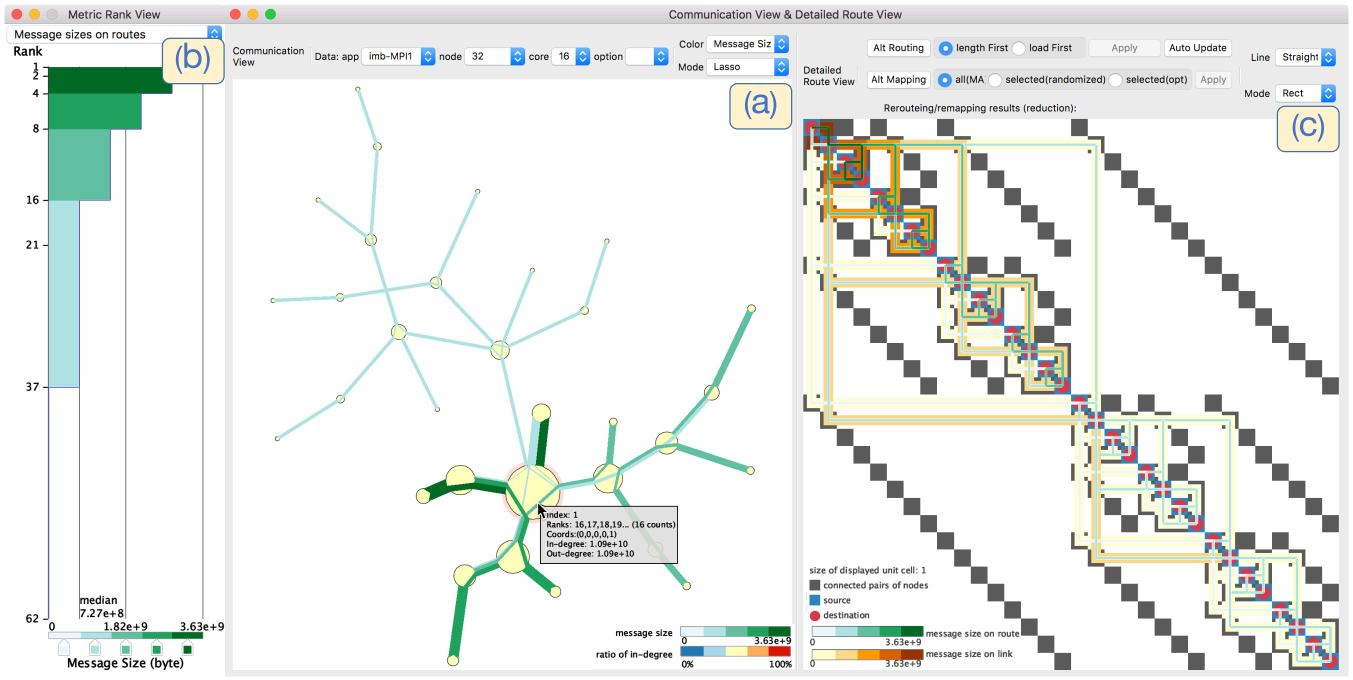
We have developed a visual analytics system for identifying and rectifying communication bottlenecks in massively parallel MPI programs (where processes communicate using message passing). These programs execute on several thousand compute nodes of supercomputers or HPC clusters. A compute node consists of 1 or more CPU cores, memory and networking components. Nodes are interconnected by a network topology such as mesh, torus, ring, etc. Fig. 1 illustrates communication flow in a supercomputer with 2D mesh interconnect and 12 compute nodes. Each node has coordinates determined by the node location and the dimensions of the interconnect. MPI ranks/processes can be placed by the job scheduler on the cores of a node. MPI rank numbers are used in the application for specifying MPI communications. The ranks are mapped onto the network using the default system mapping or a user-defined process-to-processor mapping. We consider both mappings in our work. The communication path between 2 MPI ranks is determined based on the placement of the ranks, as shown (colored paths) in Fig. 1.

A user job (application) is allocated by the job scheduler in a free partition of the supercomputer. There may be several links (hops) and nodes between two MPI ranks in an application, some of which may retard communication throughput. There are mainly two types of MPI communications – point-to-point and collective. We focus on point-to-point MPI communications, which are commonly found in many applications. This implies that there are several source-destination communication pairs. To visually identify the above communication bottlenecks, our design requirements include:

DR1 clear display of communication routes and network congestion
DR2 identification of long and congested communication routes
DR3 identification of compute nodes with high degree
DR4 alternate route suggestion to improve the performance
DR5 scalable visualization of a large number of compute nodes, network links, and communications

Our system is designed to satisfy these requirements. Fig. 2 shows visualized communications using our system. The workflow of our visual analytics process is shown in Fig. 3.

Data Collection We profile the application using performance profiling tools such as TAU [67] and HPCTW [19, 49]. We obtain


Metric Rank View

Message sizes on routes

Communication View & Detailed Route View

Data: app imb-MPI1 node 32 core 16 option

Color Message Siz

Mode Lasso

Alt Routing (● length First ○ load First) Apply Auto Update

Alt Mapping (● all(MA ○ selected(randomized) ○ selected(opt)) Apply

Line Straight

Mode Rect

Rerouting/remapping results (reduction):

index: 1
Ranks: 16,17,18,19... (16 counts)
Coords:(0,0,0,0,1)
In-degree: 1.09e+10
Out-degree: 1.09e+10

size of displayed unit cell: 1

Figure 2: The user interface of the system, which contains three views: (a) the communication view, (b) the metric rank view, and (c) the detailed route view. This example shows the communications obtained by running the MPI_Send/Recv benchmark from the Intel MPI Benchmark suite (IMB) [41] with 32 nodes, 16 cores on BG/Q. (a) shows an overview of communications between compute nodes. The route color is used for the message size in this example. (b) shows ranks and values of the selected metric from a drop-down list on the top. Here the message size on routes is selected. (c) is used for displaying the details of the routes selected in the other views.

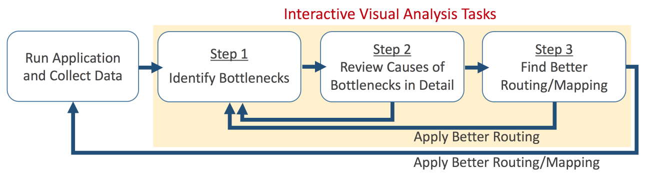

Interactive Visual Analysis Tasks
Run Application and Collect Data
Step 1
Identify Bottlenecks
Step 2
Review Causes of Bottlenecks in Detail
Step 3
Find Better Routing/Mapping
Apply Better Routing
Apply Better Routing/Mapping

Figure 3: The visual analytics process for understanding and improving parallel communications. The user analyzes the communications through the three interactive steps.

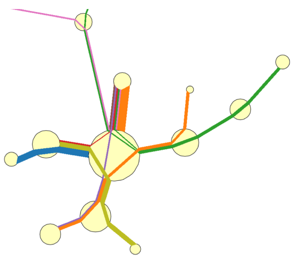


Figure 4: The visualization of routes passing through the node with high degree highlighted in Fig. 2(a). Colors are used for making each route distinguishable. Stacked polylines show routes as well as the load on links.

the source and destination MPI ranks for each communication and the corresponding message size from the profiles. The physical locations (node and core number) of the participating MPI ranks and the physical path from the source to the destination MPI rank on the supercomputer interconnect are obtained by using the routing information on the interconnect. We use a simple library [54] to obtain this information. It is difficult to collect profiling data at the user application level when the job is running. However, if such information can be extracted at run time, then we can easily extend our system to show the real-time congestion view for dynamic applications. This requires a simple script that refreshes the views whenever new data is available.

Interactive Steps The goal of our visual analytics process is to understand and improve massively parallel communications. Once the user runs their applications and collects performance data, the system can highlight the communication bottlenecks, and enables the user to review details about the bottlenecks. Our system also suggests alternate routing and mapping of processes to processors. This can be used in subsequent runs to improve overall performance.

4 Visualization Methodology

Our visual analytics system consists of three views, as shown in Fig. 2, to perform the analysis process. A user can analyze communication profiles by interactive exploration through multiple views. The three views are: (a) communication view, (b) metric rank view, and (c) detailed route view. Each view is designed for analyzing different aspects of communications and at different levels of detail. Details of properties extracted from the communications and the views are explained in the following subsections.

