Visual Informatics 2 (2018) 98–110

Contents lists available at ScienceDirect

Visual Informatics

journal homepage: www.elsevier.com/locate/visinf

A visual analytics system for optimizing the performance of large-scale networks in supercomputing systems

Takanori Fujiwara[a,*], Jianping Kelvin Li[a], Misbah Mubarak[b], Caitlin Ross[c], Christopher D. Carothers[c], Robert B. Ross[b], Kwan-Liu Ma[a]

[a] University of California, Davis, United States
[b] Argonne National Laboratory, United States
[c] Rensselaer Polytechnic Institute, United States

ARTICLE INFO

Article history:
Received 11 December 2017
Received in final form 23 February 2018
Accepted 12 March 2018

Keywords:
Supercomputing
Parallel communication network
Dragonfly networks
Time-series data
Performance analysis
Visual analytics

ABSTRACT

The overall efficiency of an extreme-scale supercomputer largely relies on the performance of its network interconnects. Several of the state of the art supercomputers use networks based on the increasingly popular Dragonfly topology. It is crucial to study the behavior and performance of different parallel applications running on Dragonfly networks in order to make optimal system configurations and design choices, such as job scheduling and routing strategies. However, in order to study these temporal network behavior, we would need a tool to analyze and correlate numerous sets of multivariate time-series data collected from the Dragonfly's multi-level hierarchies. This paper presents such a tool–a visual analytics system–that uses the Dragonfly network to investigate the temporal behavior and optimize the communication performance of a supercomputer. We coupled interactive visualization with time-series analysis methods to help reveal hidden patterns in the network behavior with respect to different parallel applications and system configurations. Our system also provides multiple coordinated views for connecting behaviors observed at different levels of the network hierarchies, which effectively helps visual analysis tasks. We demonstrate the effectiveness of the system with a set of case studies. Our system and findings can not only help improve the communication performance of supercomputing applications, but also the network performance of next-generation supercomputers.

1. Introduction

The advancement of supercomputing technology is crucial to many scientific studies and engineering designs because these studies and designs increasingly rely on large scale simulations. A powerful, massively-parallel supercomputer enables scientists to employ more sophisticated models to simulate complex phenomena or processes at a greater level of detail and accuracy. While the attempt to build the fastest supercomputers has been very actively pursued (Strohmaier et al.), it is also important to understand how to fully utilize the potential of a supercomputer. High-radix, low-diameter, hierarchical networks based on the Dragonfly topology are popular choices for building modern and next-generation high performance computing (HPC) systems. Such hierarchical networks effectively connect over ten thousand compute nodes for large-scale distributed and parallel computing. At U.S. Department of Energy's National Laboratories, several new systems (e.g., Cori (NERSC, 2016b) at NERSC, Trinity (Los Alamos National Laboratory, 2016) at Los Alamos/Sandia, and Theta (Argonne Leadership Computing Facility, b) at Argonne) deploy Dragonfly-based networks (Kim et al., 2008a) with some variations. To maximize the efficiency of these systems, effective methods and tools are needed for analyzing and studying the behaviors and performance of these networks. Random or adaptive routing is usually used with these networks in order to reduce network con-

*Corresponding author.
e-mail: tfujiwara@ucdavis.edu (Takanori Fujiwara)

https://doi.org/10.1016/j.visinf.2018.04.010

gestion and improve system performance (Jain et al., 2014). However, hierarchies in the network coupled with adaptive routing make performance analysis a challenging task. Conventional performance analysis tools cannot provide the sufficient support for analysis and exploration of large-scale hierarchical, high performance communication networks.

Most of the previous studies on Dragonfly-based networks (Jain et al., 2014; Jiang et al., 2009; Won et al., 2015; Yang et al., 2016) focus on analyzing the structural characteristics of network performance using statistical analysis. However, they do not analyze the temporal behavior of the network, which is also important for gaining insights on optimizing application performance and improving network design.

In this paper, we present a visual analytics system that we have developed for understanding the complex temporal behaviors of Dragonfly-based networks. The strength of the system is to support interactive analysis of large scale multivariate time-series that are collected from different types of network entities (e.g., network links and terminals). The basis of the support is to provide a set of time-series clustering methods for analyzing different performance metrics and variables, and in addition, integrate these methods with interactive visualizations for exploring the behavior of complex, hierarchical, high performance communication networks. In addition, by utilizing a time-series segmentation method, our system provides succinct summaries of network traffic from long time-series. The combinations of these analysis methods, visualizations, and interactions allows the user to understand the behavior of complicated networks. We demonstrate the effectiveness of our system with several case studies, in which we analyze the network behaviors collected from the simulation of two parallel applications on Theta (Argonne Leadership Computing Facility, b)—the supercomputer operated at the U.S. Argonne National Laboratory. We show that the visual analysis capability of our system leads to a better understanding of complex temporal behaviors caused by different communication patterns on the Dragonfly-based networks, as well as identifying performance bottlenecks.

2. Background and Related Work

Our visual analytics system is designed to visualize and analyze the temporal behaviors and performance of large scale networks. We mainly consider supercomputers with hierarchical structures, such as different variations of the Dragonfly-based networks. In this section, we briefly introduce the Dragonfly networks and describe the related work on visual analytics.

2.1. Large-Scale Hierarchical Networks for HPC

Multi-level, fully connected networks, such as the Dragonfly topology (Kim et al., 2008b), provide large bisection bandwidth and a low network diameter to efficiently connect more than ten thousands compute nodes. Such networks are promising options for building exascale systems. Fig. 1 shows an example of the Dragonfly network using the Cray XC (Cray Inc.) configuration. It is a hierarchical topology composed of multiple groups that are fully connected by all-to-all global links. Each

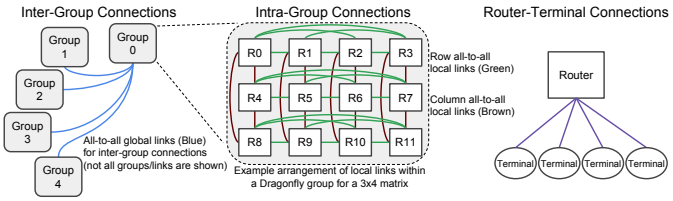


Fig. 1: The Dragonfly configuration used by Theta. All groups are fully connected by global links to form a two-level network (left). The routers in each group are also fully connected by local links (center). Each router is connected to multiple terminals (right).

group has routers arranged in rows and columns, which are connected by green and brown local links, respectively. Each router is connected to multiple terminals. Dragonfly-based networks typically use adaptive routing to reduce network congestion. When transferring packets between terminals in two different groups with no network congestion detected, adaptive routing uses the global links that directly connect the two groups. When network congestion is detected, adaptive routing redirects packets to a randomly selected group and then forwards the packets to the destination group.

2.2. Visual Analytics for Exploring Network Performance Data

Several analysis and visualization tools have been developed in order to improve the performance of parallel applications. Isaacs et al. (2014b) provide a comprehensive survey of performance visualizations. General-purpose performance tools, such as CrayPat (DeRose et al., 2007), HPCToolkit (Adhianto et al., 2010), Scalasca (Geimer et al., 2010), Vampir (Nagel et al., 1996), and TAU (Shende and Malony, 2006) can provide graphical results for analyzing network performance. However, the visualization methods used in these tools are not designed for exploration of large hierarchical networks. For example, they often use stacked timelines to show the performance of each application process without scalable methods (e.g., aggregation). In addition, while they do visualize the performance data with physical locations of the processes, they do not relate the network metrics (e.g., traffics) to the physical network topologies. Therefore, these tools lack either the scalability needed for exploring large-scale networks or the capability to analyze the complexity of hierarchical networks.

Researchers have developed visualizations for large and complex networks. Current supercomputers often have low-diameter interconnects enabling fast communications, for example, the Dragonfly (Kim et al., 2008a), Slim Fly (Wolfe et al., 2016), and fat tree (Leiserson, 1985) topologies. Several studies focus on visualizing the physical node locations in these complex networks. Landge et al. (2012) projected 3D torus networks on both 2D and 3D views in order to analyze network traffics with the topological properties. McCarthy et al. (2014) extended this 3D to 2D projection method for visualizing the 5D torus. Cheng et al. (2014) developed TorusVisND which can be applied on any high-dimensional torus networks by utilizing space-filling curve (Sagan, 1994) with a radial layout. In an approach used by Sigovan et al. (2013a), the specialized I/O communication network is preserved in a radial node-link approach,

while the communication patterns are visualized as heatmaps along each edge which show some property such as latency, message size, etc. Bhatele et al. (2016) analyzed Dragonfly-based networks by using a radial layout and a matrix view to show inter-group and intra-group links between the compute nodes. Fujiwara et al. (2017) utilized node-link diagrams and the matrix-based representations with hierarchical aggregation techniques to visualize any type of network topologies. They also provided algorithms for suggesting better routing and mapping, which can be used interactively. Li et al. (2017) developed flexible visualization for analyzing the network performance on the Dragonfly network. They applied data aggregation techniques in order to provide the visualization scalability for large scale networks. Also, their visualization can be customized based on the user's needs. However, the works above do not provide sufficient methods for temporal analysis of network behaviors and performance.

