IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 31, 2018, accepted September 24, 2018, date of publication September 27, 2018, date of current version October 29, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872344

Visualizing the Topology and Data Traffic of Multi-Dimensional Torus Interconnect Networks

SHENGHUI CHENG[1], WEN ZHONG[2], KATHERINE E. ISAACS[3], AND KLAUS MUELLER[4,5], (Senior Member, IEEE)

[1]Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen 518172, China
[2]Google, Inc., Mountain View, CA 94043, USA
[3]Computer Science Department, The University of Arizona, Tucson, AZ 85721, USA
[4]Visual Analytics and Imaging Lab, Computer Science Department, Stony Brook University, Stony Brook, NY 11794, USA
[5]SUNY Korea, Incheon 119, South Korea

Corresponding author: Klaus Mueller (mueller@cs.stonybrook.edu)

This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, under the ICT Consilience Creative Program supervised by the Institute for Information & Communications Technology Promotion (IITP) under Grant IITP-2017-R0346-16-1007, in part by NSF under Grant IIS 1274 1527200, in part by the Shenzhen Peacock Plan under Grant KQTD2015033114415450, in part by the Shenzhen Fundamental Research Fund under Project ZDSYS201707251409055, and in part by The Pearl River Talent Recruitment Program Innovative and Entrepreneurial Teams in 2017 under Grant 2017ZT07X152.

ABSTRACT Torus networks are an attractive topology in supercomputing, balancing the tradeoff between network diameter and hardware costs. The nodes in a torus network are connected in a k-dimensional wrap-around mesh where each node has $2k$ neighbors. Effectively utilizing these networks can significantly decrease parallel communication overhead and in turn the time necessary to run large parallel scientific and data analysis applications. The potential gains are considerable —5-D torus networks are used in the majority of the top 10 machines in the November 2017 Graph 500 list. However, the multi-dimensionality of these networks makes it difficult for analysts to diagnose ill-formed communication patterns and poor network utilization since human spatial understanding is by and large limited to 3-Ds. We propose a method based on a space-filling Hilbert curve to linearize and embed the network into a ring structure, visualizing the data traffic as flowlines in the ring interior. We compare our method with traditional 2-D embedding techniques designed for high-dimensional data, such as MDS and RadViz, and show that they are inferior to ours in this application. As a demonstration of our approach, we visualize the data flow of a massively parallel scientific code on a 5-D torus network.

INDEX TERMS Torus, supercomputing, networks, multi-dimensional data.

I. INTRODUCTION

Supercomputing has become an indispensable tool in the big data era. The scale of these machines permits high-fidelity simulation of phenomena infeasible to enact in real physical experiments, in scientific areas such as climate, nuclear physics, astronomy, medicine, molecular dynamics, and economics. The most powerful supercomputing systems have tens of thousands multiprocessor nodes connected via dense, regular networks. Effective utilization of these resources can enable simulations to increase in scope and accuracy or decrease the time to completion and thereby allow more simulations to run [1].

Meshes and tori are common network topologies in supercomputing. In a mesh network, nodes are arranged in a rectilinear mesh, with processors connected to their Cartesian neighbors. A torus network extends the mesh by connecting nodes at opposite ends of a single dimension. These links effectively ''wrap around'' the mesh. Torus topologies are advantageous as they significantly decrease the network diameter with few added links, in turn decreasing the resources needed for parallel processes to communicate with each other. However, the large number of nodes and links, coupled with the multi-dimensional topology of the torus, makes recognizing and understanding poor network performance exceedingly challenging. This is especially true as the torus dimensions increase - while a 3-dimensional (3D) torus can be intuitively represented in 3D space by cutting its wrap around links, such a representation is not available for the newer 5-dimensional (5D) torus networks. Attempts to visualize these higher dimensional torus networks have focused on the torus structure, using small multiples, aggregation along dimensions, slicing, folding, and 3D views.

These visualizations prioritize the Cartesian structure of the network at the potential cost of obscuring other link usage patterns analysts may need to understand the resultant traffic.

Our visual interface strikes the middle ground between emphasizing torus structure and node communication. It gives analysts insight into the torus distance between links without requiring the use of slicing or 3D perspective. Specifically, we create a circular layout of nodes arranged by a space filling curve. There are several advantages to this choice. It can be applied to a torus of any dimension, requires only two drawing dimensions, maintains the symmetry of the torus, and prioritizes space for link representation. To relate this layout back to an analyst's Cartesian conceptual model, we provide indicators of the dimensions along the circumference. These features help analysts find high-level patterns of interest while maintaining the ability to view individual links. They can monitor, within a single visual interface, the network traffic, detect bottlenecks, identify better routing strategies, debug the simulation, and so forth.

The framework we propose visualizes a multidimensional torus network by unraveling the torus topology, laying it out as a cord diagram in 2D, and mapping the traffic onto the diagram's links. Specifically, our contributions are:

- We extend the flat Hilbert curve-based torus network linearization described in our previous work [2] to a hierarchical representation, enabling an exploration of the network traffic at multiple levels of detail.
- We also extend our edge bundling framework to support these hierarchical navigation capabilities
- We compare our Hilbert layout with other embedding techniques and demonstrate the connection between the torus network and the Hilbert layout.
- We augment our interface by a visual representation of the physical address space, allowing analysts to map the visualized traffic back to the node locations.
- We devise various interaction capabilities to guide users in utilizing our tool, TorusTrafficND, and demonstrate their use in several case studies with real torus networks.

Our paper is structured as folows. Section 2 provides background on torus networks and the challenges they pose for their visual understanding. Section 3 presents related work in multivariate visualization in general and for torus networks specifically. Section 4 discusses how and why standard embedding methods will not work for the problems at hand. Sections 5 and 6 describe our proposed embedding, the visuualization methods, and the operations defined on them. Section 7 presents a case study. Section 8 presents conclusions and future work.

II. BACKGROUND ON TORUS NETWORKS

A supercomputing network typically consists of nodes connected by channels with traffic. The topology of such a network can be described by an undirected graph. Suppose G is the graph/network,

$$G = (V, E, T) \tag{1}$$

in which the vertices V are the nodes of the network, the edges E are the links or channels and T are the traffic in the links. The torus is a type of network with specific constraints on V and E. In the discussion below we follow the descriptions of Nesson and Johnsson [3].

A. NODES

An n-dimensional torus network consists of K_i extents in each dimension i, where $1 \leq i \leq n$. In total, there are then $N = \prod_{i=1}^{n} K_i$ nodes [3]. Each node V_i in the torus has unique coordinates $(x_{i1}, x_{i2}, \ldots, x_{in})$, where x_{ij} is the coordinate of node V_i in each dimension, $0 \leq x_{ij} \leq K_i - 1$, $1 \leq i \leq n$.

The offsets of all nodes $V = [V_1, V_2, \cdots, V_N]'$ can be expressed in the node coordinate matrix:

$$V = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \ldots & x_{Nn} \end{bmatrix} \tag{2}$$

In practical applications one typically chooses $K_i = 2^p$ $(i = 1, 2, \ldots, n)$, where p is an integer.

B. CHANNEL

In dimension j, $1 \leq j \leq n$, node $X = (x_1, x_2, \ldots, x_n)$ connects the node

$$(x_1, x_2, \ldots, (x_j - 1)(modK_j), x_{j+1}, x_{j+2}, \ldots, x_n)$$

and the node

$$(x_1, x_2, \ldots, (x_j + 1)(modK_j), x_{j+1}, x_{j+2}, \ldots, x_n)$$

We can find a node and its neighbors by means of connectivity. Each node has two neighbors in each dimension and $2n$ neighbor nodes in total. In other words, the degree of a node in the n-dimensional torus is $2n$. Since a node of maximum offset in one dimension connects to a node of zero offset in that dimension, the distance between node $V_i[x_{i1}, x_{i2}, \ldots, x_{in}]$ and node $V_j[x_{j1}, x_{j2}, \ldots, x_{jn}]$ in the torus network can be defined:

$$Dist(V_i, V_j) = \sum_{l=1}^{n} min(|x_{il} - x_{jl}|, K_l - |x_{il} - x_{jl}|) \tag{3}$$

where x_{ij} is the coordinate of node V_i in each dimension. The channels E can be expressed as the pairwise adjacency matrix.

$$E_{ij} = Dist(V_i, V_j) \tag{4}$$

However, this adjacency matrix is quite sparse. Only when V_i and V_j are neighbors, E_{ij} is non-zero. Note that the channels are directed.