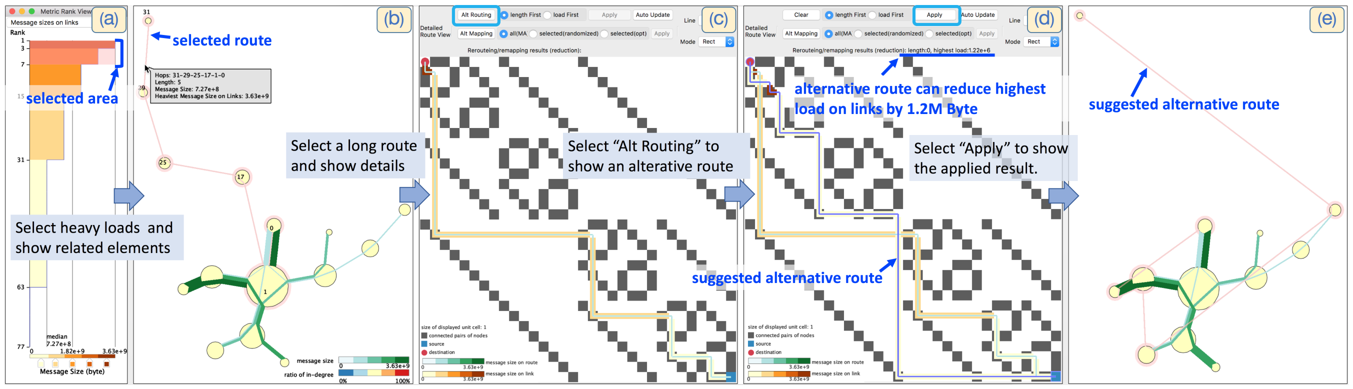
4.1 Graph Properties

We model the MPI communications as communication graphs. Our system uses graph properties, such as vertices/nodes, edges/links, and paths/routes to visualize communications. Each view conveys these properties with different representations.

Node A compute node is represented as a graph node. A node has information about an MPI rank and physical coordinates of the MPI rank in the supercomputer. The number of messages received/sent at/to an MPI rank are represented as in- and out-degree of a node.

Route A communication flow is represented as a path from a source to a destination. We call this a route in this paper. The route has information about the intermediate nodes used for the communication, length of communication (number of hops), and message size.

Link Communication between a pair of compute nodes is modeled as a graph link. The link has information about the source and


(a)
Metric Rank View
Message sizes on links
Rank
selected area
Select heavy loads and show related elements
median 7.27e+8
Message Size (byte)
(b)
selected route
Select a long route and show details
message size
ratio of in-degree
(c)
Alt Routing
Select "Alt Routing" to show an alterative route
(d)
Apply
alternative route can reduce highest load on links by 1.2M Byte
Select "Apply" to show the applied result.
suggested alternative route
(e)
suggested alternative route

Hops: 31–29–25–17–1–0
Length: 5
Message Size: 7.27e+8
Heaviest Message Size on Links: 3.63e+9

Figure 5: Interaction examples in the visual analytics process for the visualized result in Fig. 2. a) In the metric rank view, the user can see ranked link load and selects highest ranked load with the pink rectangle. b) In the communication view, the filtered results are shown. Then, the user selects one long route placed in the top left. c) In the detailed route view, the details of selected route in (b) are shown with outline colors encoding the load on links. Then, the user runs the rerouting algorithm integrated in the system. d) In the detailed route view, the suggested route is shown with a blue color. The system also shows how much the load can be reduced if the suggested route is used. e) In the communication view, the result after applying the suggested route is shown.

destination nodes of the communication as well as the load, which is the total size of all the messages that traverse on the link.

4.2 Communication View (Fig. 2(a))

The communication view shows an overview of the communications (DR1, DR2, DR3), which is mainly used for Steps 1 and 2 described in Fig. 3. A graph layout and visual encoding for each graph property are explained in the following subsections.

4.2.1 Graph Layout

Our system places nodes by using the scalable force directed placement (SFDP) [31] with communication loads as link weights. We use graph-tool [61] for calculating the layout. This method places the nodes that have more communications closer to each other. This conveys an overview of communication patterns between nodes as well as it reduces cluttered lines when the system visualizes the routes/links (DR1).

4.2.2 Node Representation

The communication view visualizes in- and out-degrees of the nodes using circles (DR3). The circle size represents the total degree (the sum of in- and out-degree). The circle color represents the ratio of in-degree to total degree. This ratio is shown using a diverging color scheme from blue to red by default, as shown in the colormap in Fig. 2(a). Additionally, detailed information including the MPI rank(s), physical coordinates, and in- and out-degree, is shown when the node is selected and hovered over with the mouse. Multiple MPI ranks on a node are shown as a comma-separated list, as shown in the mouseover text in Fig. 2(a). Using these representations, the user can identify compute nodes (and MPI ranks) with heavy communications that lead to bottlenecks as well as determine load imbalance on the links that lead to congestion. For example, in Fig. 2(a), the highlighted node in the center (marked as 'Index 1') has a high inflow and outflow of communications. This is clearly conveyed by the circle size.

4.2.3 Route and Link Representations

The route is visualized as a polyline between the source and destination nodes. Line width represents the size of the message in a communication. The line color shows the value of user-selected metric using sequential colors. Lighter colors represent smaller values and darker color represents higher values by default. The user can select one of the following metrics: message size, length of the route, or hop-bytes (i.e., the product of message size and length). Hop-bytes is a commonly used metric to determine high communication volume [12]. Fig. 2(a) uses color to show message size (shown in the drop-down list 'Color').

Visualizing the total message size between a pair of nodes (load on a link) as well as distinguishing the routes are important to identify the location and understand the cause of communication bottleneck (DR1, DR2). We use stacked polylines to achieve this. Fig. 4 shows the routes passing the highlighted node in Fig. 2(a). In this figure, we also use categorical colors to help distinguish each line. Several routes pass through the selected node as a relay point. This implies high inflow and outflow of data from the node. Since the compute nodes have a limited number of network buffers, a large number of messages passing through a node slows down all the flows through that node. Route lines are stacked on the right hand side with respect to the route direction from the center. Stacked order is based on the value of the metric selected for the line color. This helps to avoid cluttered colors between nodes and also shows total values.

Our system also provides mouse click and lasso selections for the user to select a subset of routes. Two different types of lasso selections are supported by applying similar methods as Elzen and Wijk [81]. *Within lasso* selects nodes and routes within the drawn lasso. *Within and intersecting with lasso* selects routes intersecting with the lasso and the nodes along the selected routes as well as the nodes and routes within the lasso. The second mode is useful to select nodes and routes related to a specific area (e.g., we draw a lasso around the highlighted node to show the result in Fig. 4). The user also can see detailed information of routes such as message size, link load, and route length by hovering the mouse over selected routes as shown in the mouse-over text in Fig. 5(b).

4.3 Metric Rank View (Fig. 2(b))

The metric rank view shows metrics related to communication, ranked by decreasing value. Communication bottlenecks can be caused by long hops, large message sizes, load imbalance on links, and nodes or any combination of these metrics. Therefore, to identify bottlenecks (Step 1 in Fig. 3), we designed the metric rank view for displaying length of routes, message size, and hop-bytes, as well as load on links and in-/out-degree of nodes (DR2, DR3). The user can select a metric of interest from the drop-down list located at the top. As shown in Fig. 2(b), the x-axis shows the range of values of the selected metric and the y-axis shows the ranks. Higher metric value (and thus higher rank) implies higher possibility of bottleneck. Thus, this view helps isolate bottlenecks. Colormaps are shared with the other views and the color represents the value of a selected metric. In addition, our system allows the user to change the colormap to emphasize a specific range of values by using color pickers located at the bottom. The metric rank view can also be used as an interface to filter the nodes, routes, and the links to be displayed in the other views. Fig. 5 shows an example. Nodes and routes related to the links with highest loads are indicated with the color pink in (a) and are displayed in the communication view (b).

4.4 Detailed Route View (Fig. 2(c))

The detailed route view, mainly used for Steps 2 and 3 of Fig. 3, provides information about selected routes in the other views (DR2, DR4). Fig. 5(c) shows the selected route in the communication view in Fig. 5(b). The detailed route view conveys coordinates of compute nodes in the physical topology by using an adjacency matrix.

4.4.1 Adjacency Matrix Representation

Each row and column of the adjacency matrix represents a graph node. The cell color indicates whether a pair of nodes are adjacent or not (white: not adjacent, gray: adjacent). This representation can be used to visualize connections between compute nodes in any high-dimensional and complex network topology. Furthermore, the matrix can represent the coordinates of compute nodes in the physical topology by selecting the matrix order based on the node coordinates. An example of an adjacency matrix representation for 64 compute nodes with 5D torus network topology of BG/Q [17] is shown in Fig. 6. Here, the order (shown on the right side) is based on the coordinates increasing from dimension E to A. This is generally the default order of MPI rank mapping on the physical network of machines with mesh/torus topologies. The user can change the order in case of a different mapping. The user can also specify a different order of rows and columns to observe a specific communication pattern along the dimensions. This not only helps users to identify bottlenecks but can also be useful to system developers to determine the impact of routing and mapping on communications.