On the other hand, several researchers have also studied techniques for temporal analysis. With an animation based approach, Sigovan et al. (2013b) used an "animated scatterplot" to analyze the temporal patterns in application execution. They visualized not only the occurring events with animation, but also event histories as an afterimage on a background. With this method, we can see the trends from the relationships between previous events and current events. However, it is difficult to find the patterns of lengthy performance data with analysis methods that rely on animation. Isaacs et al. (2014a) visualized execution traces and event histories of parallel applications using logical time instead of physical time. Using logical time allows the application developers to analyze the execution sequence from the program's perspective. Muelder et al. (2016) introduced the behavior lines for analyzing cloud computing performance. These lines show an overview about the behavioral similarity of multivariate time-varying performance data. However, in contrast to the works mentioned above, these methods do not provide information related with the physical network topology. Compared with these methods, our system supports analysis requiring both topological and temporal properties for large scale networks, including the Dragonfly-based networks. Additionally, in order to help the user find important patterns from the large communication data, our system integrates the time-series analysis methods, including clustering, dimensionality reduction, and change point detection.

2.3. Visualization for Time-Series Analysis

Including the temporal analysis methods for network performance data as mentioned in Section 2.2, a large variety of temporal visualizations have already been studied (Aigner et al., 2011). Here, we only summarize the most relevant works. Similar to ours, several studies use dimensionality reduction methods to provide an overview of time-series data. For example, Steiger et al. (2014) produced an overview for identifying anomalies from sensor networks. They used time-series similarity measures (including Euclidean distances and dynamic time warping (Berndt and Clifford, 1994)) and then plotted the similarities in a 2D plot with multi-dimensional scaling (MDS) (Torgerson, 1952). This method focuses on the comparison of each entity's (i.e., sensor's) value over time. On the other hand, some visualizations calculate the similarity of the state of all entities at each time point and then show their temporal differences. For example, Bach et al. (2016) visualized the similarity of multivariate data between each time point by using MDS. Then, they visualized time differences between each point with curved lines. van den Elzen et al. (2016) also applied similar methods. Jäckle et al. (2016) introduced Temporal MDS Plots. The major difference from the methods of Bach et al. (2016); van den Elzen et al. (2016) is that Jäckle et al. (2016) applied a sliding window to the temporal multivariate data in order to obtain the similarity across multiple time points. One of our components in the system applies a similar approach with Steiger et al. (2014). However, our system is designed to compare multiple network metrics with both structural and temporal characteristics in addition to the integration of the time-series analysis methods, as mentioned in Section 2.2. Bryan et al. (2017) introduced Temporal Summary Images (TSIs), which is designed for generating narrative visualizations. TSIs provides data summaries from the time-series data from their timestep selections. Inspired by this idea, we provide automatic summaries of network metrics by utilizing the time-series segmentation method.

3. Analytical Tasks and Design Requirements (DRs)

We first describe the analytical tasks required to understand network behaviors. We then present the design requirements of our system.

In order to fully utilize a supercomputing system, achieving fast communications between compute nodes is crucial. The system designers would need to select effective job allocation and network routing strategies (Won et al., 2015; Yang et al., 2016; Bhatele et al., 2016; Mubarak et al., 2017a) from a variety of options. Therefore, understanding the impact of these strategies to the network behaviors is a necessary task. On the other hand, the application developers need to know where and how communication bottlenecks occur while running their applications. For these tasks, communication data is typically collected by using performance analysis tools, such as profiling and tracing tools (Adhianto et al., 2010; Nagel et al., 1996; Shende and Malony, 2006), or by running simulations (Mubarak et al., 2017b). However, the obtained communication data is often very large (order of gigabytes (NERSC, 2016a) or more). Moreover, these communications occur between thousands of compute nodes connected by complex network topologies (e.g., high-dimensional torus (Adiga et al., 2005) and the Dragonfly network (Kim et al., 2008a)). The researchers need to analyze such complex data from temporal and topological (physical network connection) aspects.

To help the above analytical tasks, our design requirements are as follows: The system should

DR1 clearly display information related with the communication bottlenecks,

DR2 depict temporal network behaviors in order to find the cause of the bottlenecks,

DR3 provide visualizations for understanding communication patterns in the context of the underlying physical networks, and

DR4 help the user identify the communication bottlenecks in large scale data.

Our system is designed to satisfy these requirements. Understanding temporal network behaviors is especially a challenging task due to the large scale data and complexity of the analysis. To support this, our system effectively uses a variety of techniques developed for time-series analysis.

4. Visualization Methodology

In order to meet the design requirements, we develop a visual analytics system for 1) finding potential bottlenecks in the networks, 2) reviewing details of related information of the bottlenecks, and 3) analyzing the causes of the bottlenecks from temporal and topological perspectives. To achieve this, we use the workflow of our visual analytical process as shown in Fig. 2. Our visual analytics system shown in Fig. 3 consists of four components for this analytical process. The four components are: (a) behavior overview, (b) behavior detailed views, (c) behavior similarity views, and (d) topological views. Details of communication data and each of these components will be explained in the following sections.

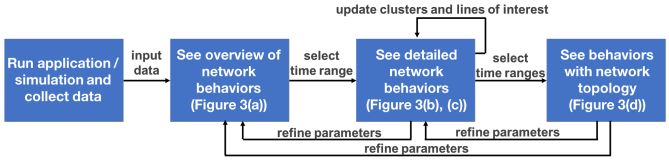


Fig. 2: The analytical flow of using the system. Each step involves one or more views.

4.1. Communication Data

For describing the communication data, we use the Dragonfly network (Kim et al., 2008a), as shown in Fig. 1; however, our visualization methods can be used for other network topologies. This can be achieved by changing the topological views (Fig. 3(d)) according to the network topology in use. Typical communication data collected from a supercomputing system includes time-series of network traffics (i.e., the total amount of data transferred on networks) for each network entity (e.g., a traffic for each global, local, and terminal link). In addition, other metrics, such as saturation time (i.e., the total time during which the buffers of the network channels are full), average packet latency (i.e., the average time for transferring each packet), and average number of hops (i.e., how many hops each packet has traversed in average) could be included in the data. These data is typically collected by using tracing tools or simulations with a user-defined sampling rate. When collecting for the saturation time metric, it is measured as the fraction of the time that the buffers in the network channels are full. Therefore, the saturation time will always either be shorter or the same as the sampling rate. These metrics are useful to identify the bottlenecks. The bottlenecks may arise due to multiple, simultaneous communications passing through the same network links, or long-hop communications along congested links, or both (Bhatele et al., 2016; Bui et al., 2015; Malakar and Vishwanath, 2017).

4.2. Behavior Overview (Fig. 3(a))

In order to help the user decide a time range of their interest, the behavior overview shows one selected statistical metric for each time point across time (DR1, DR4). Time points are encoded in x-coordinates. As for y-coordinates, from the collected dataset, the user can select a set of an entity of the network (e.g., terminals, local links, or global links), a metric (e.g., network traffic, saturation, a number of hops of communication routes), and a statistical measure (e.g., maximum value, mean value, or standard deviation) as values for the y-direction. This view is used for a time range selection with a single range selector placed at the bottom to show more detailed information in the other views. For instance, in Fig. 3(a), the time range where the mean of the terminal traffics is increasing is selected in our example.

4.3. Behavior Detailed Views (Fig. 3(b))

The behavior detailed views show the detailed information of network behavior in the selected time range from the behavior overview (DR1, DR2). Similarly with the behavior overview, x- and y-coordinates represent time points and metric values respectively. In this view, the user can select a subset of an entity of the network and a metric. We decided to provide two views since HPC researchers often want to compare the behaviors of the different network entities (e.g., behaviors of terminals and global links) or understand the relationship of cause and effect in differing situations (e.g., how the network traffic affects the network saturation time). Additionally, with just two views, we can use a sufficient amount of window space to show the detailed information. In Fig. 3(b), the network traffics on terminals and global links are shown in the upper and lower views respectively.

4.3.1. Clustering of Behaviors

Finding interesting patterns from the time-series that show the network behaviors is not a trivial task because today's leading supercomputers have many terminals and network links. For instance, Theta at Argonne National Laboratory (Argonne Leadership Computing Facility, b), which we simulated in our case studies, has more than 3,000 terminals and 20,000 network links. To help the user find the patterns, we apply time-series clustering methods (Liao, 2005; Fu, 2011) for the network behaviors (DR1, DR2, DR4). We implement different clustering methods and similarity measures as described below. Fig. 4 shows visualized results with/without the clustering methods. The user can select a method based on the type of analysis they need and the complexity of visualized data (e.g., a number of network entities and time points).