C. TRAFFIC

Packets flow through the channels. Thus the traffic across the network can be expressed as a pairwise adjacency matrix, but instead of the distances, the traffic message values are stored,

$$T_{ij} = traffic(V_i, V_j) \tag{5}$$

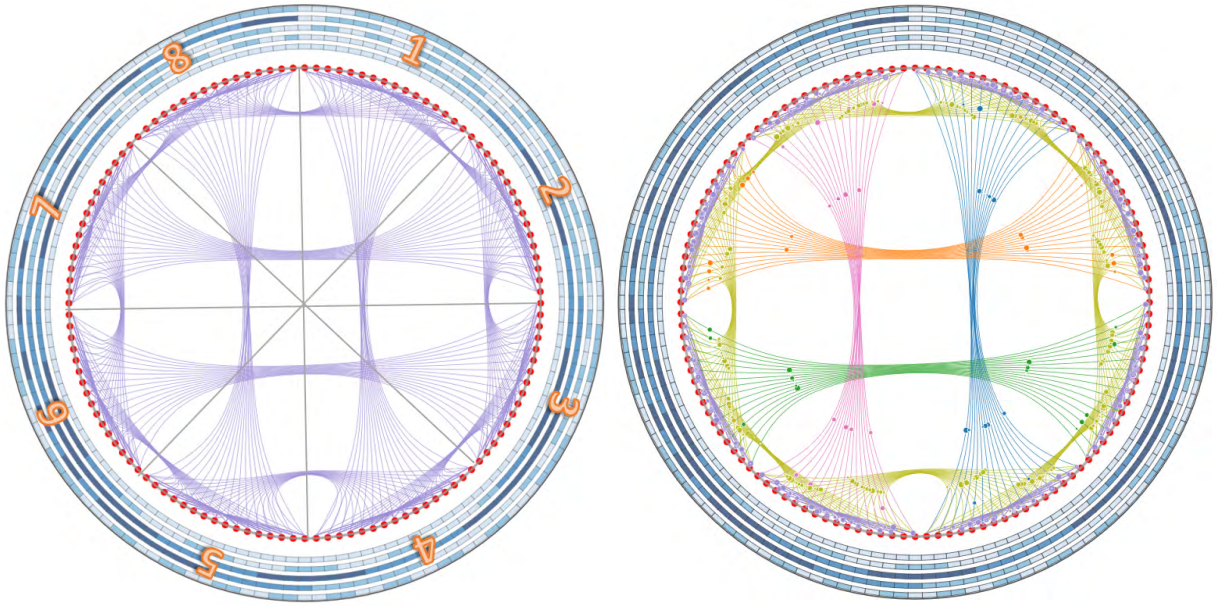


FIGURE 1. Overview of our visual interface for a 5-D torus network. Left: the complete torus network with processor nodes mapped to a ring, partitioned into 8 logical blocks defined by the Hilbert space filling curve linearization of the torus interconnect network. The lines drawn in the circle interior show the interconnections, and the five circular tracks at the disk boundary encode the physical addresses of the processor nodes. Right: The different logical scopes of the data traffic. Purple encodes interconnections (called channels) within the same Hilbert block, earth-yellow encodes channels between neighboring Hilbert blocks, while the remaining colors encode channels of remote Hilbert blocks.

The traffic can be measured using hardware performance counters. These counters offer a window to the inner workings of a system. Each counter increments upon some event, e.g., receiving a packet on a particular link. By polling the counter value and comparing it to the previous polled value, the total number of packets received via a link during the polling interval can be measured. Polling the counters of all $2n$ links per node provides a snapshot of traffic during the interval. Finer-grained information, such as timestamps and routes for individual packets or messages (groups of packets), is typically not feasible to collect.

Visualizing a multi-dimensional torus network presents the following challenges:

- **C1.** The node's address is multi-dimensional. The visualization should retain multi-dimensional similarity, especially local neighborhoods.
- **C2.** Analysts have an understanding of the torus topology and should be able to relate data to this context.
- **C3.** Nodes are connected by physical links. The visualization should retain these physical semantics.

III. RELATED WORK ON VISUALIZATION

In this section we provide a brief review of related work of visualization for multiple dimensions and then specifically for supercomputing networks.

A. MULTI-DIMENSIONAL VISUALIZATION

A torus network is a multi-dimensional structure. There are numerous visualization methods to deal with multi-dimensional data. In parallel coordinates [4], the attributes define the vertical axes while the samples form patterns of polylines. Likewise, in Radviz [5] and Generalized Barycentric Coordinate plots [6], the attributes constitute the vertices of a regular sided polygon and the samples form patterns in the polygon's interior. In all of these modalities, the axes or vertices, respectively, are placed in regular and predefined ways and do not create diagnostic patterns on their own. Biplots [7] and dynamic scatterplots [8], on the other hand, are more descriptive since the attribute axes project into the sample distribution's Principal Component Analysis (PCA) [9] basis. These plots provide some insight about element similarity in terms of the data distribution. General low-dimensional space embedding techniques, such as Multidimensional Scaling (MDS) [10], [11] and linear discriminant analysis (LDA) as used by Choo *et al.* [12], do not retain topology information, that is, one can no longer see which points are directly connected neighbors and which ones are not. Conversely, our framework maps the nodes onto a line first, and then folds them into a circle to evoke the torus connectivity. This radial organization of the linear node embedding could well preserve locality in the torus network.

Providing means for interaction through selection and brushing is important in a network visualization system [13]. Via interaction analysts can hone into network regions of interest and tune out others. Brushing often works in conjunction with linked views [14] and allows analysts to view the data in terms of different aspects. Our system provides all of these types of interactions - linking, brushing, etc.

B. NETWORK TRAFFIC VISUALIZATION

The most common depiction of a torus network is as a Cartesian mesh representation with the ''wrap around'' links either removed or shown extending into space [15]–[17]. Sometimes this view does not show links at all, instead showing the nodes only with the links understood to be between them [18]–[21], making these tools less suitable for network analysis. To mitigate the issue of occlusion in 3D mesh representations of 3D tori, Landge *et al.* [22] introduced a set of three occlusion-free 2D projections visualizing 3D tori, linked with the standard 3D view. Each such projection removes over half of the links. Higher dimensional torus networks do not permit the above representations due to the increased dimensionality required. Our approach can conceptually be applied for any number of dimensions.

Visualizations of higher dimensional torus networks typically employ slicing or aggregating along dimensions [19], [21], [23], [24], often with a small multiples view for navigation. Adjacency matrices have been used for topology-agnostic network layout [25], with Fujiwara *et al.* [26] employing them for multi-dimensional tori. They plot simulated message routes on the matrix as well as a force-directed node-link layout for examining routes through a node of interest. Instead we focus on a more global view of the data traffic. In earlier preliminary work the first and last author of this paper [2] proposed the idea of linearizing the multi-dimensional network via a Hilbert space filling curve. This provided a dimension reduction of the network into a 1D line wrapped into circle which was trivial to navigate. However, in that work we did not focus on conveying the physical address space and topology of the torus which is necessary to understand the data traffic patterns. In addition, our visual interface is specifically designed for profiles collected from network counters and the mechanisms for block mapping, bundling, and address tracking are much clearer. All in all, the system reported in this paper is much more mature.

We make use of a radial layout. Radial layouts have been employed for other types of supercomputing networks as well, including tree-based networks [27] and highly-connected hierarchical (dragonfly) networks [28], [29]. Radial layouts have also been used in other performance visualization contexts including serial program traces [30] and simulated memory traces [31]. A general overview of visualization for high performance computing systems is provided by Isaacs *et al.* [32] and one on radial layouts is provided by Draper *et al.* [33].

IV. TORUS NETWORK EMBEDDING WITH STANDARD METHODS

As stated above, there are three challenges in visualizing a multi-dimensional torus network: (1) projecting the multi-dimensions into 2D, (2) maintaining the contextual information of the analyst's understanding of the torus topology, and (3) preserving the semantics of the physical links. Our first goal is to embed the multi-dimensional torus network into 2D to resolve the challenges associated with multi-dimensionality.

For our running example, we use a 128 node 5-dimensional torus network with dimensionality (1, 4, 4, 4, 2). We first try to embed this network based on the node locations in multi-dimensional space.

A. MULTI-DIMENSIONAL NETWORK EMBEDDING

As a first attempt to visualize the network, we try Multidimensional Scaling (MDS). MDS is a popular method to embed multi-dimensional data for visualization while preserving the pairwise similarity in the multi-dimensional space. Here we have a pairwise node physical address similarity matrix S. For the embedding of the torus network, maintaining the original neighborhood structure is important. In our study, we constructed different similarity matrices and compare them as follows.

(1) We create an MDS plot of the processor nodes based on a full processor-link distance matrix. Here a distance S_{ij} would be the Manhattan distance it takes to go from V_i to V_j. See Fig. 2 (left). The nodes are colored in red while the links are shown as blue lines. We observe an embedding that has local clusters, some more emphasized as others, but spread out within each cluster.

(2) In this case we reduce the scope of nodes a given node can reach. We consider the k-nearest dimensions for each node (we use $k = 3$) and write a value of infinity into the link distance matrix for the other dimensions. Here a distance S_{ij} is the Manhattan distance between V_i and V_j bound within k-hops. This emphasizes the local neighborhood structure. Compared to Fig. 2 (middle), the structure of the torus topology is more clear.

(3) In this final case, we maximally emphasize the local neighborhood by further sparsifying the distance matrix. Here $S_{ij} = 1$ for immediately adjacent nodes and infinite otherwise. See Fig. 2 (right). It yields a sparse distance matrix in which just 8 elements of each 128-element row is less then infinity. This embedding looks the most regular of the three.