4.4.2 Route and Link Representations

The matrix-based visualization is effective when the size of the graph is large (e.g., more than 20 nodes), as evaluated in [32] for various tasks. However, it is less helpful than node-link diagrams for path finding [32]. To overcome this shortcoming, we visualize routes with the adjacency matrix used in [68]. A single hop between two nodes in the route can be represented as a pair of horizontal and vertical lines in the matrix. In order to avoid occlusion in an individual route, we draw a horizontal line from the source node to the gray square first and then draw a vertical line from the gray square to the target node to show the path between the source and the target nodes. By repeating the previous step for each hop, a route can be drawn as a collection of horizontal and vertical line pairs passing through gray squares corresponding to intermediate nodes. A blue square indicates that a node is a source, and a red circle indicates that a node is a destination. In this view, the line width of the route is constant. The user can optionally show an outline on the link that encodes the load using sequential colors as shown in Fig. 6. This clearly indicates the load on each link of the route. Visualizing this route and link information together is useful for reviewing the cause of bottlenecks. For example, when two routes of large message sizes pass through the same link of high load, there is high possibility to improve the performance by rerouting one of them.

5 Rerouting and Remapping Suggestions

Fig. 4 shows a high-degree node that can impede the communications through that node. It is possible to reduce the load on the node and its links by rerouting some of the network traffic. Our system can highlight this and suggest alternative routes for the communications. We can also suggest alternative process-to-processor mapping (refer §3) that can alter the communication path and hence reduce the congestion. Our system provides rerouting and remapping suggestions

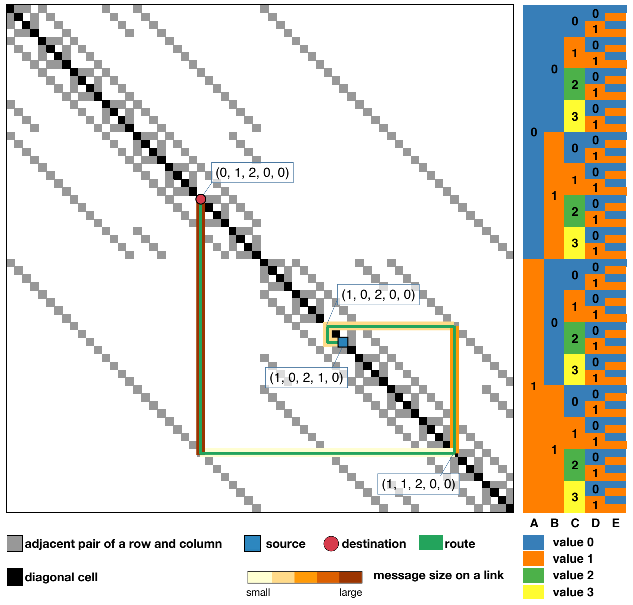


Figure 6: An example of matrix adjacency representation, and route representation for 64 compute nodes connected with 5D torus. Dimensions A, B, C, D, and E have sizes 2, 2, 4, 2, 2 respectively. The order of rows and columns is based on increasing value of coordinates starting from E and ending at A as shown in the right. The adjacency matrix is visualized by using the gray squares to show adjacency between a row and a column. The route from source (1,0,2,1,0) to destination (0,1,2,0,0) is visualized using horizontal and vertical lines passing through (1,0,2,1,0), (1,0,2,0,0), (1,1,2,0,0), and (0,1,2,0,0).

for user-selected routes as described below. These suggestions help to find better routing/mapping in Step 3 in Fig. 3 (DR4).

5.1 Rerouting Algorithms

Our rerouting algorithm for a selected route (route can be selected/filtered in the communication or metric rank view) tries to minimize 1) the length of the selected route or 2) the highest load (cumulative message size) on any single link. These two factors are critical bottlenecks for achieving high performance. The user prioritizes one of these two criteria. Let l, l', L, L', δ be the length of the selected and the alternative routes, the heaviest load on links in the selected and alternative routes, and a tolerance value that specifies a limit on the increase in length of the alternative route. Note that communication on a slightly longer route via less congested links is preferable over shorter hop communication on highly congested links. δ is 0 by default. An alternative route is found by satisfying one of the following conditions based on the selected criterion.

$$min(l') \quad s.t. \quad l' \leq l + \delta, L' \leq L \tag{1}$$

$$min(L') \quad s.t. \quad l' \leq l + \delta, L' \leq L \tag{2}$$

The algorithm finds all possible alternative routes with the same source and destination nodes as the selected route under the condition that $l' \leq l + \delta$ and $L' \leq L$ by using a breadth-first search. Then, the algorithm selects the best alternative route satisfying Equation 1 or 2 based on the selected criterion. This algorithm quickly finds a solution. For example, it takes around 30 ms for a route of length 10 on a graph with 2,048 vertices and 36,864 links on a 4 GHz Intel Core i7 processor with 16 GB memory.

The above algorithm finds the optimal solution for a single route. We developed a heuristic for multiple routes as well. Rerouting one route affects suggestions for other routes because it changes the load on multiple links. Our heuristic sorts the selected routes based on the selected criterion and then uses the above algorithm on the sorted order. This suggests better routing after the user sets the prior criterion and clicks the button ‘Alt Routing’ placed on top of the detailed route view (Fig. 2(c)). Fig. 5(d) shows a suggested alternative route (in blue) for the selected route in Fig. 5(c). The load on links in the suggested route is also displayed when the user selects the option to show it. In addition, the total reduction in route length and link load are shown at the top of the detailed route view. The system also allows the user to see the communications after the suggestions have been applied using ‘Apply’ button (refer Fig. 5(e)).

Different applications exhibit diverse communication patterns. Thus, it may be hard to implement a generic global routing algorithm that benefits all applications. Our routing suggestions recommend custom routing improvements to an application, and are supplementary to any generic system-specific improvements.

5.2 Remapping Algorithms

Optimal process mapping is an NP-hard problem [12]. A process mapping to minimize the total hop-bytes can be modeled as Quadratic Assignment Problem (QAP) under the assumption that the routes pass through one of the shortest paths between the source and the destination nodes [74]. QAP is also a well known NP-hard problem. Therefore, a heuristic or problem size reduction is required to find an approximate solution. We modified the heuristic in [74] to suggest better remapping for all processes. We also support optimal and randomized algorithms for remapping selected routes. The remapping algorithms can be applied from the detailed route view interactively. Additionally, the suggested mapping can be output as a text file that can be used for rerunning the analyzed application with the recommended mapping.

5.2.1 Problem Formulation as QAP

Here we describe the problem formulation as QAP. We introduce notations as follows. n is the number of graph nodes. A weight matrix $W = (w_{ij}) \in \mathbb{R}^{n\times n}$ is a matrix of the total bytes in a communication between graph nodes i and j. Note that a graph node can have multiple MPI ranks when we run the application on multiple cores. A distance matrix $D = (d_{uv}) \in \mathbb{R}^{n\times n}$ is a matrix of the length of the shortest path between compute nodes u and v. ϕ is a permutation representing an assignment of the graph nodes (i.e., the MPI ranks) to the compute nodes. $\phi(x)$ represents a compute node that is assigned to a graph node x. Minimizing the hop-bytes can be modeled as the following QAP: find the permutation ϕ that satisfies the objective function below.

$$\min \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} w_{ij} d_{\phi(i)\phi(j)} \tag{3}$$

5.2.2 Heuristic to Solve QAP

Several heuristic approaches have been developed for QAP, such as simulated annealing, greedy randomized adaptive search, and genetic algorithm. We use the memetic algorithm (MA) for solving QAP because it can provide better results [10, 58]. For all the above heuristics, the computation cost increases significantly as the problem size increases. In our case, when the number of compute nodes exceeds 512, the computation for remapping does not finish within reasonable amount of time (e.g., 10 minutes). Thus, for large scale problems, we use graph partitioning for reducing problem sizes, and then the heuristic can solve QAP within a few minutes even when the number of compute nodes is 4,096 (see Supplementary Document for detailed comparison of the performance). A heuristic using graph partitioning is described in Algorithm 1.