Our system supports the Hartigan-Wong method (Hartigan and Wong, 1979) as a k-means clustering, the partitioning

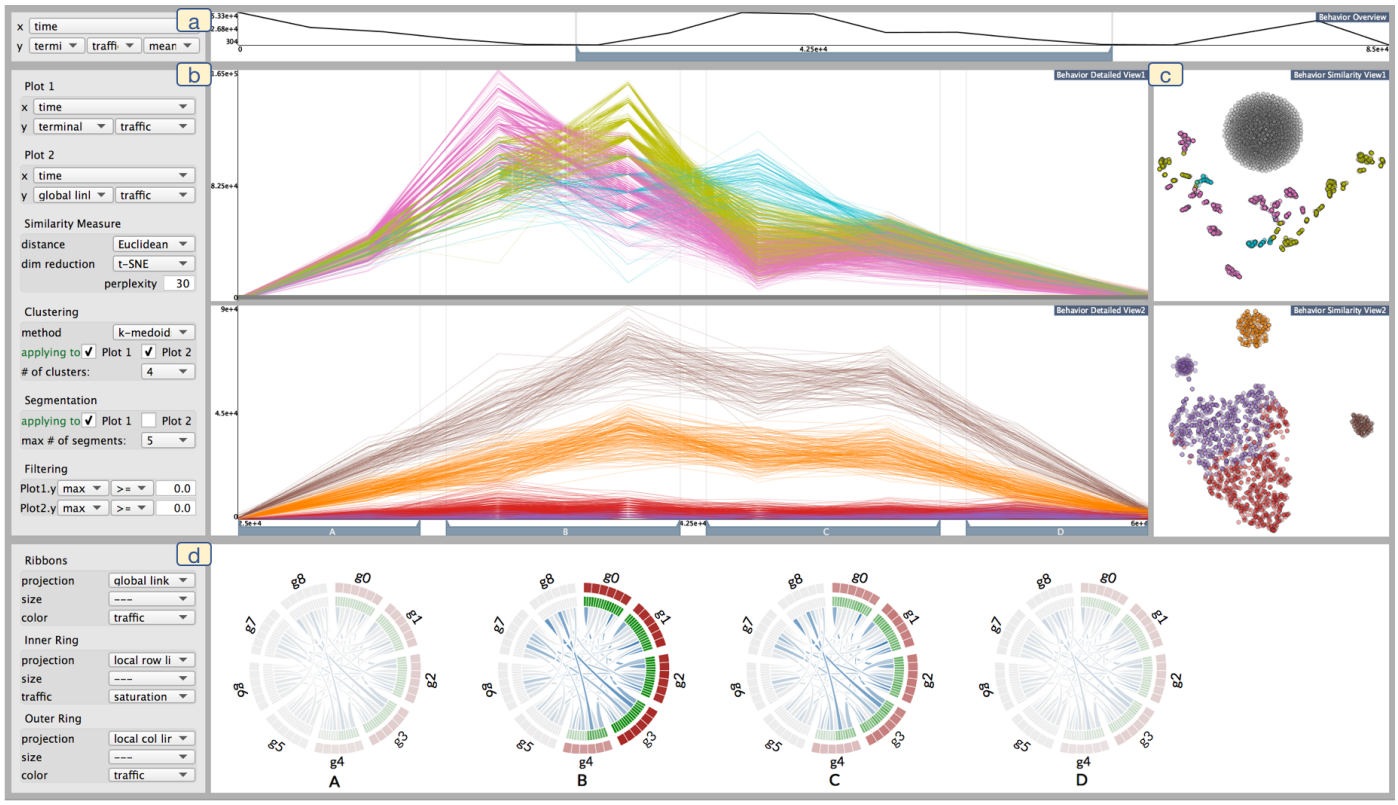


Fig. 3: The user interface of the system, which contains four components: (a) the behavior overview, (b) the behavior detailed views, (c) the behavior similarity views, and (d) the topological views. This example shows the network behaviors obtained by running the algebraic multigrid (AMG) solver application with 1,728 MPI ranks. (a) shows an overview of network behavior about the selected network entity and metric. (b) shows details of network behaviors in the selected time range in (a). (c) shows the similarity of each time-series, as shown in (b), by using the dimensionality reduction method. In (d), summaries of the metrics of each network entity in the selected time ranges with the network topology information. The visualization methods, which are introduced by Li et al. (2017), are used for (d).

around medoids (PAM) as a k-medoids clustering (Kaufman and Rousseeuw, 2009), and the complete-linkage clustering as a hierarchical clustering. k-means clustering is the fastest method within these options, with a time complexity of $O(nk)$ (n is a number of observations and k is a number of cluster centers). It requires observations of l-dimensional vectors as inputs. Thus, we use each time-series as one observation and each point's metric value will be used as an element of the vector. Also, to avoid the initial centroid dependency, our system runs k-means clustering multiple times with different initial centroid seeds and then selects the best result. Fig. 4(a) and 4(b) show examples of visualizations without and with k-means clustering respectively. While k-means clustering uses observations as inputs, the other two clustering methods use dissimilarity between each observation as their inputs. Even though their complexity ($O(n^2)$) is worse than k-means, these clustering methods are useful for analysis since similarity measures developed for the time-series can be applied. Our system supports three similarity measures: Euclidean distance, dynamic time warping (DTW) (Berndt and Clifford, 1994), and time warp edit distance (TWED) (Marteau, 2009). Euclidean distance $d_{\mathrm{E}}(\mathbf{x}, \mathbf{y})$ between two time-series $\mathbf{x}$ and $\mathbf{y}$ is calculated with

$$d_{\mathrm{E}}(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{l} (x_i - y_i)^2 \right)^{\frac{1}{2}} \tag{1}$$

Here, l is the length of the time series, and x_i and y_i are the ith element of the time series $\mathbf{x}$ and $\mathbf{y}$, respectively. By being categorized to the elastic similarity measures, DTW and TWED have flexible matching in time. DTW measure ($d_{\mathrm{DTW}}(\mathbf{x}, \mathbf{y})$) can be obtained by calculating the accumulated cost with dynamic programming:

$$\begin{aligned} d_{\mathrm{DTW}}(\mathbf{x}, \mathbf{y}) &= D_{l,l} \\ D_{i,j} &= f(x_i, y_j) + min\{D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}\} \end{aligned} \tag{2}$$

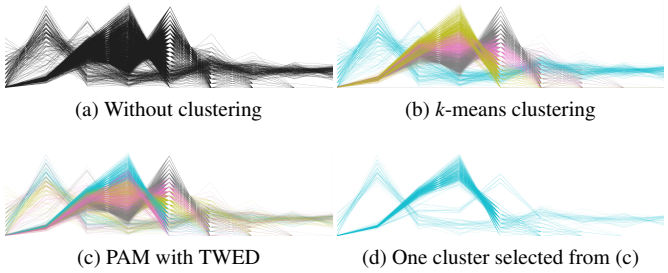
for $i = 1, \cdots, l$ and $j = 1, \cdots, l$. Matrix D is initialized with $D_{i,j} = \infty$ except for $D_{0,0}$. $D_{0,0}$ is initialized to $D_{0,0} = 0$. $f(x_i, y_j)$ is the local cost function. We use the square of the difference between x_i and y_j (i.e., $f(x_i, y_j) = (x_i - y_j)^2$). Similarly, TWED metric ($d_{\mathrm{TWED}}(\mathbf{x}, \mathbf{y})$) can also be calculated with dynamic programming:

$$\begin{aligned} d_{\mathrm{TWED}}(\mathbf{x}, \mathbf{y}) &= D_{l,l} \\ D_{i,j} &= min\{D_{i,j} + \Gamma_{\mathbf{xy}}, D_{i-1,j} + \Gamma_{\mathbf{x}}, D_{i,j-1} + \Gamma_{\mathbf{y}}\} \end{aligned} \tag{3}$$

for $i = 1, \cdots, l$ and $j = 1, \cdots, l$, with

$$\begin{aligned} \Gamma_{\mathbf{xy}} &= f(x_i, y_j) + f(x_{i-1}, y_{j-1}) + 2\nu|i - j| \\ \Gamma_{\mathbf{x}} &= f(x_i, x_{i-1}) + \nu + \lambda \\ \Gamma_{\mathbf{y}} &= f(y_j, y_{j-1}) + \nu + \lambda \end{aligned} \tag{4}$$

We use $f(x_i, y_j) = |x_i - y_j|$ for TWED. λ is a mismatch penalty and ν is a stiffness parameter. Following the parameter choices