The MDS embeddings of Fig. 2 look mysterious at first but they can be easily explained. The right-most, immediate-neighbor embedding shows only 64 nodes, although there are 128 nodes in the network. This is because each red node shown really is a superposition of two neighboring nodes. MDS creates these pairs, as opposed to triplets, since any third node would not be a nearest neighbor to one of the other two nodes, resulting in a large repelling force in the MDS procedure. In fact, the red lines are also pairs of lines since they emerge from two nodes in each pair, connecting to the two nodes of a neighboring pair.

The 2-hop MDS plot in the center of of Fig. 2 also features superimposed pairs of nodes, but here the relationship to other pairs of nodes is not infinite for the nodes one hop over. This leads to the more diversified embedding. Finally, the MDS plot for the fully linked case on the right shows all 128 nodes since the link distance matrix is fully expressed.

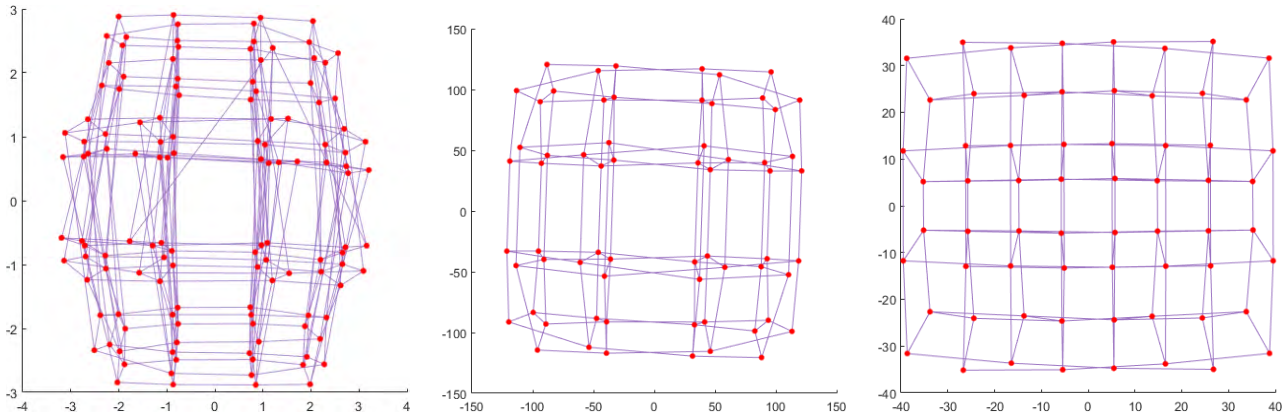


FIGURE 2. MDS embedding for different scopes in the link distance matrix (left) full, (center) three hops, and (right) immediate neighbor.

MDS's ability to embed the multi-dimensional node structure in the 2D plane meets challenge **C1**. However, even though we can explain the mappings generated (see above), relating the placement of the nodes back to the torus topology is difficult (**C2**). In other words, we cannot obtain the contextual relation - the relation between nodes and dimensions. To overcome this we propose a contextual embedding, as described in the following.

B. CONTEXTUAL VISUALIZATION - NODES AND DIMENSIONS

A prominent visualization method that can convey the relation of a set of data points to a set of dimensions is RadViz (see Section III.A). It uniformly spaces the attributes as dimensional anchors along the circumference of a circle. The location of the data points is then determined by a weighting formula where data point attributes with higher values receive a higher weight so as to increase the attraction of the point to the location of its corresponding attributes. For a node $V^* = [x_1^*, x_2^*, x_3^*, \ldots, x_n^*]$, the corresponding location P^* is:

$$P^* = \sum_{i=1}^{n} w_i v_i^* \tag{6}$$

where the weight $w_i = \frac{x_i^*}{\sum_{j=1}^{n} x_j^*}$, and v_i^* is the location of ith dimension on Radviz.

The Radviz visualization is shown in Fig. 3. The context of a torus node location is visualized via its distance from the dimension anchors on the circle. (Note that the first node, (0, 0, 0, 0, 0) is mapped directly onto the "Dim0" marker since this dimension has only one value in our example.) We notice a denser distribution of nodes in the center. This is because there is generally a higher likelihood of mid-range node addresses. Yet, despite the added context, it remains difficult to assign a concrete torus location (address) to any of the nodes in the RadViz display. The locational mapping is still too indirect for this.

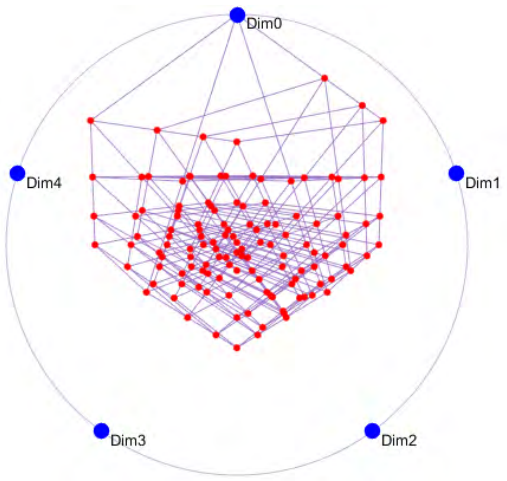


FIGURE 3. Contextual visualization of the network using Radviz.

We conclude that while standard multi-dimensional embeddings like MDS can project the network into 2D and preserve neighborhoods, they give little context about the torus topology or the physical links. Radviz gives more context about the torus topology, but not the physical semantics. This motivates the need for another visual design that can capture local behavior, torus dimensions, and physical semantics all at the same time. The next section describes such a design, which is one of the novel aspects of our paper.

V. TORUS EMBEDDING AND TRAFFIC VISUALIZATION WITH TorusTrafficND

High-dimensional spaces are naturally difficult to comprehend. In Section IV, we discussed multi-dimensional embeddings that partially address challenges **C1** and **C2**, but not **C3**–the physical semantics of the network. Conversely, our visual interface – TorusTrafficND -- preserves the neighborhood structure as well as the topology and contextual meaning. Our approach first linearizes the torus topology and then orders nodes by locality in a single dimension. When laying

out the linearized topology on a circle, the arrangement is compact and allows links to be drawn across the circle's interior. We add navigational elements to better relate this node layout to the underlying torus dimensions.

A. NODE MAPPING

To visualize the torus network, we first map the nodes from multidimensional space to the 2D plane. The layout we present here can preserve the locality and clearly visualize the group connections.

1) LOCALITY PRESERVING MAPPING

Section IV-A demonstrated a sequence of projections that increasingly emphasized the connections of a node. We note that *locality* (**C1**) in the torus can serve as a proxy for connectivity because all direct connections are local, while indirectly connected but local nodes have small distances. Furthermore, we wish to preserve the sense of regularity of the torus network. Thus, rather than using an embedding where the context of the torus is obscured, we are using a space-filling Hilbert curve to balance locality with regularity.

When mapping the nodes of the torus network onto 2D space, we wish to preserve locality as much as possible: nodes that are connected should remain close to each other. The Hilbert space-filling curve achieves better locality than the natural linear enumeration [2]. As the Hilbert curve is defined for any dimension, we can use it to linearize any high dimensional torus network.

2) BLOCK TO BLOCK CONNECTION

Based on the locality preservation, we can group some nodes as blocks. This assists researchers to capture the group behavior. Fig. 4 shows how a 2D Hilbert curve can be applied to a 2D torus (with its wrap around edges not shown) and then linearized and placed onto a radial layout.

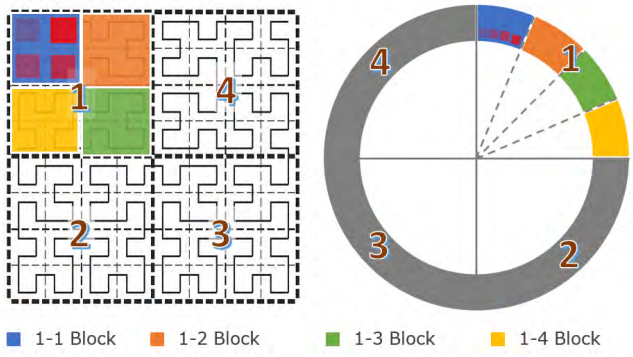


FIGURE 4. Hilbert Curve in 2D and circular layout.

The Hilbert curve starts by dividing the overall network into 4 blocks, marked **1**, **2**, **3** and **4**. These 4 blocks are laid out clockwise along the circle. The division of the network by the Hilbert curve continues recursively. Take block **1** for example; this block is divided into different sub-blocks, starting from the blue block and follows the curve lines through the orange, green, and yellow blocks. The process keeps the local neighbor structure within each block and the sub-blocks are also ordered clock-wise along the circle. The next level – the sub-sub-block level – like for example the blue block (1-1 block) is further divided into smaller blocks, as shown as red squares ordered from low opacity to high opacity. This process continues until it reaches each single node. All blocks maintain the local neighborhood structure at that level and can be tracked back and forth via the original physical semantics (**C3**). Preserving block locality and neighborhoods allows users to explore the torus network from overview to detail hierarchically since the block structure occurs at every scale of power of two (see Fig. 4).