We partition the weight matrix W into m submatrices of size $p\times p$ ($p = n/m$) using a multilevel k-way partitioning [47] implemented in Metis [46]. For partitioning the distance matrix D, we use the adjacency matrix $A = (a_{uv}) \in \mathbb{R}^{n\times n}$ of the compute nodes instead of D itself. This gave us huge improvements in solution for QAP. We partition A into m submatrices, each of size $p \times p$, and then generate m submatrices of the same size from D, where each row

Algorithm 1 Remapping suggestion heuristic for all routes

```
Input: weight matrix $W$, distance matrix $D$, adjacency matrix $A$, current permutation $\phi$, the number of subgraphs $m$
 1: Partition $W$ and $A$ into $m$ subgraphs $W_s[m]$ and $A_s[m]$, respectively.
 2: Generate $m$ subgraphs $D_s[m]$ from $D$ by referring to $A_s[m]$.
 3: Generate a weight matrix $W'$ from $W_s[m]$ by calculating the total weights
    between every pair of subgraphs.
 4: Generate a distance matrix $D'$ from $D_s[m]$ by calculating the average
    distance between every pair of subgraphs.
 5: Apply MA to solve QAP whose weight and distance matrices are $W'$
    and $D'$. This produces a permutation $\phi'$.
 6: for each $w_s$ in $W_s[m]$ do
 7:     Obtain $d_s$ allocated to $w_s$ from $D_s$ by referring to $\phi'$.
 8:     Apply MA to solve QAP whose weight and distance matrices are $w_s$
        and $d_s$. This produces the permutation $\phi_s$.
 9:     Store $\phi_s$ in $\Phi_s[m]$.
10: end for
11: Generate a permutation $\psi$ for $W$ and $D$ from $\phi'$ and $\Phi_s[m]$
12: return the permutation, $\phi$ or $\psi$, that generates the minimum cost
```

and column correspond to the submatrices of A. Next, we generate matrices W' and D' whose sizes are $m \times m$, corresponding to the partitioned matrices W and D (lines 3–4). The value of each cell in W' is the sum of the message sizes passed between two sets of nodes corresponding to a pair of submatrices in W. Similarly, the value of each cell in D' is the average of the distances between two sets of nodes corresponding to a pair of submatrices in D. Next, MA is applied to solve QAP for W' and D' (line 5). This provides an assignment of the submatrices of W to D. MA is applied again to decide an assignment within the assigned submatrices. The main difference from [74] is that we use the adjacency matrix to partition the distance matrix. This ensures that D' represents a more preserved distribution of distances in D. This results in lower cost and better solution for QAP as shown in §7.2 and Supplementary Document.

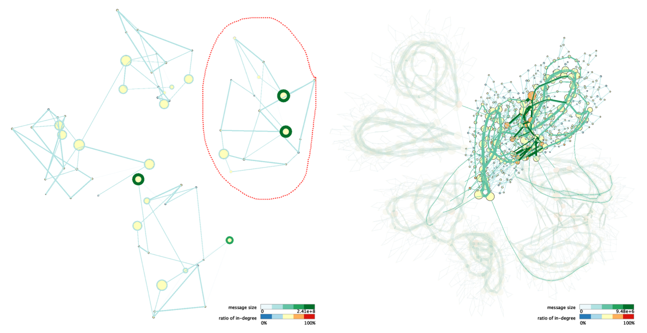
5.2.3 Optimal and Randomized Algorithms for Selected Routes

Algorithm 1 described in §5.2.2 cannot be applied selectively on specific routes. This implies that extra computation time is incurred even when the user is interested only in specific routes or nodes. Also, it cannot guarantee that specific routes of interest will be better mapped. Therefore, our system provides optimal and randomized algorithms for remapping suggestions for selected routes.

The optimal algorithm evaluates the cost of each possible remapping and selects the best one. Let n be the total number of the compute nodes and m be the number of source and destination nodes included in the selected routes. There are ${}_n\mathrm{P}_m$ alternate mappings for the selected nodes. This leads to a combinatorial increase in the number of calculations as n or m increases. Our system also provides a randomized approach for remapping selected routes to solve cases, where the number of selected routes is large, within a reasonable amount of time. Randomly generated permutations are repeatedly selected and their costs are calculated. The minimum cost permutation out of each trial is used. Performance evaluation of each algorithm is shown in Supplementary Document. Both the optimal and the randomized algorithms can also be used to refine the results obtained by applying Algorithm 1 for remapping all the routes. This coupling helps to improve the overall communication performance as well as the data transfer time along specific routes.

6 Scalable Visualization Methods

Extreme scale leadership-class systems have several thousand nodes. This impairs visualization because standard graph visualization results in clutter and we would have too few pixels to show communications in a standard matrix-based visualization. To overcome this, our system provides scalable visualization methods based on



(a) aggregate graph (b) unaggregated graph

Figure 7: (a) aggregate and (b) unaggregated graphs of communications visualized in the communication view. The communications are obtained by running MiniMD benchmark [37] on 2,048 nodes (1 core/node) on BG/Q [17]. The nodes that have the same A, B, C coordinates are aggregated in (a). In this case, the lengths of the D and E dimensions are 16 and 2. Therefore, every 32 contiguous nodes are aggregated into one node. One cluster, containing self communication loops with large message sizes, is selected with the lasso drawn in red. The nodes and routes within the selected area in (a) are shown with higher saturation colors in (b).

aggregation and folding for the communication view and the detailed route view (DR5).

6.1 Node Aggregation for the Communication View

Graph visualization of communications between compute nodes can easily generate highly cluttered results due to various factors. First, every node is within a few hops from several nodes due to low-diameter networks with high bisection bandwidths. Also, the number of communication routes is relatively large when compared to the number of nodes (e.g., running MiniAMR [37] with 2,048 nodes produces over 60,000 different communication routes). To abstract complex communications, we apply node aggregation based on their physical coordinates. A visualization of communications between 2,048 nodes on 5D torus with node aggregation is shown in Fig. 7(a). Our system aggregates nodes that have the same coordinate values in specific user selected dimensions, such as A, B, and C dimensions in this figure. The message sizes on aggregate routes are the sum of the individual message sizes between the aggregated nodes. The graph layout is calculated using the load on aggregated links. Also, an aggregate node may include communication routes among the nodes that have been aggregated together. To show this information without adding clutter, our system draws a circle around aggregated nodes that have such internal communications.

Our system also allows the user to select one or more aggregate nodes or aggregate routes to visualize without aggregation to enable the user to review details of the selected area of interest. Users can select by using mouse click or drawing a lasso around the elements of interest. Fig. 7(b) shows the visualized result of the selected elements in Fig. 7(a). The layout for the unaggregated graph is calculated by using the node positions in the aggregate graph as input to the pin and groups parameters of SFDP. These parameters place some restrictions on the placements and groupings of nodes which allows users to keep the mental map between the views as consistent as possible.

6.2 Scalable Visualization for the Detailed Route View

A critical problem with the matrix representation is its scalability. For example, if the user wants to visualize communications between 4,096 nodes, the matrix representation uses 4,096 rows and columns. At this scale, it is hard to distinguish each cell and route due to limitations in screen space and resolution. The matrix order in the detailed route view (shown in Fig. 6) stores spatial coordinates hierarchically according to dimension order. This ordering plays an important role to achieve scalable methods as described below.

6.2.1 Resolution Reduction

Our first approach is to reduce the resolution of the matrix. The visualized image should convey a summary of communications even at reduced resolution [26]. To achieve this, we use the spatial order of the mapping rule. For example, with the 5D torus network, we use the dimension mapping described in Fig. 6. Here, we used the default process-to-processor mapping order of the supercomputer to determine the matrix order in the detailed route view. However, the number of dimensions and the matrix order can be changed based on the user's environment. Our system detects the number of pixels available for a single cell based on the window size, matrix size, and zooming scale. When the number of pixels is lower than a threshold, our system reduces the resolution of the matrix by recursively aggregating rows and columns from lower dimensions. If the aggregated cell includes an adjacent pair of nodes, the source, or the destination node (a gray square, blue square, or red circle), a circle/square of corresponding color is placed on the cell. The routes are represented using the same method described in §4.4.2. Visualized results using the original resolution and a reduced resolution for the selected routes in Fig. 7(b) are shown in Fig. 8(a) and (b).

6.2.2 Magnifying Lens

With the above method, the resolution is reduced in all regions without considering user's requirement. Our system provides magnifying lens to see a specific region at higher resolution. The magnifying lens shows zoomed results within the selected square. The resolution reduction algorithm is also applied on the magnifying lens when the number of pixels available for a cell in the magnified region is lower than the threshold. An example visualization is shown in Fig. 8(c).