(a) Without clustering (b) k-means clustering

(c) PAM with TWED (d) One cluster selected from (c)

Fig. 4: Time-series clustering results of the network traffic with different clustering methods. Colors represent the clustering number where the lines belong to. (a) Without clustering, it is difficult to find important patterns. (b) With k-means clustering, we can easily see the patterns. For example, the light blue lines show that the corresponding traffics are high at the beginning and after that they fall into low values. (c) With the partitioning around medoids (PAM) (Kaufman and Rousseeuw, 2009) and the time-warped edit distance (TWED) (Marteau, 2009), we can see different patterns from (b). For example, as shown in (d), the cluster represented with light blue colors shows three different peaks—all of which have the same behavior of first increasing traffic, then decreasing and having low values afterwards. This shows that using clustering methods with TWED is useful to detect these kinds of patterns as one cluster.

used by Serra and Arcos (2014), we select $\lambda = 0.01$ and $\nu = 0.5$ as the default parameters. While Euclidean distance is the simplest and fastest way (complexity of $O(l)$) to calculate dissimilarity of each time-series, DTW and TWED (complexity of $O(l^2)$) have performed better for classification of time-series data according to current research (Serra and Arcos, 2014). Fig. 4(c) and 4(d) show the results from PAM with TWED. Refer to the work of Serra and Arcos (2014) for more details about the differences between these three measures.

Additionally, when the user wants to cluster the network behaviors based on multiple metrics (e.g., global link traffic and its saturation time), the clustering methods and similarity measures as stated above are also able to be used for multivariate time-series inputs. In this case, the system processes all metrics on a scale between 0 and 1. From these options, the user can select a clustering method, the number of clusters, a similarity measure, and a metric for clustering from the settings, placed on the left-hand side of the behavior detailed views.

The cluster IDs are encoded with line colors as shown in Figures 3 and 4. We select categorical colors, each of which has enough saturation to recognize the differences of each color line with a narrow width. When the same network entities are selected in the upper and lower behavior detailed views, the same color is used for the corresponding lines in order to convey the relationships between two different metrics. Also, the color scheme applied for each behavior detailed view is shared with corresponding behavior similarity view, as described in the next subsection. We have ensured that the views that show different information do not share the same colors in order to avoid misleading the users (e.g., the behavior detailed views and the topological views use different color schemes).

4.4. Behavior Similarity Views (Fig. 3(c))

The (dis)similarity of network behavior is visualized in the behavior similarity views. The behavior similarity views supplement the behavior detailed views. While the behavior detailed views show network behaviors in detail, it is difficult to convey the similarity of each behavior. The system provides the clustering methods for classifying the behaviors; however, it would not be enough in order to find the patterns that occurred in the small set of network entities (e.g., outliers and anomaly behaviors). Upper and lower behavior similarity views shown in Fig. 3(c) show results obtained by applying the dimensionality reduction for the behaviors visualized in the upper and lower behavior detailed views respectively. From these views, the user can identify clusters that would have not been detected with the clustering method used in the behavior detailed views (DR4).

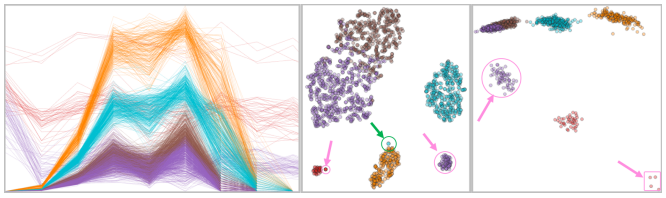
4.4.1. Dimensionality Reduction of Behaviors

We apply the dimensionality reduction method to the dissimilarities obtained by using the same similarity measures used for the time-series clustering. Our system supports the classical multi-dimensional scaling (MDS) (Torgerson, 1952) and t-Distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and Hinton, 2008). These dimensionality reduction methods can use dissimilarities of observations as inputs. By using these dimensionality reduction methods, we can place the similar behaviors close together. While the classical MDS is a linear dimensionality reduction method and good for looking at the global structure of the multi-dimensional data, t-SNE is a non-linear dimensionality reduction method and useful to visualize the local structure of the data. We set t-SNE as a default setting since we designed the behavior similarity views to support finding patterns that occurred in the small set of the network entities. In order to apply t-SNE interactively, we use Barnes-Hut t-SNE (Van Der Maaten, 2014) (while the complexity of the original t-SNE is $O(n^2)$, this implementation has only $O(n \log n)$ complexity). t-SNE has the perplexity as a tuning parameter, which changes how the local structure affects the result. In general, the perplexity is selected between 5 and 50 (Maaten and Hinton, 2008). While we set 30 as a default value, the user can change the value in the settings, placed on the left-hand side of the behavior detailed views. Examples of visualizations with different dimensionality methods are shown in Fig. 5.

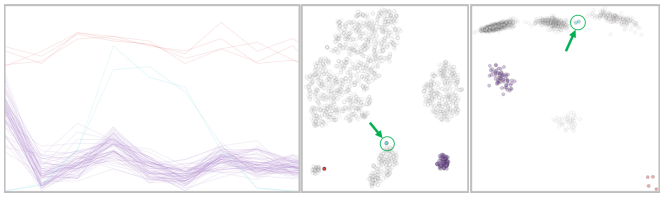
Existing visualization methods (Muelder et al., 2016; Bach et al., 2016; van den Elzen et al., 2016) depict the similarity of all observations at each time point. However, our method can summarize the similarity of each observation's behavior across time rather than at each time point. This method is more useful when we want to find the similar behaviors rather than the similar state of observations.

4.5. Topological Views

All the views described above depict the network behaviors from their time-varying aspects. However, visualizing the context of the underlying physical network is important in order to understand the bottlenecks and its relation to them (Landge et al., 2012; McCarthy et al., 2014; Cheng et al., 2014; Bhatele et al., 2016; Fujiwara et al., 2017; Li et al., 2017). On the other hand, exploring the full time series through such visualizations is a time-consuming task. Therefore, we decide to apply the



(a) Network traffics with clustering (b) t-SNE (c) MDS



(d) Network traffics after selection (e) t-SNE (f) MDS

Fig. 5: Examples of time-series dimensionality reduction. (a) shows network traffics clustered with PAM with Euclidean distances. (b) and (c) are results after dimensionality reduction with t-SNE and MDS respectively. When compared to the clustering result in (a), we can find clusters of smaller size in t-SNE and MDS, as indicated with green and pink arrows in (b) and (c). (d), (e), and (f) show small clusters selected from (b). As shown with the green arrows in (b), (e), and (f), MDS in (c) does not separate the small cluster from the light blue cluster. From these, we can see that t-SNE detects local structure from the data.

similar concept of the temporal summary images (TSIs) (Bryan et al., 2017). Our topological views visualize summaries of the network behaviors with the physical network information during the automatically or manually selected segments (DR3, DR4).

4.5.1. Segmentation of Behaviors

TSIs (Bryan et al., 2017) provide data summaries from the time-series data by using their automatic timestep selections. Since TSIs are developed for generating narrative visualizations (Segel and Heer, 2010), their timestep selections are more focused on capturing changes in the visualizations. In our case, our system should provide summaries that would help the time-series analysis, not narrate it. Therefore, we apply the change point detection, which is developed for time-series analysis (Aminikhanghahi and Cook, 2017). We choose the E-Divisive method (James and Matteson, 2015; Matteson and James, 2014) because we want to detect multiple change points for a set of time-series in a reasonable amount of time in order to use it interactively. Fig. 6 shows an example of segmentation with the E-Divisive method. Segments are visualized with the range sliders placed at the bottom of the behavior detailed views and lines with light gray color in the behavior detailed views. The user is also allowed to adjust the segments manually by using the sliders.

4.5.2. Visualization of Behavior Summaries

After segmentation of the time-series data, our system visualizes the summaries of behaviors for the selected time ranges. We calculate mean values for the metrics for each network entity (e.g., the traffic for global links), then depict them with the

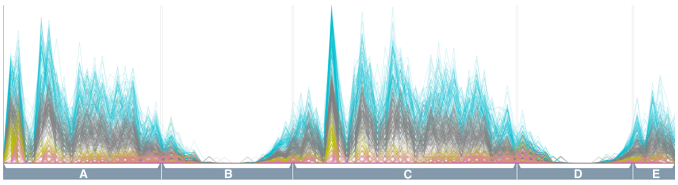


Fig. 6: An example of segmentation for the network behaviors. The E-Divisive detects five segments from multiple lines.

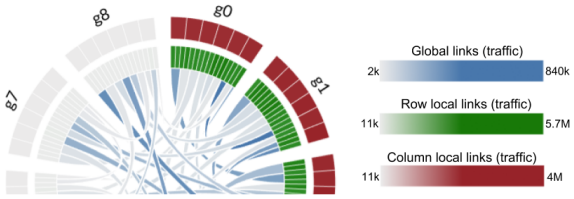


Fig. 7: Radial view with concentric rings for visualizing Theta's interconnect network.

visualization method developed by Li et al. (2017). An example of this visualization is shown in Fig. 3(d). Each alphabetical label corresponds to the label placed in each segment. Each radial view provides a visual summary of the entire network for the selected time range, showing the different traffics of the all types of network links as heatmaps (brighter color means higher traffic). The colors used on the views are the same as the colors used to distinguish the type of network links, as described in Fig. 1: blue for global links, green for row local links, and brown for column local links. Each view shows the aggregated metrics based on the structural and topological properties of the network, as shown in Fig. 7. The ribbons at the center show the aggregated global link traffic between all the groups in the network. The inner ring of the radial view shows the aggregated traffic for all of the row (green) local links. The outer ring shows the aggregated traffic for all of the column (brown) local links. The example in Fig. 3(d) shows network traffics on Theta (Argonne Leadership Computing Facility, b). Since Theta has 9 groups, each radial view has 9 corresponding partitions. Also, each group on Theta has 96 routers, which are arranged in 16 rows and 6 columns. 16 sections of each group in the inner ring (green) show the aggregated traffic on the row local links. 6 sections of each group in the outer ring (brown) represent the aggregated traffic on the column local links. For example, the view corresponding to the time range B has high traffics for all row and column local links in group 0–3 (indicated with g0–3). With the behavior detailed view in hand, by referring these summaries together, we can correlate between the temporal and structural behaviors.

4.6. User Interactions

Our system provides a rich set of user interactions to help the user find the important patterns from large time-series data. Many of the interaction are linked between multiple views.

Behavior overview: As already mentioned in Section 4.2, the metric shown in the view can be selected by the settings on the left. Also, the user can select the time range by using the range slider placed at the bottom of the view. The behavior detailed views (Fig. 3(b)) and the behavior similarity views

(Fig. 3(c)) will be automatically updated after this selection. At the same time, this updates the automatic segmentation in the behavior detailed views, as described in Section 4.5.1.

Behavior detailed views: With the settings placed on the left, the user can select a set of the network entity and its metric, which will be visualized in the upper and lower views. Also, the clustering method, the number of clusters, the similarity measure, and the plot(s) used for clustering and segmentation can be changed. The time ranges used for segmentation can be changed by using the range sliders at the bottom. When the user updates these settings, the behavior similarity views and the topological views are also updated with the corresponding settings.

As for selecting a subset of lines, our system provides three methods. First, the user can apply filtering to the metric value for each view from the settings. Second, the user can select which clusters to visualize in the view from a context menu, which will be displayed with right-mouse click. Additionally, the system provides a freeform selection that selects intersected lines with the freeform drawn by the user. After the freeform selection, the user can filter out the unselected lines. These selections will update the corresponding element in the other views. When the same network entity is shown in multiple views (e.g., the behavior detailed views show traffic and saturation of global links), the corresponding lines will be also filtered out. In addition, when filtering out hierarchically lower levels of network entities (e.g., terminals), in the other views, higher levels of network entities (e.g., global links), which do not have any connections to the said lower level entities, can be filtered out.

Behavior similarity views: The user can select the similarity measures, the dimensionality reduction method, and the perplexity for t-SNE, as explained in Section 4.4.1, from the settings placed on the left. As the MDS or t-SNE results may contain cluttered regions, zooming and panning of the views by using a mouse are supported to display certain regions more clearly or to reduce overlapping of points. In addition, our system provides a lasso selection for the user to select a subset of points of their choice. Same as the behavior detailed views, the user can filter out the unselected points. Then, the corresponding visualization in the other views will be also updated.

4.7. Implementation

We implement the system with multiple programming languages and libraries. We use C++, Qt[1], and OpenGL for the visualizations of the behavior overview, the behavior detailed views, and the behavior similarity views. In addition, by using RInside[2] package, we embed R in C++ codes in order to use analysis methods. As for the topological views, in order to utilize the Web API developed by Li et al. (2017), we use WebSocket[3] for sending the data and settings to the server. Then, we visualize the received result in the Chromium browser[4], which can be integrated into the Qt application. The Web API in the work of Li et al. (2017) is developed with a combination of HTML5, CSS, and JavaScript.

5. Case Studies

We demonstrate the effectiveness of our system by analyzing the network behaviors and performance of parallel applications on the Dragonfly-based network. We focus on analyzing the performance on Intel Knights Landing (Sodani et al., 2016) based Cray XC40 (Cray Inc.) supercomputer, Theta (Argonne Leadership Computing Facility, b), at Argonne National Laboratory. Theta has 3,456 terminals (computer nodes) with each having 64 cores. It serves as the forerunner to the CORAL Aurora system (Argonne Leadership Computing Facility, a), which will be the next leadership supercomputer built in Argonne National Laboratory. Theta has 9 Dragonfly groups with each group having 96 routers arranged in a 16×6 matrix and each router connecting to four terminals.

Since our case studies involve changing the network configuration and routing mechanism, it is expensive and difficult to conduct such studies on Theta's actual system. Therefore, we use the CODES network simulation toolkit (Cope et al., 2011) to model Theta's Dragonfly networks. The simulation enables a controlled environment where the performance of the job is not impacted by external factors, such as communication interference or link failures. The simulation also provides a number of metrics, such as link saturation, packet latency and number of hops traversed. CODES employs the Rensselaer's optimistic simulation system (ROSS) (Carothers et al., 2002; Barnes Jr et al., 2013), a high-performance, parallel discrete-event simulator, that allows massive simulations to be run accurately at a packet level detail. The CODES simulation of the Theta system is validated to have a very high accuracy (Mubarak and Ross, 2017; Mubarak et al., 2017c). The data we captured from simulations is similar to the data collected from the real system. Therefore, our system and methods can also be applied on both data in the same manner. For the simulations, we collected time-series metrics (traffic and saturation time) for all network links (global, local, and terminal). For the terminals, in addition to the traffic and saturation time, we collected packet latency and an average number for hops traversed.

5.1. Case Study 1: Exploring Temporal Behaviors

In this case study, we analyze the performance of the algebraic multigrid (AMG) solver application with 1,728 MPI ranks contiguously assigned to Theta's terminals. AMG is part of unstructured mesh physics packages (Yang et al., 2002), and it has a 3D nearest neighbor communication pattern, which is one of the traffic patterns that represent exascale workload behaviors (NERSC, 2016a). Although we know the communication pattern of AMG at the application level, its temporal behaviors when running on the Dragonfly network are unclear. Direct visualization of massive multivariate time-series using line charts often results in a cluttered view, as shown in Fig. 4(a). By using the time-series clustering method for our system, the terminals are automatically clustered into different groups based on the

[1] Qt, https://www.qt.io/, accessed: 2018-2-5

[2] RInside, https://cran.r-project.org/web/packages/RInside/, accessed: 2018-2-5

[3] WebSocket, https://tools.ietf.org/html/rfc6455, accessed: 2018-2-5

[4] Chromium, https://www.chromium.org/, accessed: 2018-2-5

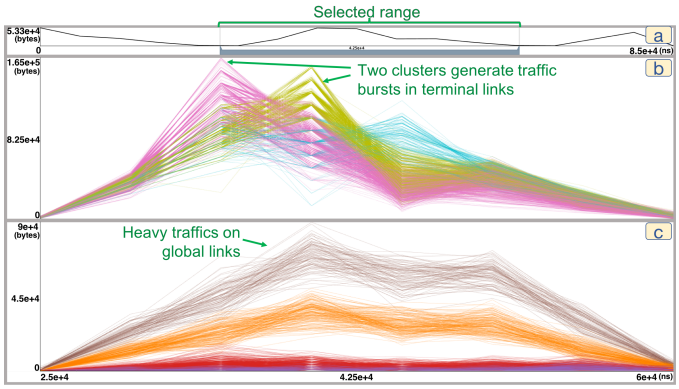


Fig. 8: Network behaviors obtained by running the AMG solver application with the adaptive routing and 1,728 MPI ranks contiguously assigned to Theta's terminals. (a) shows the mean traffics on the terminal links. (b) and (c) shows traffics on the terminals and global links between the selected range in (a), respectively. We can see that the pink and yellow clusters in (b) and the brown cluster in (c) have heavy traffics.

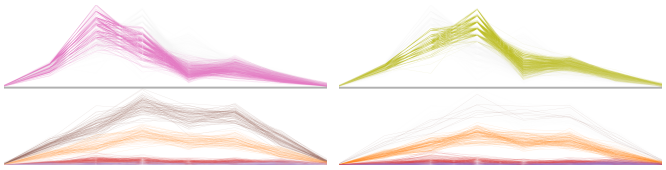


Fig. 9: Visualization of two clusters which have heavy traffics on the terminals in Fig. 8(b). For each (a) and (b), traffics on the terminals and the global links are shown in the upper and lower views respectively. (a) visualizes the pink cluster, which depicts the first heavy traffics on the terminals. From (a), these heavy traffics on the terminals involve heavy traffic on the global links which is indicated with brown colors. (b) visualizes the yellow cluster, which depicts the second heavy traffics on the terminals. When compared with (a), this cluster mostly involves the orange traffics on the global links. This result in (b) shows the adaptive routing helps reduce the communication bottlenecks.

traffic characteristics. We visualize the mean traffics on the terminal links in Fig. 8(a). Then, we select a clear peak as seen around the middle region with the time slider. The upper view in Fig. 8(b) shows the details of the traffics in the selected range. We cluster the result by using PAM with Euclidean distances as the similarity measure. We can see that two clusters, visualized with pink and yellow colors of the terminals, are generating traffic bursts at two different time points: one burst shortly after the other one (Going further into detail, there are more than 1,500 terminal links in the two clusters. During the two peaks from 35 msec to 40 msec, each terminal link has about 100 Kbytes of traffic in average, as observed from Fig. 8(b). Therefore, the total traffic from the two clusters is approximately 30 Gbytes/sec around the peaks whereas the total traffic in low activity periods is less than 10 Gbytes/sec.). This result shows that our clustering method effectively reveals the temporal characteristics of the AMG application workload on Theta's network.

5.2. *Case Study 2: Correlating Application and Network Performance*

Analyzing the effectiveness of routing strategies for a large-scale complex network is challenging. From Case Study 1,

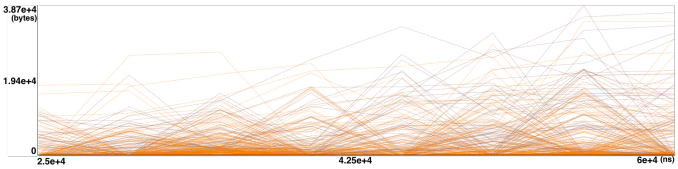


Fig. 10: The maximum saturation time on the global links visualized in Fig. 8(c).

we found the two clusters of the terminals, which caused traffic bursts at different time points. Such traffic bursts may also cause network congestion on global and local links. To compare traffics on the terminals and global links, we visualize these in Fig. 8(b) and (c) respectively. From the result, we can see that heavy traffics also may occur on global links, as shown with a brown-colored cluster. Next, we use the provided user interactions in order to select each cluster of the terminals (pink, yellow) from the upper view. Then, we show only the related global links on the lower view. As the results shown in Fig. 9, two traffic bursts, indicated by the pink and yellow clusters, cause heavy traffics on the global links. In addition to heavy traffics, the global links have high saturation time, as shown in Fig. 10. While the traffic bursts caused severe network congestion, most of the traffic can go through the terminal and local links, but they cannot be sent across the global links immediately via minimal routes. As adaptive routing redirects traffic to the non-minimal routes on global links, the traffic on the global links increases. As shown in Fig. 9(a), the first traffic burst (pink) causes heavy traffics on the global links, which is mostly related to the brown cluster. On the other hand, as shown in Fig. 9(b), the second traffic burst (yellow) mostly relates to the second cluster of global links (orange). Red and purple clusters of global links are associated to both the pink and yellow clusters of the terminals. These global links are mostly associated with the minimal routes. Because adaptive routing uses indirect paths due to network congestion, these global links have low traffic. This case study shows the effectiveness of our system by reviewing the temporal behaviors that relate to the application and the routing strategy.

5.3. *Case Study 3: Network Behavior When Using Many MPI ranks*

In this case study, we analyze the network performance of Theta running the AMG application using 13,824 MPI ranks, with each terminal running 4 MPI ranks using contiguous job placement policy. First, we visualize the maximum saturation time on the terminals in the behavior overview, as shown in Fig. 11(a). Then, we select the range where we can see the peaks of the maximum saturation time. In Fig. 11(b1) and (b2), we show the traffics on global and column local links respectively. PAM with Euclidean distances is applied for each view. Additionally, in Fig. 11(d), in order to analyze this with the topological information, we visualize the traffic of the entire network using the radial views as we described in Section 4.5.2. From these visualizations, while the first peaks of traffics on both the global and local links (indicated the time range B) relate to most of the links, as shown in B of Fig. 11(d), only

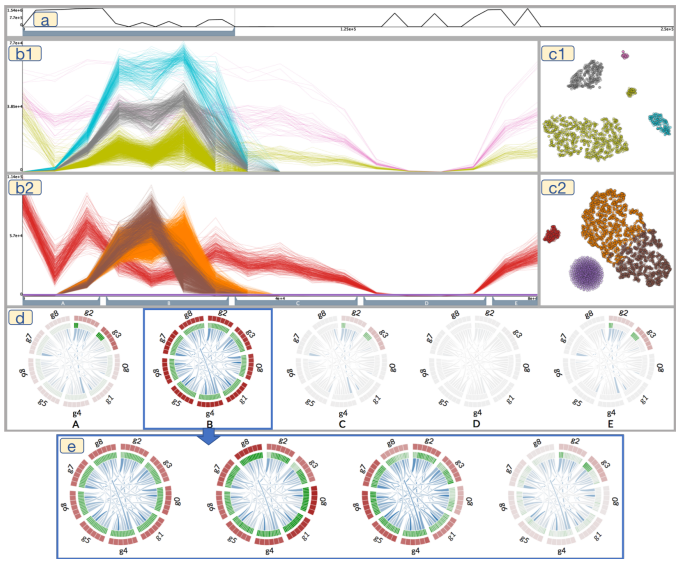


Fig. 11: Network behaviors in the AMG application using 13,824 MPI ranks with the adaptive routing. In (a), the maximum saturation time on terminals is visualized. In (b1) and (b2), traffics on global and column local links are shown respectively. We apply PAM with Euclidean distances as a clustering method for each view. Also, t-SNE is used as a dimensionality reduction in (c1) and (c2). In (d), network traffic summaries for each segment are detected by the E-Divisive method. Ribbons, inner ring, and outer ring colors represent traffics on global, row local, and column local links respectively. (e) shows network traffic at each time point in the time range B.

limited links are utilized after these peaks (indicated the time ranges C, D, E). With further investigations to analyze four time intervals within the time range B (Fig. 11(e)), we noticed that the workload on global links is imbalanced, as indicated by the blue ribbons at the center of the radial views. These visualizations and analysis results indicate two directions to improve the performance: (1) to find a way to utilize unused links in order to reduce the traffics; (2) to use alternative job placement policies. Instead of contiguous job placement policy, a random job placement policy that randomly allocate jobs to different groups or routers can be used to distribute the workload for better load balancing and utilization of the global links.

5.4. *Case Study 4: Analyzing Differences of Communication Patterns by Applications*

While the previous three case studies is about the AMG application, in this case study we analyze the network performance and behaviors for running the multigrid application with 1,000 MPI ranks on Theta. Multigrid has a sparse and irregular communication pattern, which is used in massively parallel block-structured adaptive mesh refinement (AMR) codes (Bell et al., 2012). We focus on analyzing traffics and saturation times on global links because network congestion of global links often causes bottlenecks in such communication patterns. As shown in Fig. 12(a), we visualize the maximum value of saturation time on global links in the behavior overview and then select a time range where the maximum value is increasing. Traffics and saturation times on global links are displayed in the upper and lower behavior detailed views respectively (Fig. 12(b1) and (b2)). Then, we cluster them by applying PAM with Euclidean distances based on both traffics and saturation times (i.e., using multivariate time-series clustering). We can see that this clustering clearly reveals yellow traffics increase the maximum saturation times, as shown in Fig. 12(b1) and (b2), which is a potential bottleneck. The analyst can use this information to identify the source of performance problems, whether it is due to job mapping, application behavior, or routing algorithm. When using simulation, network designers can also tune different parameters such as routing algorithms, link buffer sizes, and bandwidth to see if these network bottlenecks can be reduced.

Through the case studies, we analyzed network behaviors with adaptive routing and contiguous job allocation under the different size of MPI ranks and applications. Also, we demonstrated how we can find the potential bottlenecks. From the analysis results, the application developer can consider the way to improve the performance (e.g., changing mapping of MPI ranks), while the system designer can consider further improvements of routing and job allocation strategies (e.g., using random job allocation).

6. Discussion and Limitations

One of our main contributions is summarizing and classifying a large-scale time-series communication data. We discuss our design choices related with this by comparing potential alternatives. While we depict the network behaviors with two views and time-series clustering methods, the behavior lines used by Muelder et al. (2016) can be used for showing similar information. Muelder et al. (2016) visualized cloud computing performance data, which consists of about 10 attributes, with their behavior lines. The behavior lines are useful to compare the behavior similarity of network entities for each time points; however, they cannot show the detailed values of the data. The communication data that is analyzed with our system has a few attributes (e.g., traffic and saturation time) in general. In this case, comparing by using multiple windows is more effective in order to see cause and effect at the same time. We demonstrated the effectiveness of this design in Case Study 2 and 4. As for the behavior similarity views, these views show the similarity between each network entity's behavior across time. In contrast, other methods (Muelder et al., 2016; Bach et al., 2016; van den Elzen et al., 2016) visualize the similarity of the state of all entities at each time point. While the latter way is good for summarizing the changes of the entire network behaviors for each point, they cannot show the similarity between the behaviors of each network entity across time. Communication bottlenecks often happen in specific network entities where traffic concentration occurs. Therefore, understanding the similarity between behaviors from each network entity is more important than showing the similarity of entire behaviors.

Next, we discuss the limitations of this work. Even though we have designed the system for a large scale network, which has more than thousands of compute nodes, the scalability issues in visualization and computation complexity could occur when we need to visualize a data larger than our current scale. As for the visualization, the larger scale data would make more cluttered lines and points in the behavior detailed views and the behavior similarity views respectively. This problem could be solved by applying aggregation of the data (e.g.,

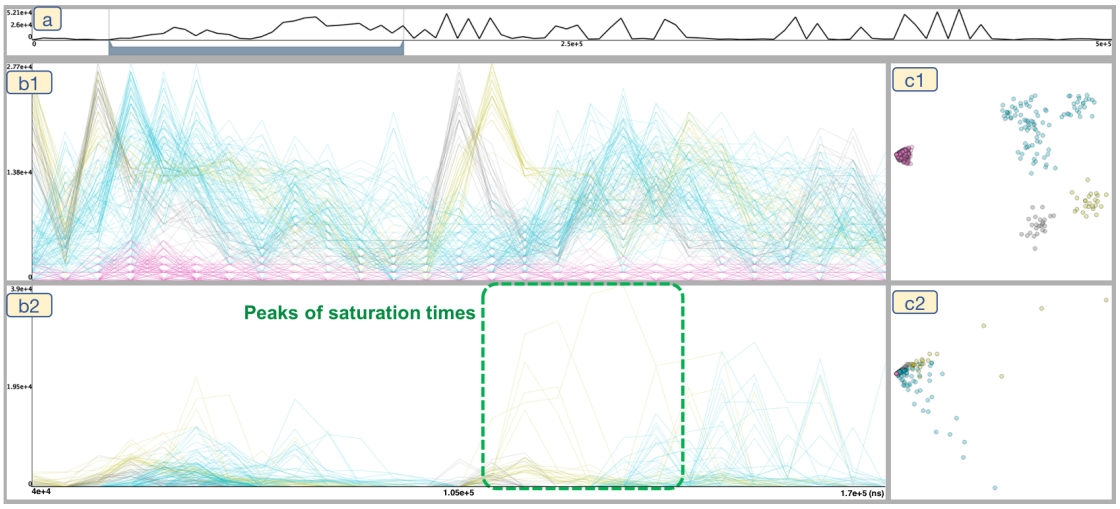


Fig. 12: Network behaviors in the multigrid application using 1,000 MPI ranks with the adaptive routing. In (a), the maximum saturation time on global links is visualized. In (b1) and (b2), traffics and saturation times on global links are shown respectively. We apply the multivariate time-series clustering by using PAM with Euclidean distances based on both traffics and saturation times on the global links. Also, MDS is used as a dimensionality reduction in (c1) and (c2).

based on the similarity of their behaviors). While we did not use edge bundling techniques (Zhou et al., 2013) to distinguish each behavior, such techniques could also reduce clutter. From the run-time perspective of similarity calculation, clustering, dimensionality reduction, and segmentation, a visual analytics system needs to provide fast algorithms in order to apply them interactively. Thus, we carefully chose algorithms based on their computational complexities and effectiveness for classification (Liao, 2005; Fu, 2011; Serra and Arcos, 2014; Aminikhanghahi and Cook, 2017). Because decreasing performance is unavoidable for larger scale data, aggregation of the data, as mentioned above, could help reduce the computation cost. Also, we can consider algorithms that have lower complexity, such as the Approximated-tSNE (Pezzotti et al., 2017). Another limitation is the scope of identification of the bottlenecks. In the case studies, we demonstrated how to identify bottlenecks. As an example, Case Study 4 shows how we can identity which traffics on global links relate to the long saturation time. However, if the user needs to find the cause of bottlenecks in more detail (e.g., why such high traffic even occurred), then the user would need to analyze communication routes of the messages (Fujiwara et al., 2017). Cooperating visualization methods developed by Fujiwara et al. (2017) can be considered as a further research direction.

7. Conclusions

We have designed a visual analytics system for understanding the dynamic behavior of a large-scale supercomputer's interconnects. To assist the user's analysis tasks, our system provides views of the performance data at different levels of granularity and also snapshots of the network traffic. While many studies on Dragonfly networks focus only on analyzing the topological properties, our work supports both the topological and temporal analysis of network behaviors. To address the challenges presented by the complexity of hierarchical networks to applying data mining techniques for analyzing multiple sets of multivariate time-series data, our system couples time-series clustering methods with interactive visualizations to correlate the performance metrics at different levels of the network hierarchies. Our case studies have proven the effectiveness of these integrated capabilities for network performance analysis.

For future work, we plan to enhance our system in several ways. Currently, the users of our system must manually choose which clustering method and similarity metric they would want to use. It is possible to automatically make these selections for the users based on the data characteristics such as anomalies found in the time series. We also plan to employ GPU computing to accelerate data processing and visualization calculations such that our system can handle large and high-resolution data. Finally, we plan to extend our system to support analyzing the temporal behavior of job interference caused by different job placement policies as well as analyzing the communication pattern and congestion of the network.

Acknowledgments

This research was sponsored by the Advanced Scientific Computing Research Program, the Office of Science, U.S. Department of Energy through grants DE-SC0014917, DE-SC0012610, and DE-AC02-06CH11357.

References

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

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

Aigner, W., Miksch, S., Schumann, H., Tominski, C., 2011. Visualization of time-oriented data. Springer Science & Business Media.

Aminikhanghahi, S., Cook, D.J., 2017. A survey of methods for time series change point detection. Knowledge and information systems 51, 339–367.

Argonne Leadership Computing Facility, a. Aurora, Argonne's next generation supercomputer. https://aurora.alcf.anl.gov/. Accessed: 2017-12-11.

Argonne Leadership Computing Facility, b. Theta, Argonne's Cray XC System. https://www.alcf.anl.gov/theta. Accessed: 2017-12-11.

Bach, B., Shi, C., Heulot, N., Madhyastha, T., Grabowski, T., Dragicevic, P., 2016. Time curves: Folding time to visualize patterns of temporal evolution in data. IEEE Transactions on Visualization and Computer Graphics 22, 559–568.

Barnes Jr, P.D., Carothers, C.D., Jefferson, D.R., LaPre, J.M., 2013. Warp speed: executing time warp on 1,966,080 cores, in: Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, pp. 327–336.

Bell, J., Almgren, A., Beckner, V., Day, M., Lijewski, M., Nonaka, A., Zhang, W., 2012. Boxlib users guide. github. com/BoxLib-Codes/BoxLib .

Berndt, D.J., Clifford, J., 1994. Using dynamic time warping to find patterns in time series., in: KDD workshop, pp. 359–370.

Bhatele, A., Jain, N., Livnat, Y., Pascucci, V., Bremer, P.T., 2016. Analyzing network health and congestion in dragonfly-based supercomputers, in: IEEE Parallel and Distributed Processing Symposium, pp. 93–102.

Bryan, C., Ma, K.L., Woodring, J., 2017. Temporal summary images: An approach to narrative visualization via interactive annotation generation and placement. IEEE Transactions on Visualization and Computer Graphics 23, 511–520.

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

Carothers, C.D., Bauer, D., Pearce, S., 2002. Ross: A high-performance, low-memory, modular time warp system. Journal of Parallel and Distributed Computing 62, 1648–1669.

Cheng, S., De, P., Jiang, S.H.C., Mueller, K., 2014. TorusVisND: Unraveling High-Dimensional Torus Networks for Network Traffic Visualizations, in: Proceedings of IEEE Workshop on Visual Performance Analysis, pp. 9–16.

Cope, J., Liu, N., Lang, S., Carns, P., Carothers, C., Ross, R.B., 2011. CODES: Enabling co-design of multilayer exascale storage architectures, in: Proceedings of the Workshop on Emerging Supercomputing Technologies.

Cray Inc., . Cray XC40. http://www.cray.com/sites/default/files/resources/CrayXC40Brochure.pdf. Accessed: 2017-12-11.

DeRose, L., Homer, B., Johnson, D., 2007. Detecting application load imbalance on high end massively parallel systems , 150–159.

van den Elzen, S., Holten, D., Blaas, J., van Wijk, J.J., 2016. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. IEEE Transactions on Visualization and Computer Graphics 22, 1–10.

Fu, T.c., 2011. A review on time series data mining. Engineering Applications of Artificial Intelligence 24, 164–181.

Fujiwara, T., Malakar, P., Reda, K., Vishwanath, V., Papka, M.E., Ma, K.L., 2017. A visual analytics system for optimizing communications in massively parallel applications, in: Proceedings of IEEE Conference on Visual Analytics Science and Technology.

Geimer, M., Wolf, F., Wylie, B.J., Ábrahám, E., Becker, D., Mohr, B., 2010. The scalasca performance toolset architecture. Concurrency and Computation: Practice and Experience 22, 702–719.

Hartigan, J.A., Wong, M.A., 1979. A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) 28, 100–108.

Isaacs, K.E., Bremer, P.T., Jusufi, I., Gamblin, T., Bhatele, A., Schulz, M., Hamann, B., 2014a. Combing the communication hairball: Visualizing parallel execution traces using logical time. IEEE Transactions on Visualization and Computer Graphics 20, 2349–2358.

Isaacs, K.E., Giménez, A., Jusufi, I., Gamblin, T., Bhatele, A., Schulz, M., Hamann, B., Bremer, P.T., 2014b. State of the art of performance visualization, in: Eurographics Conference on Visualization (EuroVis) STARs, pp. 141–160.

Jäckle, D., Fischer, F., Schreck, T., Keim, D.A., 2016. Temporal MDS plots for analysis of multivariate data. IEEE Transactions on Visualization and Computer Graphics 22, 141–150.

Jain, N., Bhatele, A., Ni, X., Wright, N.J., Kale, L.V., 2014. Maximizing throughput on a Dragonfly network, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 336–347.

James, N.A., Matteson, D.S., 2015. ecp: An R package for nonparametric multiple change point analysis of multivariate data. Journal of Statistical Software 62.

Jiang, N., Kim, J., Dally, W.J., 2009. Indirect adaptive routing on large scale interconnection networks, in: ACM SIGARCH Computer Architecture News, pp. 220–231.

Kaufman, L., Rousseeuw, P.J., 2009. Finding groups in data: an introduction to cluster analysis. volume 344. John Wiley & Sons.

Kim, J., Dally, W.J., Scott, S., Abts, D., 2008a. Technology-Driven, Highly-Scalable Dragonfly Topology, in: International Symposium on Computer Architecture, pp. 77–88.

Kim, J., Dally, W.J., Scott, S., Abts, D., 2008b. Technology-driven, highly-scalable dragonfly topology, in: ACM SIGARCH Computer Architecture News, pp. 77–88.

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

Leiserson, C.E., 1985. Fat-trees: universal networks for hardware-efficient supercomputing. IEEE Transactions on Computers 100, 892–901.

Li, J.K., Mubarak, M., Ross, R.B., Carothers, C.D., Ma, K.L., 2017. Visual analytics techniques for exploring the design space of large-scale high-radix networks, in: IEEE International Conference on Cluster Computing, pp. 193–203.

Liao, T.W., 2005. Clustering of time series data—a survey. Pattern recognition 38, 1857–1874.

Los Alamos National Laboratory, 2016. The Trinity advanced technology system. http://www.lanl.gov/projects/trinity. Accessed: 2017-012-11.

Maaten, L.v.d., Hinton, G., 2008. Visualizing data using t-SNE. Journal of Machine Learning Research 9, 2579–2605.

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

Marteau, P.F., 2009. Time warp edit distance with stiffness adjustment for time series matching. IEEE Transactions on Pattern Analysis and Machine Intelligence 31, 306–318.

Matteson, D.S., James, N.A., 2014. A nonparametric approach for multiple change point analysis of multivariate data. Journal of the American Statistical Association 109, 334–345.

McCarthy, C.M., Isaacs, K.E., Bhatele, A., Bremer, P.T., Hamann, B., 2014. Visualizing the five-dimensional torus network of the IBM Blue Gene/Q, in: Proceedings of IEEE Workshop on Visual Performance Analysis, pp. 24–27.

Mubarak, M., Carns, P., Jenkins, J., Li, J.K., Jain, N., Snyder, S., Ross, R.B., Carothers, C.D., Bhatele, A., Ma, K.L., 2017a. Quantifying I/O and communication traffic interference on Dragonfly networks equipped with burst buffers, in: IEEE International Conference on Cluster Computing, pp. 204–215.

Mubarak, M., Carothers, C.D., Ross, R.B., Carns, P., 2017b. Enabling parallel simulation of large-scale HPC network systems. IEEE Transactions on Parallel and Distributed Systems 28, 87–100.

Mubarak, M., Jain, N., Domke, J., Wolfe, N., Ross, C., Li, K., Bhatele, A., Carothers, C.D., Ma, K.L., Ross, R.B., 2017c. Toward reliable validation of hpc network simulation models, in: Winter Simulation Conference.

Mubarak, M., Ross, R.B., 2017. Validation Study of CODES Dragonfly Network Model with Theta Cray XC System. Technical Report. Argonne National Laboratory.

Muelder, C., Zhu, B., Chen, W., Zhang, H., Ma, K.L., 2016. Visual analysis of cloud computing performance using behavioral lines. IEEE Transactions on Visualization and Computer Graphics 22, 1694–1704.

Nagel, W.E., Arnold, A., Weber, M., Hoppe, H.C., Solchenbach, K., 1996. Vampir: Visualization and analysis of mpi resources .

NERSC, 2016a. Characterization of the DOE mini-apps. http://portal.nersc.gov/project/CAL/doe-miniapps.htm. Accessed: 2017-12-11.

NERSC, 2016b. Cori Cray XC40 System at NERSC. http://www.nersc.gov/users/computational-systems/cori/. Accessed: 2017-12-11.

Pezzotti, N., Lelieveldt, B.P., van der Maaten, L., Höllt, T., Eisemann, E., Vilanova, A., 2017. Approximated and user steerable tSNE for progressive visual analytics. IEEE Transactions on Visualization and Computer Graphics 23, 1739–1752.

Sagan, H., 1994. Space-filling curves. Springer-Verlag.

Segel, E., Heer, J., 2010. Narrative visualization: Telling stories with data. IEEE Transactions on Visualization and Computer Graphics 16, 1139–1148.

Serra, J., Arcos, J.L., 2014. An empirical evaluation of similarity measures for time series classification. Knowledge-Based Systems 67, 305–314.

Shende, S.S., Malony, A.D., 2006. The TAU parallel performance system. The International Journal of High Performance Computing Applications 20, 287–311.

Sigovan, C., Muelder, C., Ma, K.L., Cope, J., Iskra, K., Ross, R.B., 2013a. A visual network analysis method for large-scale parallel I/O systems, in: IEEE International Parallel and Distributed Processing Symposium, pp. 308–319.

Sigovan, C., Muelder, C.W., Ma, K.L., 2013b. Visualizing large-scale parallel communication traces using a particle animation technique, in: Computer Graphics Forum, Wiley Online Library. pp. 141–150.

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

Steiger, M., Bernard, J., Mittelstädt, S., Lücke-Tieke, H., Keim, D., May, T., Kohlhammer, J., 2014. Visual analysis of time-series similarities for anomaly detection in sensor networks, in: Computer graphics forum, Wiley Online Library. pp. 401–410.

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

Torgerson, W.S., 1952. Multidimensional scaling: I. theory and method. Psychometrika 17, 401–419.

Van Der Maaten, L., 2014. Accelerating t-SNE using tree-based algorithms. Journal of Machine Learning Research 15, 3221–3245.

Wolfe, N., Carothers, C.D., Mubarak, M., Ross, R.B., Carns, P., 2016. Modeling a million-node slim fly network using parallel discrete-event simulation, in: Proceedings of ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, pp. 189–199.

Won, J., Kim, G., Kim, J., Jiang, T., Parker, M., Scott, S., 2015. Overcoming far-end congestion in large-scale networks, in: IEEE International Symposium on High Performance Computer Architecture, pp. 415–427.

Yang, U.M., et al., 2002. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. Applied Numerical Mathematics 41, 155–177.

Yang, X., Jenkins, J., Mubarak, M., Ross, R.B., Lan, Z., 2016. Watch out for the bully!: job interference study on Dragonfly network, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, p. 64.

Zhou, H., Xu, P., Yuan, X., Qu, H., 2013. Edge bundling in information visualization. Tsinghua Science and Technology 18, 145–156.