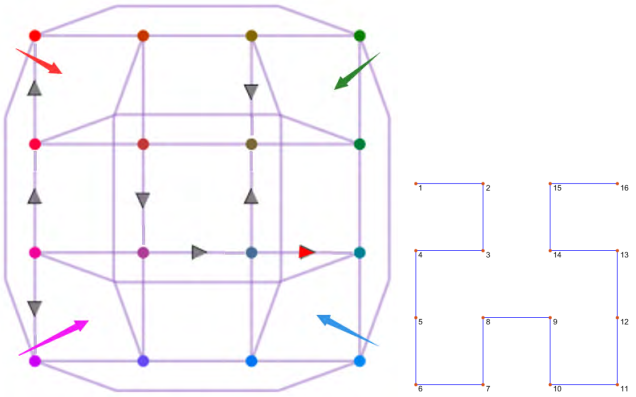


FIGURE 5. The process of 2D torus network linearizion. Left: the original 2D torus with 4 nodes at each dimension. Right: the Hilbert curve for this particular network.

As a smaller working example, we use a simple 2D torus ($k = 2$) with four nodes in each dimension. Fig. 5 (left) shows the standard node-link layout. The triangles show the direction of traffic and the nodes are colored based on their physical address. We can clearly see four different blocks: red, green, blue and purple. The color-coded arrows point to the center of each block of four. We use the Hilbert layout - Fig. 5 (right) - to linearize it, see Fig. 6. The colored arrows point to the same blocks as they do in Fig. 5 (left). All of the blocks are drawn compactly and their neighbor relationships stay the same. The nodes inside a block also maintain the original neighborhood structure.

3) PHYSICAL ADDRESS NAVIGATION

The Hilbert layout breaks the order of the original physical coordinates at block boundaries and this can make it difficult for analysts to map them to the original space. To relate the layout back to the torus dimensions, we provide k rings around the node layout disk, addressing challenge **C2**. These rings encode the offsets of the nodes in each of the k dimensions. Spans of the same value mean that the nodes of the span have the same offset within the dimension assigned to the ring, see Fig. 6. Brightness encodes the offsets, with brighter fields indicating a smaller physical coordinate value in that dimension.

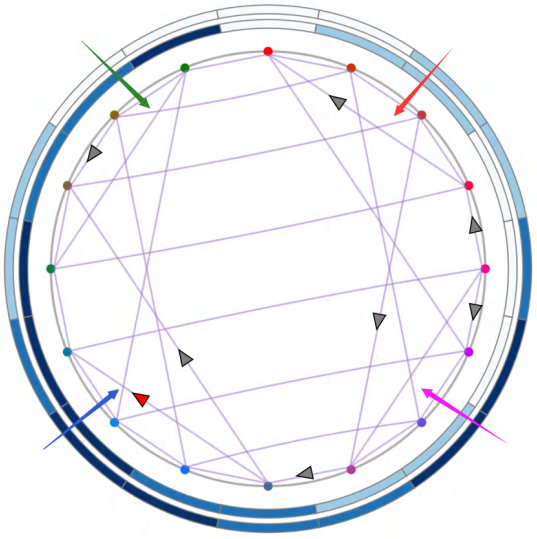


FIGURE 6. Visualizing the 2D torus network with a Hilbert layout.

We can read each dimension's physical address on the ring and relate the traffic to it, visualized as directed lines. For example, see the traffic marked with a red, filled arrow in Fig. 5. This traffic can also be observed in our visualization, marked with the same symbol, in Fig. 6. We can tell this traffic moves from node [2,2] to node [2,1] by reading the ring encodings.

B. CHANNEL BUNDLING

Channels are represented by lines and are drawn where a physical link exists between two nodes. The radial layout is well-suited to depict connections between entities, as has been recognized in other performance visualizations [30]. Another reason we layout the nodes around the circle is that it provides space to draw the links in the circle's interior.

When a large number of links needs to be shown, clutter can become a significant problem, see Fig. 7 (a). It is difficult to visually track the lines - take the green lines for example - since there are many lines crossing each other. To overcome this problem we utilize the edge-bundling technique [34]. In addition, in order to track the traffic and explore it from overview to detail, we provide an hierarchical exploration scheme.

1) EDGE BUNDLING

To reduce the clutter, we use edge-bundling based on block identity and emphasize grouping behavior among node neighborhoods. The edge bundling first takes the center of a given group of channels and then attracts the channels to the center. Suppose $E' = [e_1, e_2, \cdots, e_p]$ being a group of channels. Its center C is easy to obtain as the mean of the coordinates of E'. Then all channels of this group will be attracted by C. In order to preserve the block connections in the original torus network, we require that any group of channels E' connects two blocks consistent with the blocks imposed by the Hilbert layout (see section V-A.2). Due to the multi-scale property of the Hilbert layout the block size can vary based on the analyst's preference, exposing different granularities in the edge bundling (see section V-B.2 for a closer discussion).

Fig. 7 (b) shows the link display with edge bundling enabled. We observe that there are dense connections within each block. Compared to Fig. 7(a), the green connections become much clearer and we can easily track them by visual inspection. We also observe channels between neighboring blocks that are not close in the circular (Hilbert curve-induced) layout, but are all directly connected in the torus network. These are Block 8 and 5, Block 1 and 4, Block 2 and 7, and Block 3 and 6. Note that these connections also include the back-links of the torus network.

Based on the connections, we nominally divide the channels into intra-block channels, adjacent block channels, and non-adjacent block channels, where *adjacency* refers to the Hilbert curve induced layout (all of these blocks are adjacent in the torus network). These configurations are distinguished by different colors in the visualization. As shown in Fig. 8 (a), intra-block channels are colored purple, while adjacent neighboring blocks are colored earth-yellow. For the connections where the blocks are non-adjacent, we color the channels orange, blue, green, or pink, respectively.

a: TRAFFIC

After drawing nodes and channels in the interface, we are set to plot the traffic onto them. For the simple example described above in Fig. 6, we visualize traffic by adding an arrow to the link-line of the corresponding channel. The packets marked in red from node [2,2] to node [2,1] are clearly visible in our layout.

When the link is bidirectional we place a circle on the link with its placement determined by the node dominating the traffic (see Fig. 8 (a)). The size of the circle encodes the number of packets sent. To avoid overplotting of these marks, we add a small random factor to the locations of the mark along the channel. This ensures that the traffic marks are still close to the dominating source nodes. In this example, we see that the left channels in the blue cluster send more packets than they receive. Conversely, the yellow and orange clusters in the center have a similar number of outgoing and incoming packets.

2) HIERARCHICAL EXPLORATION SCHEME

The block-based edge bundling bears trade-offs. Having just a small number of large blocks can remove clutter in the link display effectively, but its dense edge bundles can make it difficult to visually isolate specific nodes receiving or originating data traffic. On the other hand, a large number of small blocks gives rise to thin edge bundles making it easy to determine specific traffic source or target nodes, but some traffic might get obscured due to the heavy line overdraw.

To allow users manage these trade-offs in an interactive fashion we generate a block hierarchy. The hierarchy is

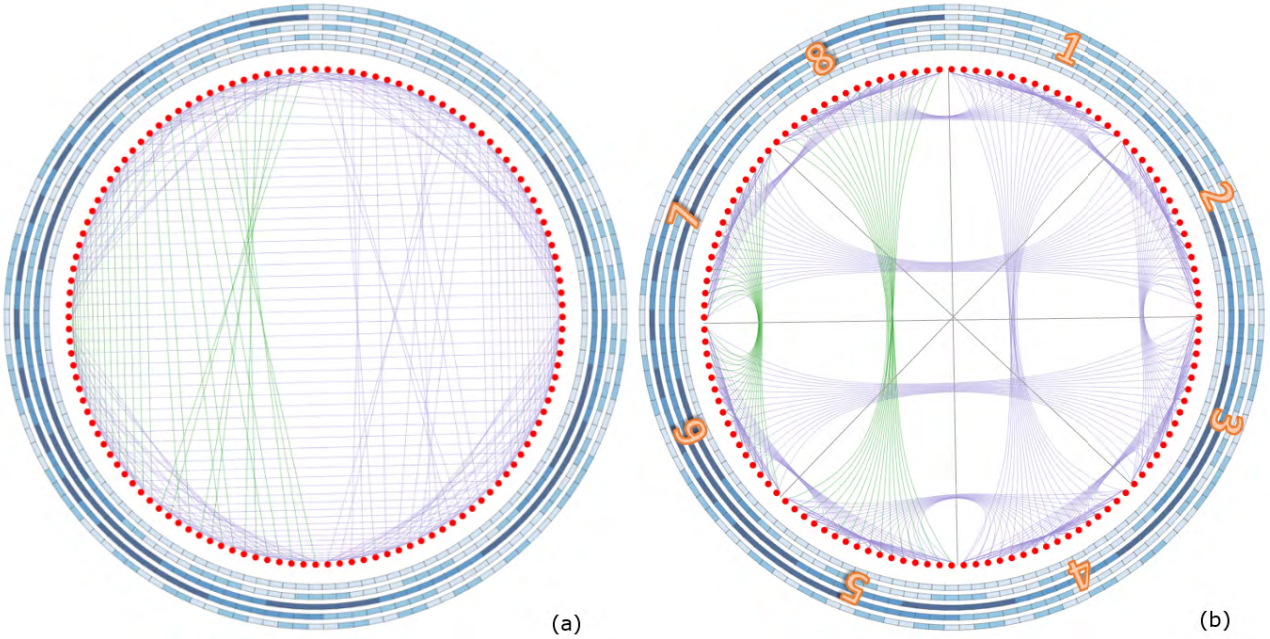


FIGURE 7. Link visualization (a) without edge bundling, and (b) with edge bundling.

formed by subdividing the Hilbert curve, creating contiguous blocks in the torus network and linear segments of nodes in the circular layout. It allows us to generate an overview (few blocks and link bundles) as well as detail views at different levels of granularity (increasingly smaller blocks and thinner more localized edge bundles).