6.2.3 Folding Unrelated Regions

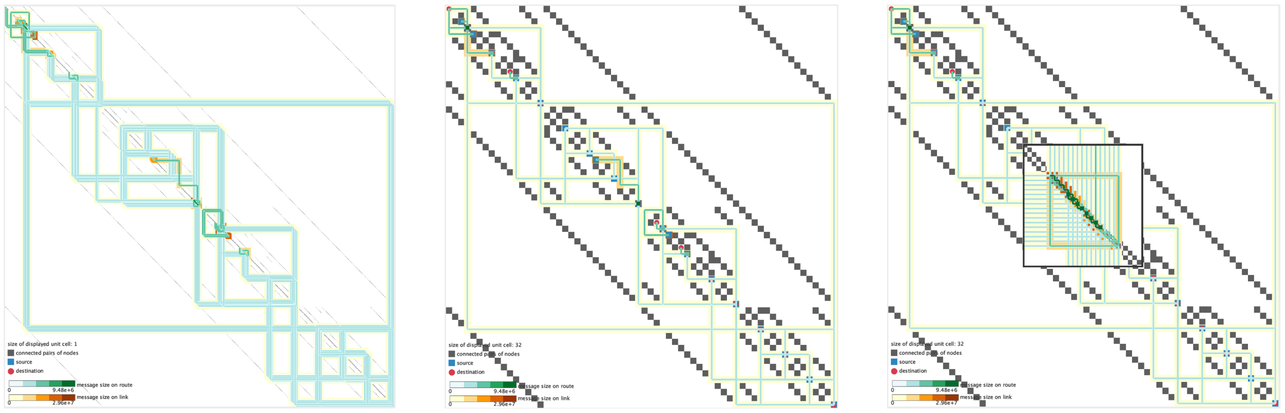
In the case of route between two nodes that are far from each other in the detailed route view, the user can hardly see every node on the route in one image. Fig. 9(a) shows an example. To address this issue that cannot be solved with resolution reduction and magnifying lens, our system supports folding regions unrelated to the routes. The user first selects a level in the matrix order, which partitions the matrix into blocks. A block is folded if no displayed route passes through it. Modern supercomputers are designed to have small diameter interconnects. Therefore, the communication routes often use a few tens of nodes. For example, the maximum diameter of BG/Q, which has 49,152 nodes, is 27. This indicates that most routes pass through less than 27 nodes. Thus, this method can work well for most routes when the user wants to display only a few at a time.

The above methods (resolution reduction, magnifying lens, and folding) can be used together based on the user's requirements.

7 Case Studies

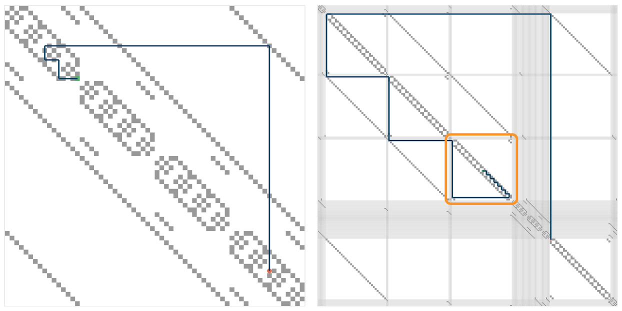
We present analysis results of visualizing communication profile data obtained from running various applications on an IBM Blue Gene/Q [17, 33] system, Mira [59], and an Intel Knights Landing [71] based Cray XC40 [20] supercomputer, Theta [78], at the Argonne National Laboratory. Mira has a 5D torus [16] network topology whereas Theta has a dragonfly topology [48]. We used low-overhead profiling tools such as TAU [67] and HPCTW [49] to collect communication profile data which was maximum of 4 GB. Processed data from these are available online[1]. We provide results for a subset of experiments due to space limitation. We demonstrate how the communication bottlenecks are identified and a better routing/mapping is found by using our visual analytics system which

[1]`https://github.com/takanori-fujiwara/par-comm-data`



(a) original result (b) with resolution reduction (c) with resolution reduction and magnifying lens

Figure 8: The comparison of visualized results of the communication routes selected in Fig. 7(a). Since the communications visualized in Fig. 7 use 2,048 nodes, each matrix cell has less than 1 pixel. In (b), the D and E dimensions are aggregated from (a) to use at least one pixel for each cell. In this example, the D and E dimensions have length of 16 and 2 so that each 32 cells are aggregated to a single cell. In (c), the magnifying lens in the center shows a higher resolution result.



(a) with resolution reduction (b) with folding unrelated regions

Figure 9: The comparison of a communication route visualized (a) without and (b) with folding unrelated regions. Resolution reduction is applied in (a). Even with magnifying lens or zooming in a specific region, the user will not be able to see the details of the route with high resolution since it spans across a wide region. After folding unrelated regions, the user can see the route in detail as indicated with an orange square (b).

was developed using C++ and Python. We were able to visualize a huge number of communications (more than 60,000) on a large number of nodes in less than 5 seconds.

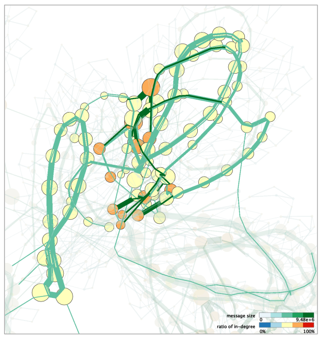
7.1 Molecular Dynamics Simulation

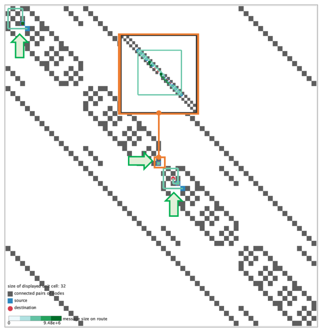
We analyze the communications of MiniMD which is a molecular dynamics application from the Mantevo Benchmarks [37]. We ran MiniMD on 32–8,192 nodes on BG/Q with 1–16 MPI ranks/node. We present an analysis example of visualization of a 0.5M atom simulation on 2,048 nodes and 1 rank/node with node aggregation, shown in Fig. 7(a). First we observe that there are several disjoint node clusters with minimum cross-communication between them. This indicates that many communications are confined to subsets of the available nodes. This is especially true for the communications in the cluster enclosed by the red lasso. Here, two of the aggregate nodes have many communications among their aggregated nodes with large message sizes (indicated by the dark green self-loops). Therefore, we select this cluster with the lasso for detailed analysis. We then inspect the selected communications without aggregation, as shown in Fig. 7(b). However, too many routes impede our analysis. Therefore, we investigate using the metric rank view, by showing only the communications that cause the top 5% highest load and thus may cause network congestion. The results are shown in Fig. 10.

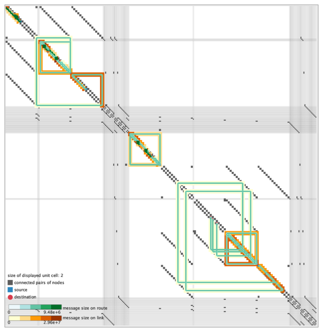
In Fig. 10(a), we see that many of these communications have long lengths and large message sizes and thus can lead to communication bottlenecks. Additionally, these communications occur in three distinct physical locations (Fig. 10(b)). We fold unrelated regions (refer §6.2.3) to see detailed routes in each location (Fig. 10(c)). Here, we note that most of the communications are along few links. This indicates that we can reduce the maximum link load by communicating along under-utilized neighboring links. We query the system for rerouting suggestions for these multiple routes, with the priority criterion of maximum load on links (refer §5.1). The suggested routes are shown in blue (Fig. 10(d)). We can see that the suggested routing spreads the traffic and better utilizes the available links. The system recommends that this routing can reduce the maximum load on 19 out of 73 routes. The expected average reduction in load on 19 routes is 18.4% and the maximum reduction is 50.1%. In addition, the system also shows potential reduction of maximum load (data size on link) by 88 MB. This is expected to vastly improve communication performance. Our rerouting suggestion also predicted an average reduction of maximum load on the links of up to 33.23% for 1–8M atom simulations on Mira with 4–32K ranks.

7.2 Adaptive Mesh Refinement (AMR)

Next, we visualize the communications in MiniAMR from the Mantevo Benchmarks [37]. It is a mini-application to study AMR codes at scale. It does 3D stencil calculation and has an irregular communication pattern with adaptation. We ran MiniAMR on 32–4,096 BG/Q nodes. The visualization for 2,048 nodes with 2 ranks/node using node aggregation is shown in Fig. 11(a). This has about 63,406 communications. From Fig. 11(a), we can see that large messages are sent within aggregated nodes. Compared to the results described in Fig. 7(a), there are no clearly disjoint clusters and there are more communication routes between aggregate nodes. Thus there are possibly many long routes, that can result in high total hop-bytes. Therefore, we next analyze the top 10% of routes with the highest hop-bytes in detail. The visualization is described in Fig. 11(b). By visualizing both the lengths (using color) and message sizes (using width), we can see that many of the routes with the largest hop-bytes tend to be long routes with small message sizes. This implies that we can reduce the total hop-bytes by remapping the MPI ranks to reduce the lengths of the communication routes. We use our heuristic for remapping all ranks as described in §5.2.2. Our experimental results on Mira showed 38.08% reduction in hop-bytes using the suggested mapping. This shows the usefulness of our system. We also found


(a) the communication view


(b) the detailed route view before folding unrelated regions


(c) the detailed route view after folding unrelated regions

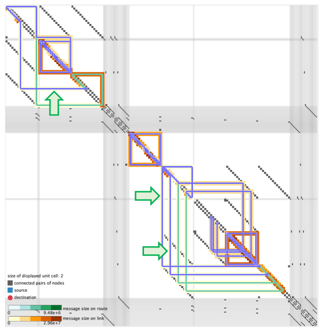
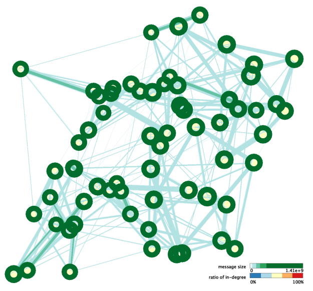

(d) routes suggested by our rerouting algorithm

Figure 10: Filtered results from Fig. 7(b). Communications along the links in the top 5% when ranked by highest load, are selected in the metric rank view. We observe in (a) that most of these communication routes have long lengths and thus may cause bottlenecks. In (b), we see that these communications occur in three different physical locations (indicated with arrows). An orange square in the middle shows a visualized result using magnifying lens. To see a detailed view of these locations, we apply folding to unrelated regions in (c). Note that many routes use the same congested links. In (d), the system suggests alternative routes (indicated with arrows) through unused links in (c).