Suppose a certain level of the hierarchy has m blocks. Then each block contains s nodes, where $s = N/m$. Here m is required to be even and $N \mod m = 0$. The nodes are divided into m different blocks as $[V_1, V_2, \cdots, V_s]$, $[V_{s+1}, V_{s+2}, \cdots, V_{2s}]$, $\cdots$, $[V_{(m-1)s+1}, V_{(m-1)s+2}, \cdots, V_{ms}]$. The channels connecting to two different blocks form a group and are bundled.

In the 128-node example shown in Fig. 8, we divided the full network into 4 (Fig. 8b), 8 (Fig. 8c) and 16 (Fig. 8d) blocks, respectively. Fig. 8(b) provides a good overview of the network traffic. We can easily see that there are two types of traffic, colorized in green and blue (the colors could denote different magnitudes of volume, or the like). We also quickly see the portions of the network the different traffic occurs. However, it is difficult to make out details of the traffic (in fact there are a small number of blue lines embedded into the green lines which are yet very difficult to pick out).

Next, the user selects the next level of the hierarchy, giving rise to the configuration of Fig. 8(c). The green line cluster is divided into two groups. While the link detail is better observed, it is still difficult to spot the traffic of the blue lines.

Finally, selecting the bottom level of the block hierarchy produces the visualization of Fig. 8(d). The green line cluster is subdivided further. Now the blue lines are clearly separated (pointed to by a black arrow), enabling the user to easily identify the corresponding nodes associated with that traffic.

VI. INTERACTIONS AVAILABLE IN TorusTrafficND

To further aid performance analysts, we developed a few more interactions with TorusTrafficND.

A. FILTER

A good way to manage the amount of detail shown is via filtering. This is particularly attractive when working with larger torus networks when the lines and marks denoting channels and traffic can become hard to distinguish. We provide filtering capabilities that allow analysts to hide some of the links. Specifically, we offer filters at two levels of detail: node and group.

The node filter focuses on a single node. When the mouse hovers over the node, only the channels connected to this node, the traffic packet marks, and the physical address in the k rings will be drawn. The other channels, traffic markers, and physical addresses will disappear. Fig. 9 left shows the results of filtering on node [0,3,2,0,1]. We observe this node has two outgoing packages - one big and one small - to [0,3,1,3,1] and [0,3,1,0,1] respectively. It is also receiving a package from node [0,3,1,3,1] and two small packages from node [0,3,1,0,1] and [0,0,2,0,1] respectively.

The group filter, on the other hand, allows analysts to focus on a group of links through brush selection. Fig. 9 right shows the results of highlighting the pink and the blue group. We observe some packages circled in red that are difficult to find in Fig. 1 (right) due to the overlapping earth-colored links.

B. ZOOM

The filter interaction can help analysts separate channels of interest from clutter, however, some details may still be

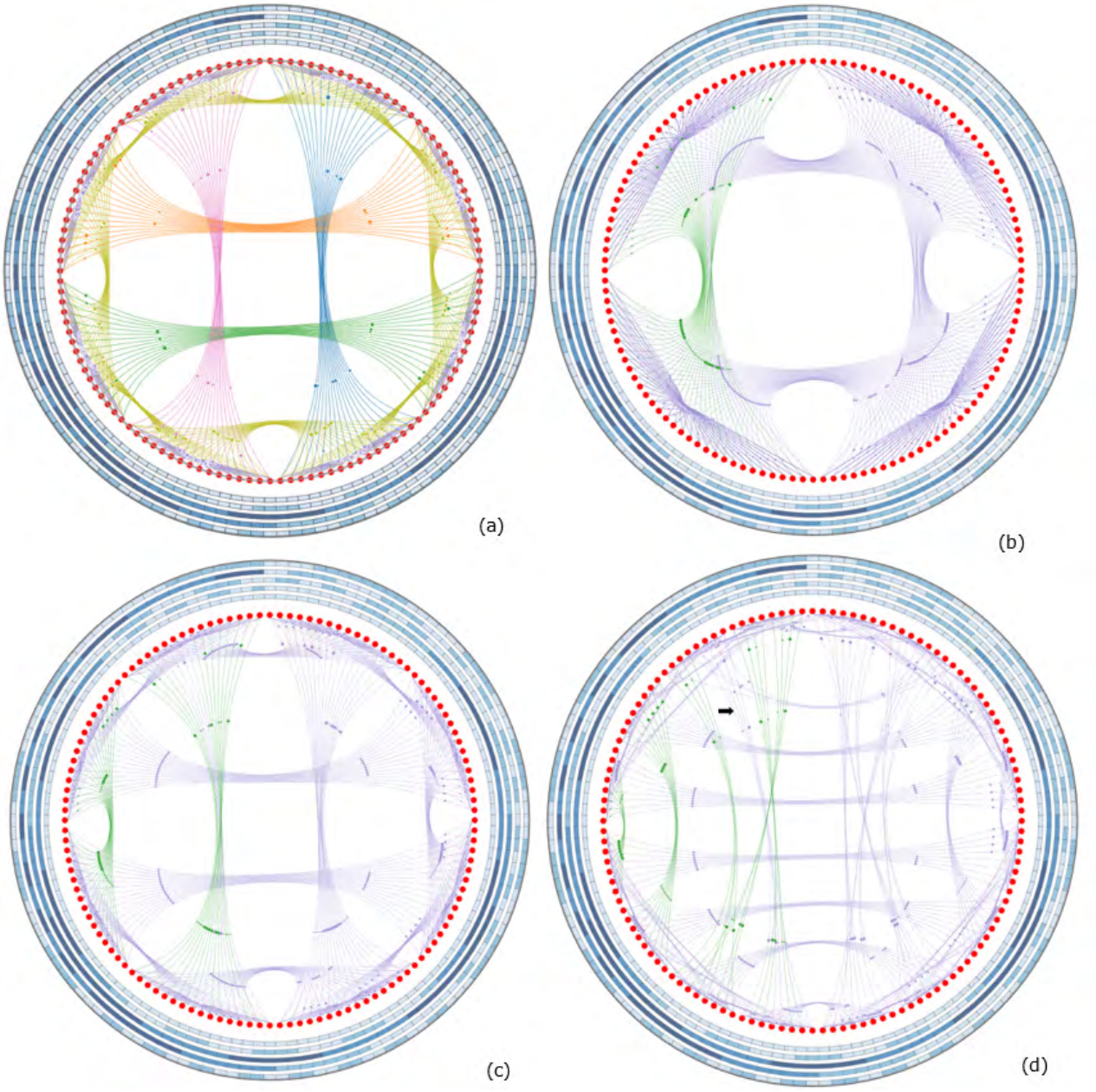


FIGURE 8. Data traffic visualization. (a) Colored by block adjacency in the Hilbert curve layout, (b-d) Different levels of block granularity: (b) 4 blocks, (c) 8 blocks, and (d) 16 blocks.

difficult to comprehend, especially the compact group traffic on the border. We thus enable users to select and zoom into an area with a rectangular brush. Fig. 11 shows a zoomed in view selected in this manner. To obtain the detail, the user draws a rectangle on top of the desired area and the corresponding zoom is created. In this example, we find some detailed traffic inside this box, see right figure. There is also a small package circled in red which is difficult to find in the original view.

C. DIMENSION SELECTION

Analysts familiar with the torus networks of supercomputers often think in terms of the physical coordinates. Therefore, we want to tie traffic information to this familiar structure with user selection. Analysts can click a value of a particular dimension on the outside ring. Then only the channels and traffic related to this dimension and meeting this threshold will remain. Fig. 10 shows the results of selecting the traffic for all channels where ''Dim2'' is three.

D. USER FEEDBACK

We tested our system with a group of 20 participants, 10 of which were active users of supercomputers, call them experts. First, we asked the participants about the network traffic by showing them the data in the conventional tabular display

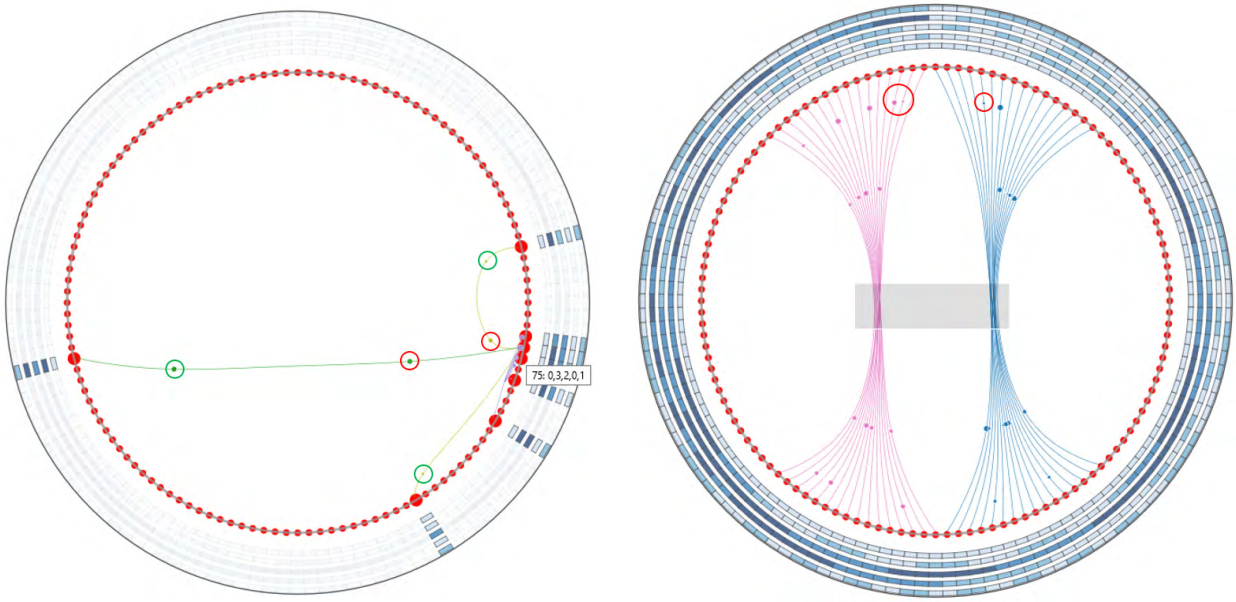


FIGURE 9. Filter in the interface. Left: node filter. Right: group filter.

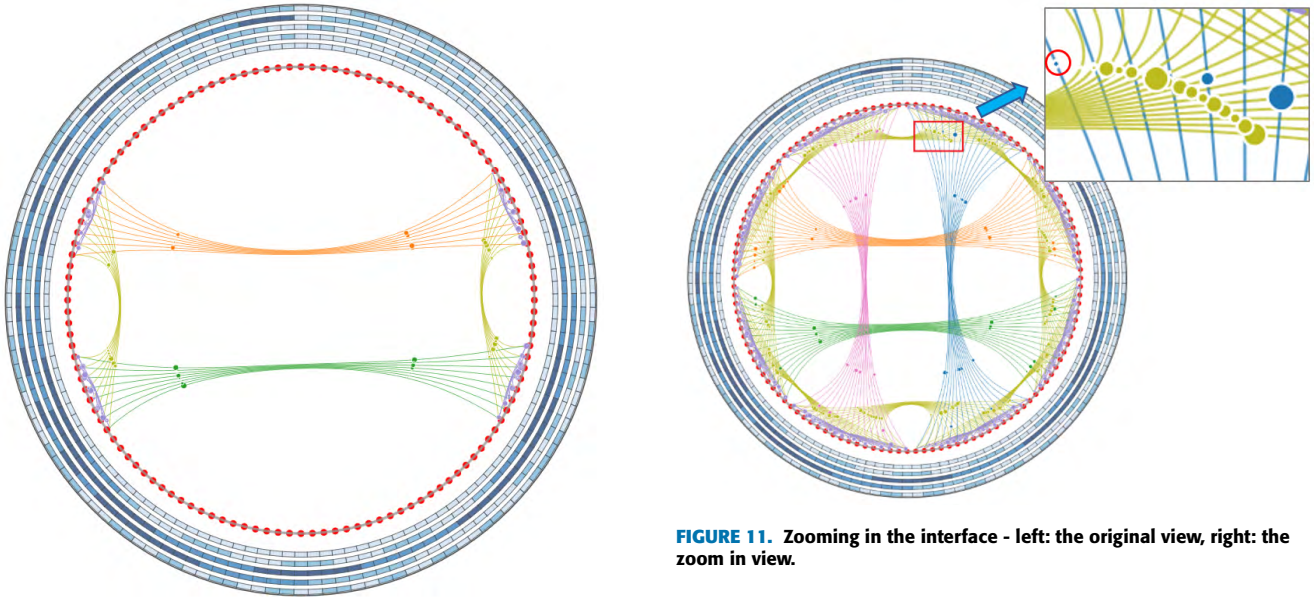


FIGURE 10. Dimension selection in the interface.

FIGURE 11. Zooming in the interface - left: the original view, right: the zoom in view.

(source, destination, and number of packets). Then we loaded these examples into our tool. We asked our participants to (1) point out locations of network traffic congestions, (2) point out which channels are free, and (3) point out which channels receive more packages than they send out.

We found that 9 of the 10 non-experts could not answer these questions or gave wrong answers when using the standard traffic statistic numbers only. But they were able to gain a good understanding of them when visualizing the network traffic via our interface, using the interactive tools it provides. This shows that our system has great value in lowering the barrier of entry into high performance computing and promoting a more effective use of these resources. On the other hand, the experts we asked were already well versed in reading the numerical statistics, but 9 of the 10 experts found that our system gave valuable additional insight and overall made network traffic analysis tasks easier and faster. For example, the experts could only identify some traffic congestions with conventional means, but with our tool they were able to discover more of these, and in a shorter time.

VII. CASE STUDY

We now use TorusTrafficND to visualize and explore network data obtained by executions of pF3D, a laser-plasma interaction code used in the National Ignition Facility, across

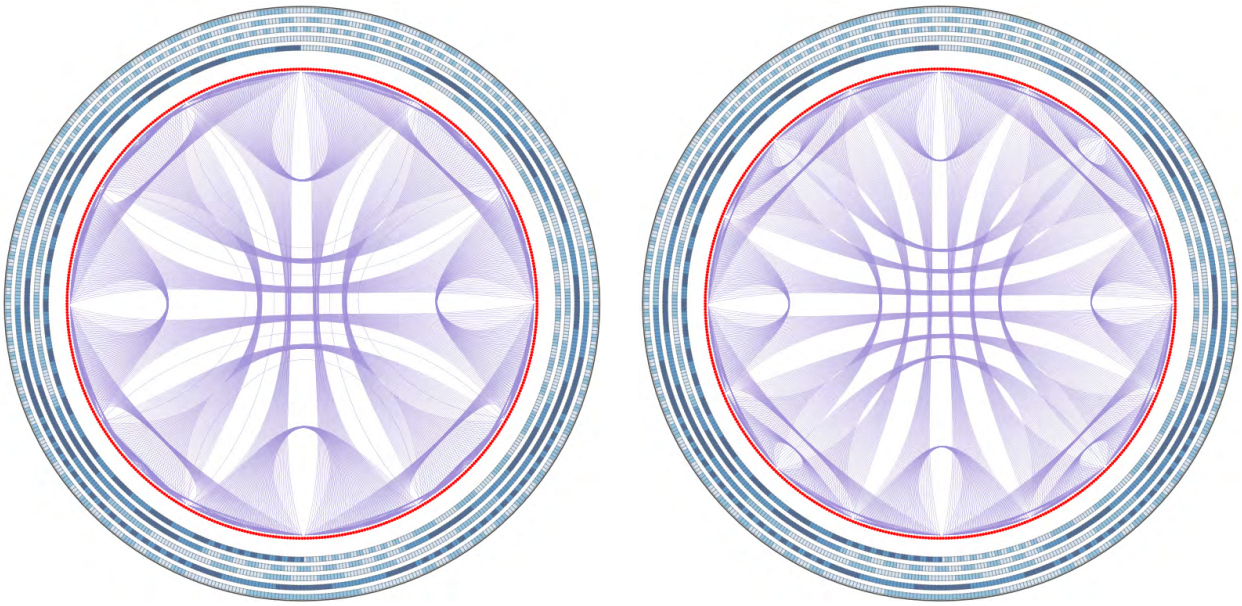


FIGURE 12. Torus network divided into 12 (left) and 16 (right) blocks, respectively, for the 4 × 4 × 4 × 4 × 2 (512 nodes) network.

multiple scales. The data was collected [23] from a real world torus network: Vulcan, a 24,576-node, five Petaflop/s IBM Blue Gene/Q (BG/Q) system at Lawrence Livermore National Laboratory. Each node in the BG/Q architecture has 16 CPU cores with the option to run 1 to 4 hardware threads per core, hence up to 64 processes may be running per node (32 in our dataset). The nodes are connected by a 5D torus network with dimensions A, B, C, D and E. Each network link permits less than a microsecond latency and offers a unidirectional link bandwidth of 2GB/s.

Few applications utilize all of the nodes on the machine. Instead, several applications may be run simultaneously. BG/Q systems aid this style of use by re-configuring the links such that each parallel application using a sufficient number of nodes has its own private torus. Traffic from other applications is not possible. Thus, when studying the behavior of one application, we can restrict our focus to a torus of the size and arrangement of one application.

The dataset contains hardware performance counters measurements obtained during execution of pF3D. In this application, the three-dimensional physical space is divided into a grid of dimensions X, Y, and Z. The cubes of the grid are allocated to the processes (one per each hardware thread used). Each process communicates with all processes that share its Y and Z coordinates (those in the X direction) and then all processes that share its X and Z coordinates (those in the Y direction). Thus, we expect a lot of traffic in the XY-planes of pF3D. By default, the pF3D cubes are assigned to each hardware thread on a single node before moving onto the next node and repeating the process. The nodes were visited in an order that incremented the E offset first, followed by the D, C, B, and A offsets respectively.