(a) all routes colored based on their message sizes

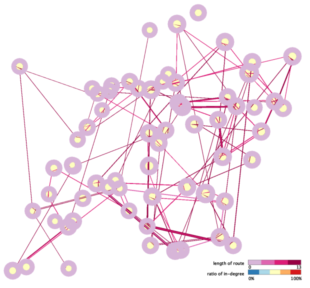
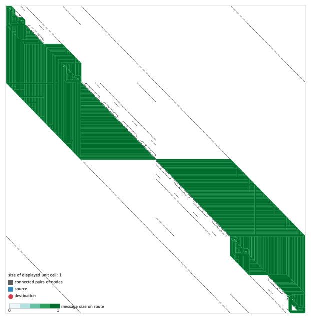

(b) filtered routes colored based on their length

Figure 11: Communications in MiniAMR. (a) communications are visualized using node aggregation. (b) shows top 10% routes with highest hop-bytes, selected in the metric rank view. Color represents route length. Many routes have large hop-bytes due to long hops. Thus, remapping can reduce communication time.


(a) default routing on BG/Q

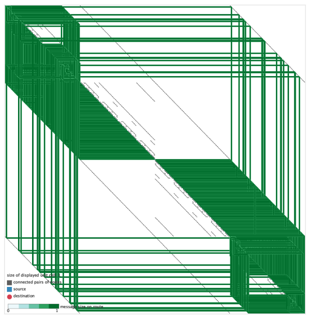

(b) with optimizations in [55]

Figure 12: Matrix MPI I/O communications with 512 compute nodes using 16 ranks/node. (a) Network traffic is localized in two regions. Approach in (b) tries to use many under-utilized links.

that there was no improvement when we used the approach in [74] instead of an adjacency matrix for partitioning the distance matrix (see Supplementary Document for other results).

7.3 Evaluation of Algorithms for MPI I/O

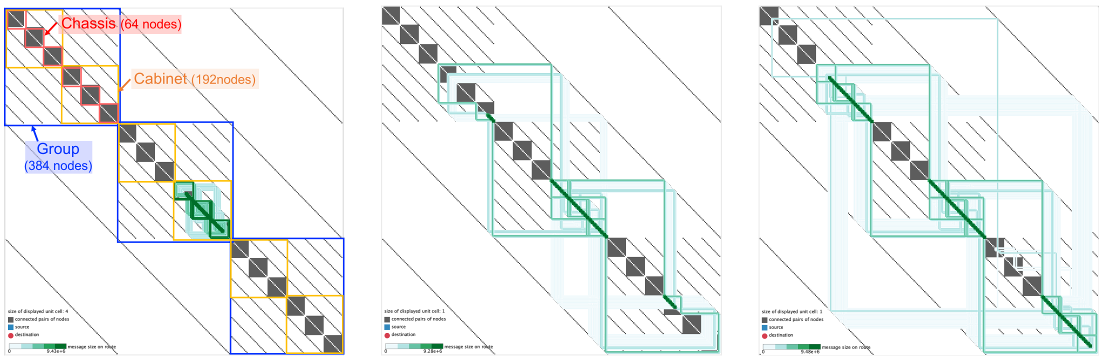
MPI I/O is used for parallel I/O from several MPI processes [75]. We demonstrate the usage of our system to evaluate optimized I/O developed in [55]. We compare the communications in default and optimized I/O on BG/Q, as shown in Fig. 12. We can see that the optimized approach (Fig. 12(b)) does better load balancing by using many unused network links in Fig. 12(a). This also shows inefficiency in the default routing. We also found from the metric rank view that the maximum load on link was reduced by 9% and the total route length was reduced by 8% in the optimized approach. This demonstrates the efficacy of our system in comparing various algorithms and routing policies.

7.4 Communications in the Dragonfly

In this case study, we analyze communications on Theta, which has a hierarchical dragonfly topology [20,48]. Four compute nodes are packaged in one compute blade. Sets of sixteen blades are packed in a chassis and sets of three chassis are mounted in a cabinet. A group comprises of two cabinets. The compute nodes, blades, chassis, and cabinets have all-to-all connections at each level. Our system is generic enough to comprehend this type of network by using a different order in the adjacency matrix representation used in the detailed rank view. The matrix order is based on the network tier, starting from the compute node level to the group level. We ran MiniMD on 32 to 1,024 nodes of Theta with 1 MPI rank/node. Visualizations of communication profiles for weak-scaling simulations with 0.3M, 0.7M, and 1.3M atoms on 128, 256, and 512 nodes respectively are shown in Fig. 13. In Fig. 13(a), we can see that all communications are in one cabinet (indicated by an orange square), and thus are likely to be efficient. On the other hand, in Fig. 13(b) and (c), our job was allocated in different chassis and groups. We note inter-group communications which may result in inter-job interference due to communications through other nodes. This can negatively affect application performance. This is more prominent in the case of 512 nodes. Thus our system enables system administrators to view the effect of job allocations on application communications.

8 Discussion and Limitations

We use node-link diagrams because they can visualize communications on compute nodes connected using any topology. This is helpful for developers who execute their applications on different supercomputers with diverse topologies. Moreover, our robust node-link diagrams can easily convey an overall communication pattern while focusing on the communication volume between compute nodes. Additionally, we used matrix-based representation to convey detailed communication patterns, unused network links, and the effect of process mappings and job allocations in the context of the underlying physical network. This is useful for system designers and



(a) 128 nodes (b) 256 nodes (c) 512 nodes

Figure 13: Communications in MiniMD on Theta, which has a dragonfly topology [20, 48]. Each matrix row and column represents a blade hosting four compute nodes by applying resolution reduction. Regions indicated by red, orange, and blue squares in (a) correspond to networks within a chassis, a cabinet, and a group. In (a), every communication passes through blades within a cabinet. In (b), communications cross three groups. Similar patterns can be also seen in (c).

system administrators. We considered other visual representations such as projections and radial layout before using the matrix-based representation. Visualization based on physical networks [44, 50] may show above information more intuitively when the number of dimensions is equal to or less than three and the topology is a regular mesh/torus. However, modern interconnects, as used in this work, have complex 5D torus or dragonfly topology. Our generic representation also works for other network topologies. While node-link diagrams with radial layouts [13, 18] can show physical locations by using node ordering, they may suffer from cluttered lines and are unable to highlight the unused network links.

We used our own library to find the communication paths based on the system routing. Any off-the-shelf library that provides this information can also be integrated with our system. We used HPCTW [49] and TAU [67] to profile; these have low overhead and are installed on both Mira and Theta. However, any profiling tool that provides the communication source, destination, and data size information can be used. One of the strong contributions of our work is recommending new communication routes along unused network links to avoid congestion, as shown in §7.1. System designers can use this to detect inefficient routing policies. Moreover, the visual analysis results of our system are not only useful for detecting communication bottlenecks but can also be used to visually evaluate different routing and mapping algorithms, as demonstrated in §7.3. Our system is also capable of visualizing MPI collective communications using the knowledge of the collective algorithm being used. We plan to incorporate this in future.

Next, we discuss the limitations of this work. In the communication view, line widths of links and routes are set to not exceed diameters of nodes. This may render very small message sizes less visible. However, in this work we focus on large message sizes which are potential causes of congestion. Also, if the user wants to distinguish routes in a link with different colors as shown in Fig. 4, the number of stacked routes in a link should be seven or less [53]. The communications are clearly visible for a few hundred nodes (see Supplementary Document). We used aggregate graph to overcome this limitation. However, node aggregation hides the detailed communications within aggregated nodes. Therefore, we support filtering and interactions described in §4 and §6. The detailed route view also has similar limitations. The graph size should be a few hundreds or less of nodes to show route/link information clearly (see Supplementary Document). The resolution reduction to give more scalability cannot show communications within a cell reduced resolution. In addition, the detailed route view suffers from clutter when the user tries to show many routes as shown in Fig. 12. In such cases, filtering, zooming, and analyzing together with the communication view are helpful. Another limitation is the intuitiveness of the route matrix representation; although user feedback suggests that this representation is effective in finding paths [68], it requires them to first learn how to interpret the visualized results.

9 Conclusions

We presented a comprehensive visual analytics process and system for optimizing massively parallel communications in supercomputers. The visualization methods used in our system are designed to comprehend large-scale and varied communication patterns on thousands of nodes with complex interconnects. The system detects hot spots in communications, allows users to select congested routes, and suggests alternative routes to reduce the maximum load on a link. We also introduce rerouting and remapping suggestions that can be coupled with the visualization. This is useful for application developers and system designers. Our system also enables system administrators to view effects of job allocations on communications.

Application developers can analyze potential communication bottlenecks on future supercomputers using our system. Our visual analysis also enables system designers to evaluate different network topologies for upcoming supercomputers. We visualized point-to-point MPI communications on BG/Q and Cray XC40 and MPI I/O communications on BG/Q. The case studies demonstrate the efficacy of the system to find bottlenecks and to improve the communication throughput. We obtained 38% improvement in hop-bytes following our system's suggestion for MiniAMR on BG/Q. In future, we will visualize tracing data which reveals temporal communication patterns. We also plan to evaluate various communication algorithms with more applications and understand the resulting performance improvements. Further, we would like to extend our system to understand communications in multiple applications on a shared interconnect. This will help the system administrators to design better job placement strategies.

Acknowledgments

This research has been sponsored in part by the U.S. National Science Foundation through grant IIS-1320229, and the U.S. Department of Energy through grants DE-SC0012610 and DE-SC0014917. This research has been funded in part and used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract no. DE-AC02-06CH11357. This work was supported in part by the DOE Office of Science, ASCR, under award numbers 57L38, 57L32, 57L11, 57K50, and 5080500.

References

[1] Characterization of the DOE Mini-apps. `http://portal.nersc.gov/project/CAL/designforward.htm`.

[2] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent. HPCToolkit: Tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience*, 22(6):685–701, 2010.

[3] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas. Blue Gene/L torus interconnection network. *IBM Journal of Research and Development*, 49:265–276, 2005.

[4] Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee. The Opportunities and Challenges of Exascale Computing. Technical report, US Department of Energy, Office of Science, 2010.

[5] Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee. Top Ten Exascale Research Challenges. Technical report, US Department of Energy, Office of Science, 2014.

[6] Y. Ajima, T. Inoue, S. Hiramoto, and T. Shimizu. Tofu: Interconnect for the K computer. *Fujitsu Sci. Tech. J*, 48(3):280–285, 2012.

[7] N. Ali, P. Carns, K. Iskra, D. Kimpe, S. Lang, R. Latham, R. Ross, L. Ward, and P. Sadayappan. Scalable I/O forwarding framework for high-performance computing systems. In *IEEE International Conference on Cluster Computing and Workshops*, 2009.

[8] G. Almási, P. Heidelberger, C. J. Archer, X. Martorell, C. C. Erway, J. E. Moreira, B. Steinmacher-Burow, and Y. Zheng. Optimization of MPI Collective Communication on BlueGene/L Systems. In *Proceedings of International Conference on Supercomputing*, pp. 253–262, 2005.

[9] P. Balaji, D. Buntinas, D. Goodell, W. Gropp, S. Kumar, E. Lusk, R. Thakur, and J. L. Träff. MPI on a Million Processors. In *Proceedings of European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pp. 20–30. Springer-Verlag, 2009.

[10] U. Benlic and J.-K. Hao. Memetic search for the quadratic assignment problem. *Expert Systems with Applications*, 42(1):584–595, 2015.

[11] A. Bhatele, T. Gamblin, S. H. Langer, P.-T. Bremer, E. W. Draeger, B. Hamann, K. E. Isaacs, A. G. Landge, J. A. Levine, V. Pascucci, et al. Mapping applications with collectives over sub-communicators on torus networks. In *High Performance Computing, Networking, Storage and Analysis*, pp. 1–11. IEEE, 2012.

[12] A. Bhatele, G. R. Gupta, L. V. Kale, and I.-H. Chung. Automated mapping of regular communication graphs on mesh interconnects. In *IEEE International Conference on High Performance Computing*, pp. 1–10, 2010.

[13] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer. Analyzing Network Health and Congestion in Dragonfly-based Supercomputers. In *IEEE International Parallel and Distributed Processing Symposium*, pp. 93–102, 2016.

[14] A. Bhatele and L. V. Kale. Heuristic-based techniques for mapping irregular communication graphs to mesh topologies. In *IEEE International Conference on High Performance Computing and Communications*, pp. 765–771, 2011.

[15] H. Bui, P. Malakar, V. Vishwanath, T. S. Munson, E.-S. Jung, A. Johnson, M. E. Papka, and J. Leigh. Improving communication throughput by multipath load balancing on Blue Gene/Q. In *IEEE International Conference on High Performance Computing*, pp. 115–124, 2015.

[16] D. Chen, N. Eisley, P. Heidelberger, S. Kumar, A. Mamidala, F. Petrini, R. Senger, Y. Sugawara, R. Walkup, B. Steinmacher-Burow, A. Choudhury, Y. Sabharwal, S. Singhal, and J. J. Parker. Looking Under the Hood of the IBM Blue Gene/Q Network. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012.

[17] D. Chen, N. A. Eisley, P. Heidelberger, R. M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. L. Satterfield, B. Steinmacher-Burow, and J. J. Parker. The IBM Blue Gene/Q Interconnection Network and Message Unit. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, 2011.

[18] S. Cheng, P. De, S. H.-C. Jiang, and K. Mueller. TorusVis$^{\mathrm{ND}}$: Unraveling High-Dimensional Torus Networks for Network Traffic Visualizations. In *IEEE Workshop on Visual Performance Analysis*, pp. 9–16, 2014.

[19] I.-H. Chung, R. Walkup, H.-F. Wen, and H. Yu. MPI Performance Analysis Tools on Blue Gene/L. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, 2006.

[20] Cray Inc. Cray XC40. `http://www.cray.com/sites/default/files/resources/CrayXC40Brochure.pdf`.

[21] J. C. de Kergommeaux, B. Stein, and P.-E. Bernard. Pajé, an interactive visualization tool for tuning multi-threaded parallel applications. *Parallel Computing*, 26(10):1253–1274, 2000.

[22] L. DeRose, B. Homer, and D. Johnson. Detecting Application Load Imbalance on High End Massively Parallel Systems. In *Proceedings of International Euro-Par Conference on Parallel Processing*, pp. 150–159. Springer-Verlag, 2007.

[23] L. DeRose, B. Homer, D. Johnson, S. Kaufmann, and H. Poxon. Cray performance analysis tools. In *Proceedings of International Workshop on Parallel Tools for High Performance Computing*, pp. 191–199. Springer Berlin Heidelberg, 2008.

[24] G. M. Draper, Y. Livnat, and R. F. Riesenfeld. A survey of radial methods for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):759–776, 2009.

[25] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks*. The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann, 1 ed., 2002.

[26] N. Elmqvist and J.-D. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2010.

[27] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. C. Sancho. A Survey and Evaluation of Topology-Agnostic Deterministic Routing Algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 23:405–425, 2012.

[28] A. Friedley, G. Bronevetsky, T. Hoefler, and A. Lumsdaine. Hybrid MPI: Efficient Message Passing for Multi-core Systems. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013.

[29] Q. Gao, X. Zhang, P.-L. P. Rau, A. A. Maciejewski, and H. J. Siegel. Performance visualization for large-scale computing systems: a literature review. In *International Conference on Human-Computer Interaction*, pp. 450–460. Springer, 2011.

[30] M. Geimer, F. Wolf, B. J. Wylie, E. Ábrahám, D. Becker, and B. Mohr. The Scalasca performance toolset architecture. *Concurrency and Computation: Practice and Experience*, 22(6):702–719, 2010.

[31] M. M. Geipel. Self-organization applied to dynamic network layout. *International Journal of Modern Physics C*, 18(10):1537–1549, 2007.

[32] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE Symposium on Information Visualization*, pp. 17–24, 2004.

[33] M. Gilge. *IBM System Blue Gene Solution: Blue Gene/Q Application Development*. IBM Redbooks, 2013.

[34] L. Grinberg, V. Morozov, D. Fedosov, J. Insley, M. Papka, K. Kumaran, and G. Karniadakis. A new computational paradigm in multiscale simulations: Application to brain blood flow. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, 2011.

[35] W. Gropp, T. Hoefler, E. Lusk, and R. Thakur. *Using Advanced MPI: Modern Features of the Message-Passing Interface*. Computer science & intelligent systems. MIT Press, 2014.

[36] S. Habib and et al. The Universe at Extreme Scale: Multi-petaflop Sky Simulation on the BG/Q. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012.

[37] M. A. Heroux, D. W. Doerfler, P. S. Crozier, J. M. Willenbring, H. C. Edwards, A. Williams, M. Rajan, E. R. Keiter, H. K. Thornquist, and R. W. Numrich. Improving Performance via Mini-applications. Technical Report SAND2009-5574, Sandia National Laboratories, 2009.

[38] T. Hoefler, E. Jeannot, and G. Mercier. An Overview of Process Mapping Techniques and Algorithms in High-Performance Computing. In *High Performance Computing on Complex Environments*, pp. 75–94.

Wiley, 2014.

[39] T. Ichimura, K. Fujita, S. Tanaka, M. Hori, M. Lalith, Y. Shizawa, and H. Kobayashi. Physics-based Urban Earthquake Simulation Enhanced by 10.7 BlnDOF $\times$ 30 K Time-step Unstructured FE Non-linear Seismic Wave Simulation. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, 2014.

[40] K. M. Iftekharuddin and M. A. Karim. Butterfly interconnection network: design of multiplier, flip-flop, and shift register. *Applied Optics*, 33:1457–1462, 1994.

[41] Intel Corporation. The Intel MPI Benchmarks 2017. `https://software.intel.com/articles/intel-mpi-benchmarks`.

[42] K. Isaacs. Living digital library of state of the art of performance visualization. `http://cgi.cs.arizona.edu/~kisaacs/STAR/`.

[43] K. E. Isaacs, A. Giménez, I. Jusufi, T. Gamblin, A. Bhatele, M. Schulz, B. Hamann, and P.-T. Bremer. State of the art of performance visualization. *IEEE EuroVis*, 2014.

[44] K. E. Isaacs, A. G. Landge, T. Gamblin, P.-T. Bremer, V. Pascucci, and B. Hamann. Exploring performance data with boxfish. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1380–1381. IEEE, 2012.

[45] P. Johnsen, M. Straka, M. Shapiro, A. Norton, and T. Galarneau. Petascale WRF Simulation of Hurricane Sandy Deployment of NCSA's Cray XE6 Blue Waters. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013.

[46] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[47] G. Karypis and V. Kumar. Parallel multilevel series k-way partitioning scheme for irregular graphs. *Siam Review*, 41(2):278–300, 1999.

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

[49] G. Lakner and et al. *IBM System Blue Gene Solution: Performance Analysis Tools*. IBM Redbooks, 2008.

[50] A. G. Landge, J. A. Levine, A. Bhatele, K. E. Isaacs, T. Gamblin, M. Schulz, S. H. Langer, P.-T. Bremer, and V. Pascucci. Visualizing network traffic to understand the performance of massively parallel simulations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2467–2476, 2012.

[51] S. Lang, P. Carns, R. Latham, R. Ross, K. Harms, and W. Allcock. I/O Performance Challenges at Leadership Scale. In *Proceedings of Conference on High Performance Computing Networking, Storage and Analysis*. ACM, 2009.

[52] C. E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, 100(10):892–901, 1985.

[53] L. W. MacDonald. Using color effectively in computer graphics. *IEEE Computer Graphics and Applications*, 19(4):20–35, 1999.

[54] P. Malakar. bgqroute library. `https://github.com/pmalakar/bgqroute/`.

[55] P. Malakar and V. Vishwanath. Data movement optimizations for independent MPI I/O on the Blue Gene/Q. *Parallel Computing*, 61:35–51, 2017.

[56] S. Markidis and E. Laure. *Solving Software Challenges for Exascale*. Springer, 2015.

[57] C. M. McCarthy, K. E. Isaacs, A. Bhatele, P.-T. Bremer, and B. Hamann. Visualizing the five-dimensional torus network of the IBM blue gene/Q. In *Proceedings of Workshop on Visual Performance Analysis*, pp. 24–27. IEEE, 2014.

[58] P. Merz and B. Freisleben. A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In *Proceedings of Congress on Evolutionary Computation*, vol. 3, pp. 2063–2070. IEEE, 1999.

[59] Argonne Leadership Computing Facility's Supercomputer Mira. `http://www.alcf.anl.gov/mira`.

[60] W. E. Nagel, A. Arnold, M. Weber, H.-C. Hoppe, and K. Solchenbach. VAMPIR: Visualization and Analysis of MPI Resources. *Supercomputer*, 12:69–80, 1996.

[61] T. P. Peixoto. graph-tool. `https://graph-tool.skewed.de/`.

[62] F. Petrini and M. Vanneschi. k-ary n-trees: high performance networks for massively parallel architectures. In *Proceedings of International Parallel Processing Symposium*, pp. 87–93, 1997.

[63] A. Randles, E. W. Draeger, T. Oppelstrup, L. Krauss, and J. A. Gunnels. Massively Parallel Models of the Human Circulatory System. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, 2015.

[64] H. Sagan. *Space-filling curves*. Springer-Verlag, 1994.

[65] Sandia National Laboratory. DUMPI. `http://sst.sandia.gov/using_dumpi.html`.

[66] J. Shalf, S. Dosanjh, and J. Morrison. Exascale Computing Technology Challenges. In *Proceedings of International Conference on High Performance Computing for Computational Science*. Springer-Verlag, 2011.

[67] S. S. Shende and A. D. Malony. The TAU parallel performance system. *International Journal of High Performance Computing Applications*, 20(2):287–311, 2006.

[68] Z. Sheny and K.-L. Ma. Path visualization for adjacency matrices. In *Proceedings of Joint Eurographics/IEEE VGTC conference on Visualization*, pp. 83–90, 2007.

[69] C. Sigovan, C. Muelder, K. L. Ma, J. Cope, K. Iskra, and R. Ross. A Visual Network Analysis Method for Large-Scale Parallel I/O Systems. In *International Symposium on Parallel and Distributed Processing*, pp. 308–319. IEEE, 2013.

[70] C. Sigovan, C. W. Muelder, and K.-L. Ma. Visualizing Large-scale Parallel Communication Traces Using a Particle Animation Technique. In *Proceedings of Eurographics Conference on Visualization*, pp. 141–150, 2013.

[71] A. Sodani, R. Gramunt, J. Corbal, H. S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. C. Liu. Knights Landing: Second-Generation Intel Xeon Phi Product. *IEEE Micro*, 36:34–46, 2016.

[72] B. V. Straalen, J. Shalf, T. Ligocki, N. Keen, and W.-S. Yang. Scalability challenges for massively parallel AMR applications. In *IEEE International Symposium on Parallel Distributed Processing*, 2009.

[73] E. Strohmaier, J. Dongarra, H. Simon, M. Meuer, and H. Meuer. The TOP500 supercomputer list. `http://www.top500.org`.

[74] C. Sudheer and A. Srinivasan. Optimization of the hop-byte metric for effective topology aware mapping. In *IEEE International Conference on High Performance Computing*, pp. 1–9, 2012.

[75] R. Thakur, W. Gropp, and E. Lusk. On Implementing MPI-IO Portably and with High Performance. In *Proceedings of Workshop on I/O in Parallel and Distributed Systems*, pp. 23–32, 1999.

[76] R. Thakur, R. Rabenseifner, and W. Gropp. Optimization of Collective Communication Operations in MPICH. *IJHPCA*, 19:49–66, 2005.

[77] L. Theisen, A. Shah, and F. Wolf. Down to earth-how to visualize traffic on high-dimensional torus networks. In *IEEE Workshop on Visual Performance Analysis*, pp. 17–23, 2014.

[78] Argonne Leadership Computing Facility's Supercomputer Theta. `http://www.alcf.anl.gov/theta`.

[79] O. Tuncer, V. J. Leung, and A. K. Coskun. Pacmap: Topology mapping of unstructured communication patterns onto non-contiguous allocations. In *Proceedings of ACM on International Conference on Supercomputing*, pp. 37–46, 2015.

[80] K. Vaidyanathan, K. Pamnany, D. D. Kalamkar, A. Heinecke, M. Smelyanskiy, J. Park, D. Kim, A. Shet, G, B. Kaul, B. Joo, and P. Dubey. Improving Communication Performance and Scalability of Native Applications on Intel Xeon Phi Coprocessor Clusters. In *International Parallel and Distributed Processing Symposium*, pp. 1083–1092, 2014.

[81] S. Van den Elzen and J. J. Van Wijk. Multivariate network exploration and presentation: From detail to overview via selections and aggregations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2310–2319, 2014.

[82] O. Zaki, E. Lusk, W. Gropp, and D. Swider. Toward Scalable Performance Visualization with Jumpshot. *International Journal of High Performance Computing Applications*, 13:277–288, 1999.

[83] C. Zhou, K. L. Summers, and T. P. Caudell. Graph visualization for the analysis of the structure and dynamics of extreme-scale supercomputers. In *Proceedings of ACM symposium on Software visualization*, pp. 143–149, 2003.