Task mapping is a technique for optimizing performance that changes the assignment of tasks (in our case, which data cube is operated on) to processes. Bhatele *et al.* [35] showed that task mapping can improve the performance of these communications. We turn to visualization to understand performance bottlenecks in the default task mapping at multiple scales.

We now use TorusTrafficND to visualize and explore network data on two differently sized torus networks. The first one contains 4 × 4 × 4 × 4 × 1 (256 nodes, 8,192 processes) and the second one contains 4 × 4 × 4 × 4 × 2 (512 nodes, 16,384 processes). The visualizations are shown in Fig. 13 left and right, respectively. Note how the outer k ring in the 256-node example is a single color as it is a 4D torus slice of a 5D torus. We illustrate how our network visualization can be used to explore the relationship between traffic and network topology.

Note that the 256 and 512 node examples, along with the 128 node example in Fig. 1 (right) are similar in shape, suggesting our visualization can scale with network size. We can also divide the network into more blocks for a more detailed grouping of channels. For the 512 nodes, we divide them into 8 (Fig. 13 right), 12 (Fig. 12 left) and 16 (Fig. 12 right) blocks respectively. The size and overall shape of the visualization are preserved.

Now we consider the traffic on these networks. In Fig. 13, we observe that both the 256-node (left) network and 512-node (right) cases have relatively unbalanced traffic—some channels have large traffic marks while others have none at all. The absence of marks means no traffic is crossing those channels, meaning the network is not being fully utilized. There is some variance in the traffic, however, with some groups sending relatively small amounts of traffic

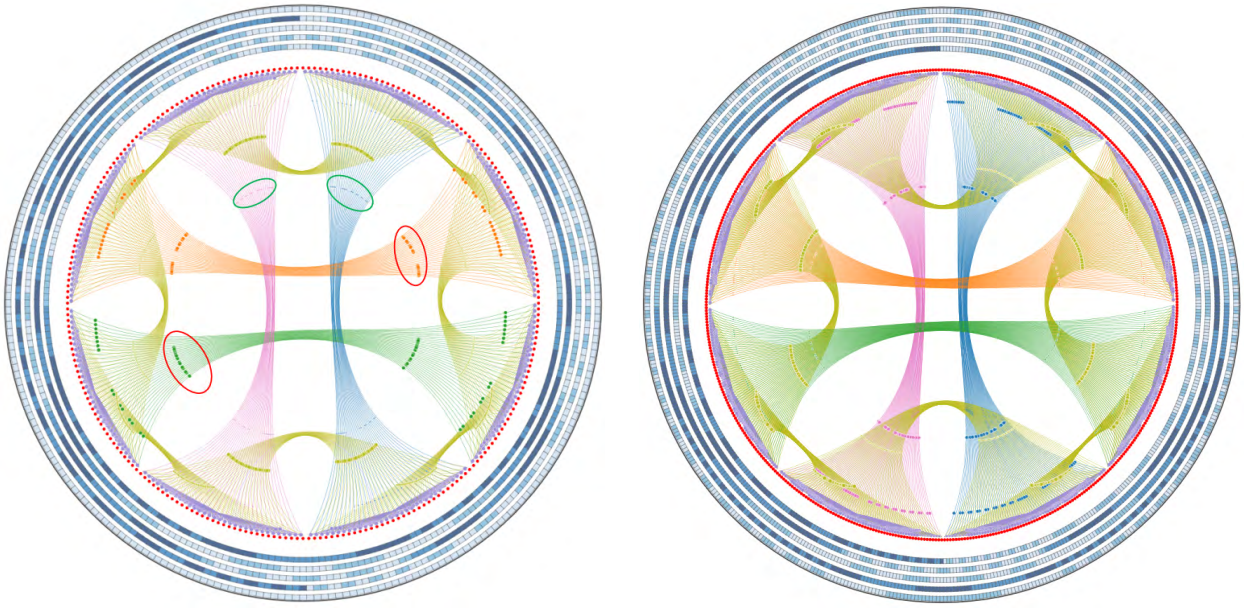


FIGURE 13. Traffic Visualization for different size torus networks divided into 8 blocks - left: 4 × 4 × 4 × 4 × 1 (256 nodes), right: 4 × 4 × 4 × 4 × 2 (512 nodes).

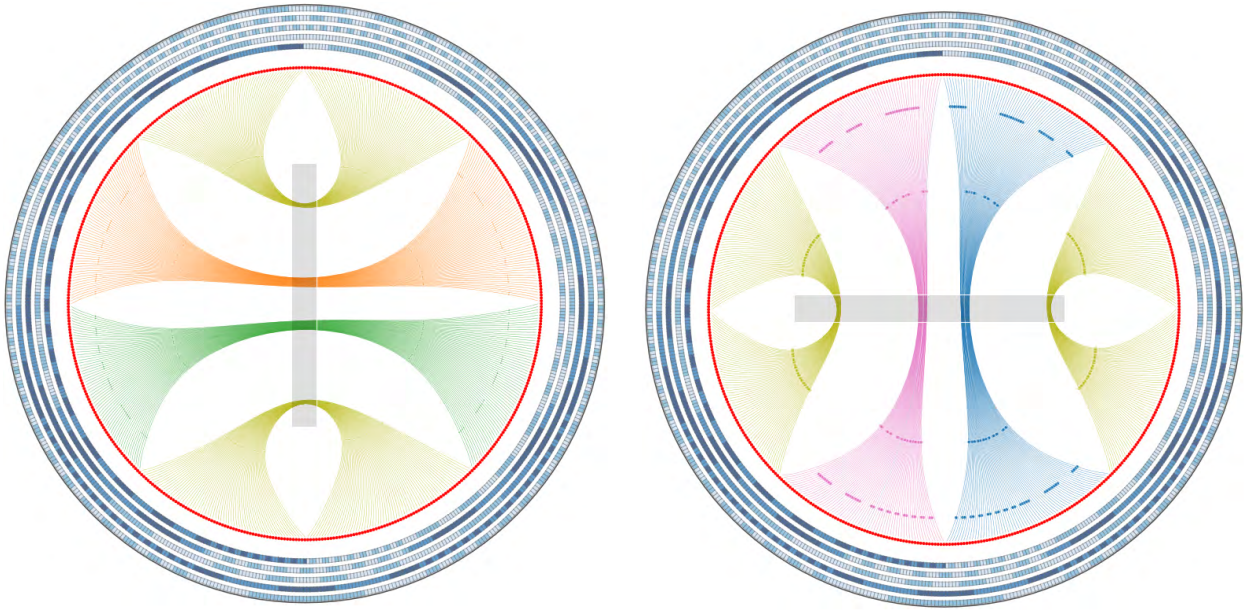


FIGURE 14. Traffic filtered horizontally (left) and vertically (right).

(circled in green) and others sending large amounts (circled in red).

To explore this potential shortcoming further, we utilize the group filter, as shown in Fig. 14 for the 512-node network. Here, the left configuration has been obtained by filtering horizontally while the right has been filtered vertically. We observe that the channels of the left configuration are barely used while those on the right carry a large amount of traffic. Referring to the k rings, we see that dimension A is being underutilized—links that vary in dimension A carry little traffic (left) while links that vary in dimension D carry a lot of traffic (right). We can see that this is due to the stark difference in value of the first and fourth k rings on either end of the grouped channels. Referring back to the 256-node visualization, we see this reliance on the D-dimension is also the case. This suggests that the default task mapping for pF3D is heavily biased to use the channels of dimension D (one of the first mapped) and that a new task mapping could be created to spread the traffic out across the links.

VIII. CONCLUSIONS AND FUTURE WORK

In the design of TorusTrafficND our goal was to provide an interactive visual tool dedicated to the exploration of data traffic in multi-dimensional torus networks. Our target was to support practical torus networks with as many as five dimensions, but our approach is not restricted to thin number. In our design we sought to address the need for better and more intuitive tools for traffic monitoring, detection of traffic bottlenecks, and aid in debugging tasks. By linearizing the multi-dimensional torus processor coordinates with a Hilbert space-filling curve we were able to overcome the challenges associated with the high dimensionality, the topological context, and the physical semantics. Mapping the Hilbert curve onto a circle and drawing the link connections into the circle's interior enables users to easily distinguish the network traffic from the processors generating it. In addition, we exploited the fractal nature of the Hilbert curve to enable a multi-resolution exploration strategy that trades overview with detail. This multiresolution viewing capability also controls the degree of edge bundling we use to provide for better visibility of the traffic in the circle interior.

The linearization of the torus to a radial layout with channels drawn across can provide structure while still showing physical semantics. The edge-bundling technique coupled with the hierarchical exploration scheme allows users to explore the traffic from overview to detail, and back. Adding physical address navigation at the circle periphery allows users to map the visualization into the context of the actual physical network. In user studies we found that the visual interface and the interactions we defined on it enabled users to make several discoveries in the data associated with real torus networks.

In the future we would like to conduct more user studies and then deploy our prototype system as a tool for use in mainstream applications of torus networks

ACKNOWLEDGMENT

(Shenghui Cheng and Wen Zhong are co-first authors.)

REFERENCES

[1] C. Xie, W. Xu, and K. Mueller, "A visual analytics framework for the detection of anomalous call stack trees in high performance computing applications," *IEEE Trans. Vis. Comput. Graphics*, to be published.

[2] S. Cheng, P. De, S. H.-C. Jiang, and K. Mueller, "TorusVisND: Unraveling high-dimensional torus networks for network traffic visualizations," in *Proc. 1st Workshop Vis. Perform. Anal. (VPA)*, Piscataway, NJ, USA, Nov. 2014, pp. 9–16.

[3] T. Nesson and S. L. Johnsson, "Romm routing on mesh and torus networks," in *Proc. 7th Annu. ACM Symp. Parallel Algorithms Archit.*, 1995, pp. 275–287.

[4] A. Inselberg and B. Dimsdale, "Parallel coordinates: A tool for visualizing multi-dimensional geometry," in *Proc. 1st IEEE Conf. Vis.*, Oct. 1990, pp. 361–378.

[5] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley, "DNA visual and analytic data mining," in *Proc. Vis.*, 1997, pp. 437–441.

[6] M. Meyer, H. Lee, A. Barr, and M. Desbrun, "Generalized barycentric coordinates on irregular polygons," *J. Graph. Tools*, vol. 7, no. 1, pp. 13–22, Nov. 2002.

[7] K. R. Gabriel, "The biplot graphic display of matrices with application to principal component analysis," *Biometrika*, vol. 58, no. 3, pp. 453–467, 1971.

[8] J. E. Nam and K. Mueller, "TripAdvisor^{N-D}: A tourism-inspired high-dimensional space exploration framework with overview and detail," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 2, pp. 291–305, Feb. 2013.

[9] I. Jolliffe, "Principal component analysis," in *Proc. Int. Encyclopedia Stat. Sci.* Springer, 2011, pp. 1094–1096.

[10] J. B. Kruskal and M. Wish, *Multidimensional Scaling*, vol. 11. Newbury Park, CA, USA: Sage, 1978.

[11] C. Xie, W. Zhong, and K. Mueller, "A visual analytics approach for categorical joint distribution reconstruction from marginal projections," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 51–60, Jan. 2017.

[12] J. Choo, S. Bohn, and H. Park, "Two-stage framework for visualization of clustered high dimensional data," in *Proc. IEEE Symp. Vis. Anal. Sci. Technol.*, Oct. 2009, pp. 67–74.

[13] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, and J.-D. Fekete, "Topology-aware navigation in large networks," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2009, pp. 2319–2328.

[14] G. M. Namata, B. Staats, L. Getoor, and B. Shneiderman, "A dual-view approach to interactive network visualization," in *Proc. 16th ACM Conf. Inf. Knowl. Manage.*, 2007, pp. 939–942.

[15] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. Comput.*, vol. 39, no. 6, pp. 775–785, Jun. 1990.

[16] R. Haynes, P. Crossno, and E. Russell, "A visualization tool for analyzing cluster performance data," in *Proc. IEEE Int. Conf. Cluster Comput.*, Oct. 2001, pp. 295–302.

[17] K. E. Isaacs, A. G. Landge, T. Gamblin, P.-T. Bremer, V. Pascucci, and B. Hamann, "Exploring performance data with boxfish," in *Proc. SC Companion High Perform. Comput., Netw., Storage Anal. (SCC)*, Nov. 2012, pp. 1380–1381.

[18] W. Spear, A. D. Malony, C. W. Lee, S. Biersdorff, and S. Shende, "An approach to creating performance visualizations in a parallel profile analysis tool," in *Proc. Eur. Conf. Parallel Process.* Springer, 2011, pp. 156–165.

[19] M. Geimer, P. Saviankou, A. Strube, Z. Szebenyi, F. Wolf, and B. J. Wylie, "Further improving the scalability of the scalasca toolset," in *Proc. Int. Workshop Appl. Parallel Comput.* Springer, 2010, pp. 463–473.

[20] O. Padron and D. Semeraro, "TorusVis: A topology data visualization tool," in *Proc. Cray User Group Meeting*, May 2014.

[21] T. Vierjahn, M.-A. Hermanns, B. Mohr, M. S. Müller, T. W. Kuhlen, and B. Hentschel, "Using directed variance to identify meaningful views in call-path performance profiles," in *Proc. 3rd Int. Workshop Vis. Perform. Anal. (VPA)*, 2016, pp. 9–16.

[22] A. G. Landge *et al.*, "Visualizing network traffic to understand the performance of massively parallel simulations," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 12, pp. 2467–2476, Dec. 2012.

[23] 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 *Proc. 1st Workshop Vis. Perform. Anal.*, Nov. 2014, pp. 24–27.

[24] L. Theisen, A. Shah, and F. Wolf, "Down to earth—How to visualize traffic on high-dimensional torus networks," in *Proc. 1st Workshop Vis. Perform. Anal.*, 2014, pp. 17–23.

[25] M. T. Heath and J. A. Etheridge, "Visualizing the performance of parallel programs," *IEEE Softw.*, vol. 8, no. 5, pp. 29–39, Sep. 1991.

[26] T. Fujiwara, P. Malakar, K. Reda, V. Vishwanath, M. E. Papka, and K.-L. Ma, "A visual analytics system for optimizing communications in massively parallel applications," in *Proc. IEEE Conf. Vis. Anal. Sci. Technol. (IEEE VAST)*, Oct. 2017.

[27] 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 *Proc. IEEE 27th Int. Symp. Parallel Distrib. Process. (IPDPS)*, May 2013, pp. 308–319.

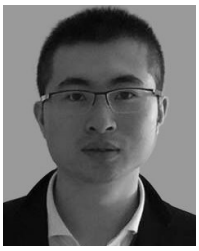
[28] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer, "Analyzing network health and congestion in dragonfly-based supercomputers," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2016, pp. 93–102.

[29] J. K. Li, M. Mubarak, R. B. Ross, C. D. Carothers, and K.-L. Ma, "Visual analytics techniques for exploring the design space of large-scale high-radix networks," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Sep. 2017, pp. 193–203.

[30] B. Cornelissen, D. Holten, A. Zaidman, L. Moonen, J. J. van Wijk, and A. van Deursen, "Understanding execution traces using massive sequence and circular bundle views," in *Proc. 15th IEEE Int. Conf. Program Comprehension (ICPC)*, Jun. 2007, pp. 49–58.

[31] A. N. M. I. Choudhury and P. Rosen, "Abstract visualization of runtime memory behavior," in *Proc. 6th IEEE Int. Workshop Vis. Softw. Understand. Anal. (VISSOFT)*, Sep. 2011, pp. 1–8.
[32] K. E. Isaacs *et al.*, "State of the art of performance visualization," in *Proc. EuroVis*, 2014.
[33] G. M. Draper, Y. Livnat, and R. F. Riesenfeld, "A survey of radial methods for information visualization," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 5, pp. 759–776, Sep. 2009.
[34] D. Holten and J. J. van Wijk, "Force-directed edge bundling for graph visualization," in *Computer Graphics Forum*, vol. 28. Hoboken, NJ, USA: Wiley, 2009, pp. 983–990.
[35] A. Bhatele *et al.*, "Optimizing the performance of parallel applications on a 5D torus via task mapping," in *Proc. 21st Int. Conf. High Perform. Comput. (HiPC)*, Dec. 2014, pp. 1–10.

SHENGHUI CHENG received the master's and Ph.D. degree in computer science from Stony Brook University, USA, in 2016 and 2018, respectively. He was a Guest Researcher at Brookhaven National Lab, USA, from 2016 to 2018, and also a Scientific Researcher at the Institute for Medical Informatics, Statistics, and Epidemiology, Leipzig University, Germany, in 2015. He is currently a Research Scientist/Fellow at the Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen. His research focuses on data visualization, visual analytics, data mining, and machine learning.



WEN ZHONG received the B.Eng. degree from the Department of Electronic Engineering and Information Science, School of Information Science and Technology, University of Science and Technology of China, in 2014, and the master's degree from Stony Brook University in 2018. He is currently a Software Engineer with Google, Inc. His research interests are visualization and machine learning.

KATHERINE E. ISAACS received the B.S. degree in computer science and the B.A. degree in mathematics from San Jose State University, the B.S. degree in physics from the California Institute of Technology, and the Ph.D. degree in computer science from the University of California at Davis, Davis, CA, USA. She is currently an Assistant Professor with The University of Arizona, researching information visualization techniques for performance analysis. In 2012, she received the Department of Energy Office of Science Graduate Fellowship.

KLAUS MUELLER received the Ph.D. degree in computer science from The Ohio State University. He is currently a Professor of computer science at Stony Brook University and a Senior Adjunct Scientist at the Brookhaven National Lab. He has authored over 170 papers which were cited over 8100 times. His current research interests include visualization, visual analytics, data science, and medical imaging. He received the U.S. National Science Foundation Early CAREER Award in 2001, the SUNY Chancellor's Award for Excellence in Scholarship and Creative Activity in 2011, and the IEEE CS Meritorious Service Certificate in 2016. He is currently the Associate Editor-in-Chief of the IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS.