A Visual Analytics Framework for Analyzing Parallel and Distributed Computing Applications

Jianping Kelvin Li*
University of California, Davis
Takanori Fujiwara*
University of California, Davis
Suraj P. Kesavan*
University of California, Davis
Caitlin Ross†
Rensselaer Polytechnic Institute
Misbah Mubarak‡
Argonne National Laboratory
Christopher D. Carothers†
Rensselaer Polytechnic Institute
Robert B. Ross‡
Argonne National Laboratory
Kwan-Liu Ma*
University of California, Davis

ABSTRACT

To optimize the performance and efficiency of HPC applications, programmers and analysts often need to collect various performance metrics for each computer at different time points as well as the communication data between the computers. This results in a complex dataset that consists of multivariate time-series and communication network data, which makes debugging and performance tuning of HPC applications challenging. Automated analytical methods based on statistical analysis and unsupervised learning are often insufficient to support such tasks without the background knowledge from the application programmers. To better explore and analyze a wide spectrum of HPC datasets, effective visual data analytics techniques are needed. In this paper, we present a visual analytics framework for analyzing HPC datasets produced by parallel discrete-event simulations (PDES). Our framework leverages automated time-series analysis methods and effective visualizations to analyze both multivariate time-series and communication network data. Through several case studies for analyzing the performance of PDES, we show that our visual analytics techniques and system can be effective in reasoning multiple performance metrics, temporal behaviors of the simulation, and the communication patterns.

Keywords: Visual analytics, information visualization, time-series data, multivariate data, performance analysis, parallel discrete-event simulation

1 INTRODUCTION

Data science and scientific research rely on parallel and distributed computing to support large-scale data processing and analysis. Parallel and distributed computing applications typically run on high-performance computing (HPC) systems, which have multiple computing nodes with multicore processors that are interconnected. Performance optimization and debugging for HPC applications are difficult because background knowledge from the application programmers is often required. In addition, performance issues and bottlenecks can occur at specific time intervals or in any one of the processes. To analyze the behaviors and optimize the performance of HPC applications, programmers need to collect various performance metrics from each computing node at different time points as well as the communication events among the nodes. The collected dataset contains multivariate time-series and communication network data, which makes data analysis and exploration challenging. Therefore, automated analytical methods based on statistical methods and unsupervised learning are often insufficient to analyze HPC datasets. To effectively explore and analyze HPC datasets, visual analytics techniques that combined interactive visualization and machine learning methods are required. Many approaches and tools have been developed for analyzing the performance of HPC applications [18]. However, conventional visualization methods and tools are insufficient for analyzing both multivariate time-series and communication network data as well as correlating them for performance analysis. In our previous work [14], we have developed a visual analytics system for analyzing Dragonfly [22] networks. We applied various time-series analysis methods and visualization techniques to allow users to explore both temporal behaviors and network traffics in a dashboard with multiple coordinated views. However, our previous system lacks support for correlating the temporal behaviors and similarity of the computing nodes to communication patterns, which is often needed for performance optimization of HPC applications. In addition, the visual analytics methods need to be generalized and extended to provide sufficient support for analyzing HPC systems and applications.

In this paper, we present a visual analytics framework for effectively analyzing HPC datasets. Our framework generalizes and extends the automated time-series analysis methods from our previous work [13] to reveal temporal behaviors and identify important time intervals for potential performance issues. For correlating multiple performance metrics with communication patterns over time, we visualize the communication patterns for important time intervals along with the time-series clustering results. In addition to closely coupling these analysis and visualization methods, our framework also supports interactive visualizations for exploring HPC datasets. To demonstrate the applicability and usefulness of our framework, we use it to develop a visual analytics system for analyzing parallel discrete-event simulation (PDES), which is a cost-effective tool for modeling and evaluating scientific phenomena and complex systems. Through several case studies, we show that the data analytics and visualization methods incorporated by our framework can effectively analyze and correlate multivariate time-series and communication network data. In addition, we discuss how our framework can be extended to support real-time analysis and monitoring of streaming data from HPC systems and applications.

2 BACKGROUND AND RELATED WORK

While HPC applications typically run on a cluster of interconnected computing nodes, a communication protocol, such the Message Passing Interface [15], is often used for coordinate for parallel and distributed computing. To analyze the behaviors and performance of HPC applications, system or application level data need to be collected on each computing node with the communication data among the nodes. As we use the application of parallel discrete-event simulation (PDES) for case study, here we first provide the background information of PDES and its data characteristics.

2.1 Parallel Discrete-Event Simulation

PDES is a cost-effective and useful tool for modeling and researching in many areas, ranging from studies of complex physical phenomena to the designs of supercomputers. In a PDES model, events are processed concurrently on different computing nodes. Conser-

*e-mail: {kelli, tfujiwara, spkesavan, klma}@ucdavis.edu
†e-mail: rossc3@rpi.edu, chrisc@cs.rpi.edu
‡e-mail: mmubarak@anl.gov, rross@mcs.anl.gov

vative synchronization and state saving can be used to assure the correctness and accuracy of simulation models while running the simulations in a parallel distributed fashion [32]. To improve the efficiency of PDES, optimistic synchronization based on the Time Wrap algorithm [20] can be used instead of conservative synchronization. With optimistic synchronization, each logical process (LP) in the simulation independently processes events without frequent global synchronization among the processing elements (PEs). When the causality constraint is broken (i.e., an LP receives an event that should have happened in the past), the LP must be rolled back immediately to execute the event again in the correct order. Although optimistic synchronization has been shown to improve the scalability, it relies on rollbacks of out-of-order events and either reverse computation or state saving to ensure the overall correctness of the simulations. Multiple factors can affect the rollback behavior of an optimistic PDES and it is difficult to understand the rollback behavior for choosing the optimal configuration of parameters, which makes debugging and tuning for PDES a challenging task.

In this work, we use the Rensselaer's Optimistic Simulation System (ROSS) [9], an open source discrete-event simulator that provides optimistic parallel event scheduling. ROSS uses a technique called *reverse computation* [10] where the model developers decide how the rollback mechanism works using user-defined functions, which allows ROSS to process up to billions of events [29]. However, the performance and efficiency of ROSS depend on the models being simulated and the associated workload, which can impose subtle performance issues and bottlenecks. Interactive analysis and visualization of the instrumented data are necessary for understanding the performance issues to mitigate the bottlenecks in order to optimize performance. By using our visual analytics system, data collected from ROSS can be used to effectively understand complex behaviors, debug simulation models, and optimize performance.

Understanding the workload characteristics of HPC applications is necessary for effective performance analysis. Like many parallel applications, ROSS uses Message Passing Interface(MPI) for distributed and parallel computing, where each PE is an MPI process. To enable optimistic synchronization, ROSS maintains its own clock called the global virtual time (GVT) and is computed as the minimum of all unprocessed events across all PEs in the simulation [20]. Additionally, ROSS pre-allocates memory for storing events when initializing simulations. An event must continue to be stored in memory until the timestamp is less than the GVT, which means the event is committed and will not be rolled back. ROSS introduces the kernel process (KP) for managing a shared processed event list for a collection of LPs mapped to a single PE. When a KP rolls back, it must roll back all its LPs to the same point in virtual time, due to which some LPs could be rolled back unnecessarily. Having too many LPs mapped to a KP can result in performance degradation due to unnecessary rollbacks. In addition, the MPI communication between PEs can cause out-of-order events which lead to rollbacks. Therefore, we need to analyze both the rollback and communication behaviors for identifying performance bottlenecks in PDES.

2.2 Data and Properties

Our framework is designed for analyzing HPC applications that collect multivariate time-series and communication network data for analyzing performance. In general, the time-series data contains numerous performance metrics for each computing node at a specific sampling rate, which is also used for collecting the communication data among all the computing nodes. For our case studies, we use the ROSS Instrumentation layer [34] to collect time series and communication network data. The instrumentation layer provides three time-sampling modes, which allow users to sample simulation engine and model data (1) immediately after GVT computation, (2) at real-time sampling intervals, or (3) at virtual time sampling intervals. When sampling data from the simulation engine, the user can collect data for numerous metrics at different granularities (PE, KP, and LP). The specific metrics used in our visual analysis case studies are:

- **Primary Rollbacks**: The number of rollbacks on a KP caused by receiving an out-of-order event.
- **Secondary Rollbacks**: The number of rollbacks on a KP caused by an anti-message (i.e., a cancellation message).
- **Virtual Time Difference**: The difference in time between a KP's local virtual clock and the current GVT. The value is typically positive, but it can be negative, indicating that the KP has not processed any events since the last GVT.
- **Network Sends**: Number of events sent by LPs over the network.
- **Network Receives**: Number of events received by LPs over the network.

The metrics collected are either directly related to rollback behavior (e.g., number of rollbacks) or communication among PEs, KPs, and LPs (e.g., network sends), which has an effect on rollback behavior.

2.3 Related Work

Various visualization methods and tools have been developed for performance analysis of HPC applications. Isaacs et al. [18] provided comprehensive surveys of performance visualizations. Several general-purpose performance tools, such as HPCToolkit [1], VAMPIR [31], and TAU [36], provide graphical results of performance profiles and traces of parallel applications. However, the visualizations that these tools provide are not designed for analyzing large-scale data. While some of them (e.g., TAU) can show the performance metric (e.g., execution time) with physical locations of the processes, they do not provide effective visualizations to show patterns of the communication-related performance metric between simulation entities (e.g., network routers, MPI ranks, etc). These missing features are necessary to analyze large-scale parallel applications.

Researchers have developed visualizations for large-scale parallel applications, such as applications running on supercomputers. Boxfish [25] projects 3D torus networks used in supercomputers on both 2D and 3D views to analyze network traffics with the topological properties. Bhatele et al. [5] 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. [14] utilized node-link diagrams and the matrix-based representations with hierarchical aggregation techniques to visualize any type of network topologies. Li et al. [26] developed flexible visualization to analyze the network performance on the Dragonfly network by applying data aggregation techniques to provide the visualization scalability for large scale networks.

To analyze optimistic PDES, Ross et al. [34] introduced a visual analytics tool specialized for the ROSS optimistic PDES framework [9]. This tool is designed to analyze the simulator performance from multiple aspects, such as communication patterns and correlations between multiple performance metrics. However, only the aggregated values of the selected metrics are visualized for analyzing the change of the performance metrics over time, which is insufficient to depict the temporal behaviors of different entities. Because various temporal aspects affect the rollback behaviors and performance issues, more advanced temporal analysis of performance metrics is required to better understand the performance and behaviors of optimistic PDES.

Several researchers have studied techniques for temporal analysis. With an animation based approach, Sigovan et al. [37] used an animated scatterplot to analyze the temporal patterns in application

execution. However, it is difficult to find the patterns of lengthy performance data with analysis methods that rely on animation. The Ravel visualization tool [17] visualizes 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. [30] introduced the "behavior lines" for analyzing cloud computing performance. These lines show an overview of the behavioral similarity of multivariate time-varying performance data. Fujiwara et al. [13] designed a visual analytics system that integrated various advanced time-series analysis and unsupervised machine learning methods to overview and analyze the network behaviors of large-scale parallel applications.

However, these approaches only support analysis of either time-series or network data. Effective visual analytics methods and tools are lacking for analyzing both multivariate time-series and communication data as well as exploring their correlations. In our work, we adapt the methods for summarizing large-scale network and the temporal analysis methods for large-scale parallel applications to facilitate effective analysis and exploration of HPC datasets.

3 Methodology

An overview of our framework is shown in Fig. 1. The multivariate time-series and communication network data are first preprocessed and indexed for efficient analysis. Time-series clustering is used to classify each process in a HPC application based on their changes of performance metrics over time, and change point detection algorithms help identify the most important time intervals. To analyze communication patterns, we can filter and aggregate the communication network data based on these time intervals to generate multiple views of the network for identifying bottlenecks. Combining these network views with the temporal behaviors views can help correlate among different performance metrics and communication patterns.

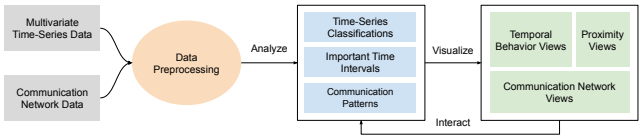
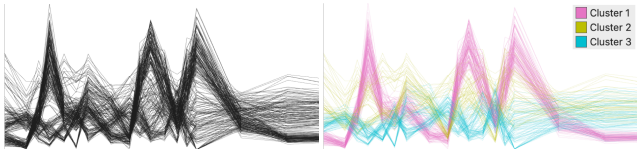


Figure 1: Overview of our visual analytics framework for analyzing HPC applications and systems.

3.1 Time-Series Clustering

Analyzing the temporal behaviors of HPC applications is useful for identifying bottlenecks and optimizing performance. By applying time-series clustering techniques, we can easily identify the subgroups among the computing nodes at different granularities.

In our approach, we employ the multiple time-series clustering methods [12] and similarity measures used in [13]. In particular, we use the Hartigan-Wong method [16] as a k-means clustering, the partitioning around medoids (PAM) as a k-medoids clustering [21], and the complete-linkage clustering as a hierarchical clustering method. k-means clustering is the fastest method among these options, with time complexity of $O(nk)$ (where n is a number of observations, and k is a number of cluster centers). However, it requires observations of l-dimensional vectors as inputs. Thus, we treat each time-series as one observation and each processor's metric value as an element of the vector. Also, to avoid the initial centroid dependency, our system runs k-means clustering multiple times (ten as a default) with different initial centroid seeds, and then selects the best result. 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 (1) they are more robust to noise and outliers [43], and (2) other similarity measures developed for the time-series analysis can be easily applied.



(a) Without clustering (b) PAM with Euclidean distance

Figure 2: Visualized results of the number of secondary rollbacks without and with clustering. Colors represent the clustering labels where the lines belong to. (a) Without clustering, it is difficult to find important patterns. (b) Using PAM clustering with Euclidean distance as the similarity measure, we can easily find the patterns. For example, the pink lines show that the corresponding simulation entities' number of secondary rollbacks have high fluctuations.

To effectively perform time-series clustering, we use three similarity measures: traditional Euclidean distance, Dynamic Time Warping (DTW) [4], and Time Warp Edit Distance (TWED) [28]. Since DTW and TWED are the elastic similarity measures, we use them to perform flexible matching in the time-series data. In comparison to Euclidean distance which 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 the recent research [35]. Fig. 2a and Fig. 2b show examples of visualizations without any clustering and with PAM with Euclidean distance as the similarity measure, respectively. The cluster labels are encoded with line colors, as shown in Fig. 2. In Fig. 2b, we can clearly see the different patterns of performance behaviors. Refer to the works of [13, 35] for more details about each similarity measure and the differences between these three measures.

3.2 Time-Series Dimensionality Reduction (DR)

Because time-series clustering methods can only reveal certain temporal patterns, it might fail to inform some important patterns which are derived from a small set of the entities (e.g., sub-clusters and outliers). DR methods can supplement time-series clustering because they depict more detailed similarities between each entity.

We leverage classical Multi-Dimensional Scaling (MDS) [39] and t-Distributed Stochastic Neighbor Embedding (t-SNE) [42]. For calculating a similarity between each time-series, we use the same similarity measures used with the time-series clustering. By using these DR methods, we can notice entities with similar behaviors being close together. While the classical MDS is a linear DR method and good for looking at the global structure of the multi-dimensional data, t-SNE is a nonlinear DR method and useful to visualize the local structure of the data. Because t-SNE often requires a longer calculation time, to apply t-SNE more interactively, we use Barnes-Hut t-SNE [41] (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 controls a balance of the effects from local and global structures of the data [42]. While a large perplexity will preserve more of the distance relationship in the global structure, a small perplexity will focus on preserving the distance relationship between a small number of neighborhoods In most applications, the perplexity is set between 5 and 50 [42]. We set the default value to be 30, and the user can change the value based on the analysis.

Examples of visualizations with different DR methods are shown in Fig. 3. These examples show that DR-based visualization helps find subgroups and outliers within the clusters of time-series.

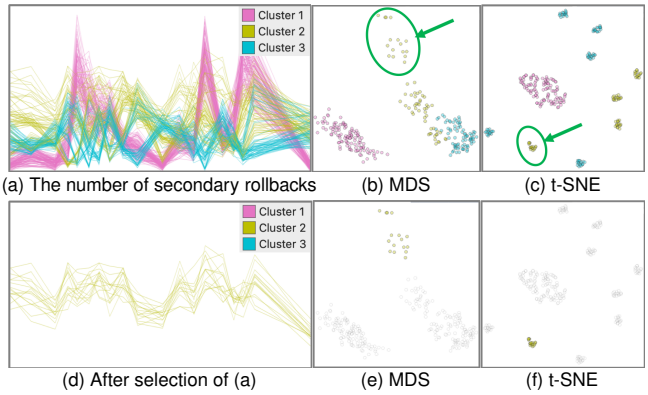


Figure 3: Examples of time-series DR. (a) shows the number of secondary rollbacks clustered with PAM with Euclidean distances. (b) and (c) are results after dimensionality reduction with MDS and t-SNE, respectively. When compared to the clustering result in (a), we can find clusters of smaller size in MDS and t-SNE, as indicated with green circles and arrows in (b) and (c). (d), (e), and (f) show the small clusters selected from (b) or (c). When compared with MDS in (b), t-SNE in (c) finds more small clusters (e.g., cyan clusters in (b) are separated in four small clusters in (c)).

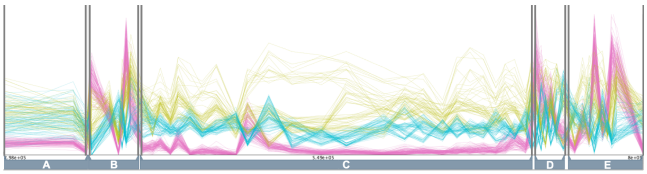


Figure 4: An example of segmentation for the temporal performance metric. The E-Divisive detects five segments (from A to E) from multiple lines.

3.3 Time-Series Segmentation with Change-Point Detection

With time-series clustering and DR, we can quickly analyze temporal patterns in performance metrics. Remaining required analysis is to understand the effect of communication patterns on the performance [34]. However, exploring and comparing communication data one-by-one at each time is a time-consuming task. To help effectively compare communication data across time, we want to obtain a temporal summary of the changes in communication data. We achieve this by segmenting time-series data and summarizing each segmentation with visual aggregates (e.g., sums or mean values). This idea is inspired by the temporal summary images (TSIs) [8], which is designed for generating narrative visualizations by summarizing the time-series data.

To segment time-series data, we can use the change point detection, which is developed for time-series analysis [2]. We use the E-Divisive method [19] because it can detect multiple change points for a set of time-series in a reasonable amount of time. Fig. 4 shows an example of segmentation with the E-Divisive method. Combining time-series segmentation with visualization for communication data (discussed in the Sect. 3.4), we provide a visual summary of changes in communication patterns in our system, described in Sect. 4.

3.4 Visualization of Communication Patterns

For parallel and distributed computing applications, exploring the communication pattern between the processes is essential. In our framework, we use the hierarchical circular visualization techniques for visual analysis of the communication patterns. As shown in Fig. 5, in a hierarchical circular visualization, lines or ribbons at the center represents the communications (e.g., network sends,

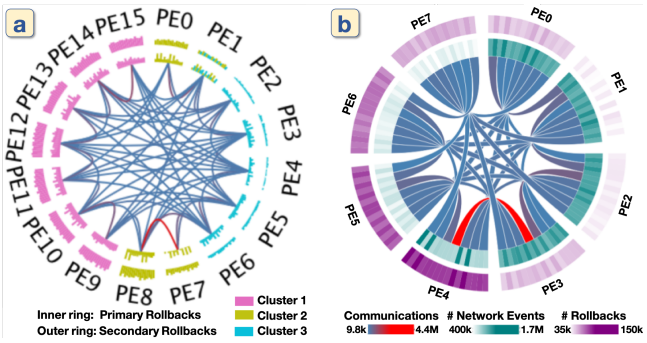


Figure 5: Hierarchical circular visualizations for showing communication patterns between entities as well as a correlation between performance metrics. While the color of the ribbons shows a value of the selected communication metric, the color of the rings can be used to encode clustering results (a) or metric values (b).

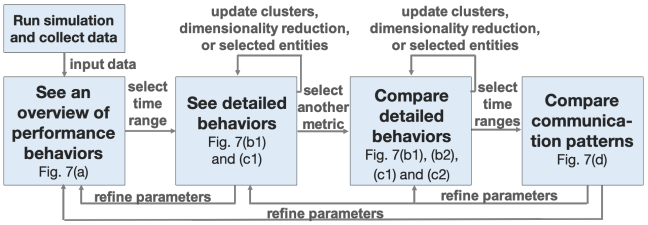


Figure 6: The analytical flow of using the system. Each step involves one or more views.

network receives, the sum of these, or the maximum value of these) between entities, while different performance metrics (e.g., primary and secondary rollbacks) can be stacked on the circumferences (or rings) to show their correlation to the communication patterns. Hierarchical aggregation is used to group the entities for organizing the circular visualization, so load balancing and distributions can be easily observed.

In addition to the communication patterns and performance metrics of the computing nodes, we also encode the time-series clustering results in the circular hierarchical visualization. Fig. 5a provides an example that we visualize both the time-series clustering results (encoded with color) and performance metrics (encoded with the size of the bars). Comparing with Fig. 5b where we only show the performance metrics, Fig. 5a allows us to correlate time-series clustering results along with the communication patterns, understanding how computing nodes with different temporal behaviors communicate with each other. This allows application programmers to gain insights on how to assign and map processes on the network to optimize performance and remove bottlenecks.

The selected communication metric is encoded with a blue-to-red colormap. In Fig. 5a, the colors of the circular bar charts are used to denote KP's cluster labels obtained with the time-series clustering method described in Sect. 3.1 and the heights of the circular bar charts are used to show the values of performance metrics. Alternatively, we can use circular heatmaps instead of circular bar charts to use color or opacity to represent the performance metrics of the KPs, as shown in Fig. 5b. As shown in these two examples, communication hot paths (shown in red lines or ribbons) and the workload distributions are clearly revealed.

4 Visual Analytics System

Based on our framework, we have developed a visual analytics system for analyzing the performance and behaviors of the ROSS PDES engine. The design of our system and user interface is based on a similar approach used in our previous work for analyzing network

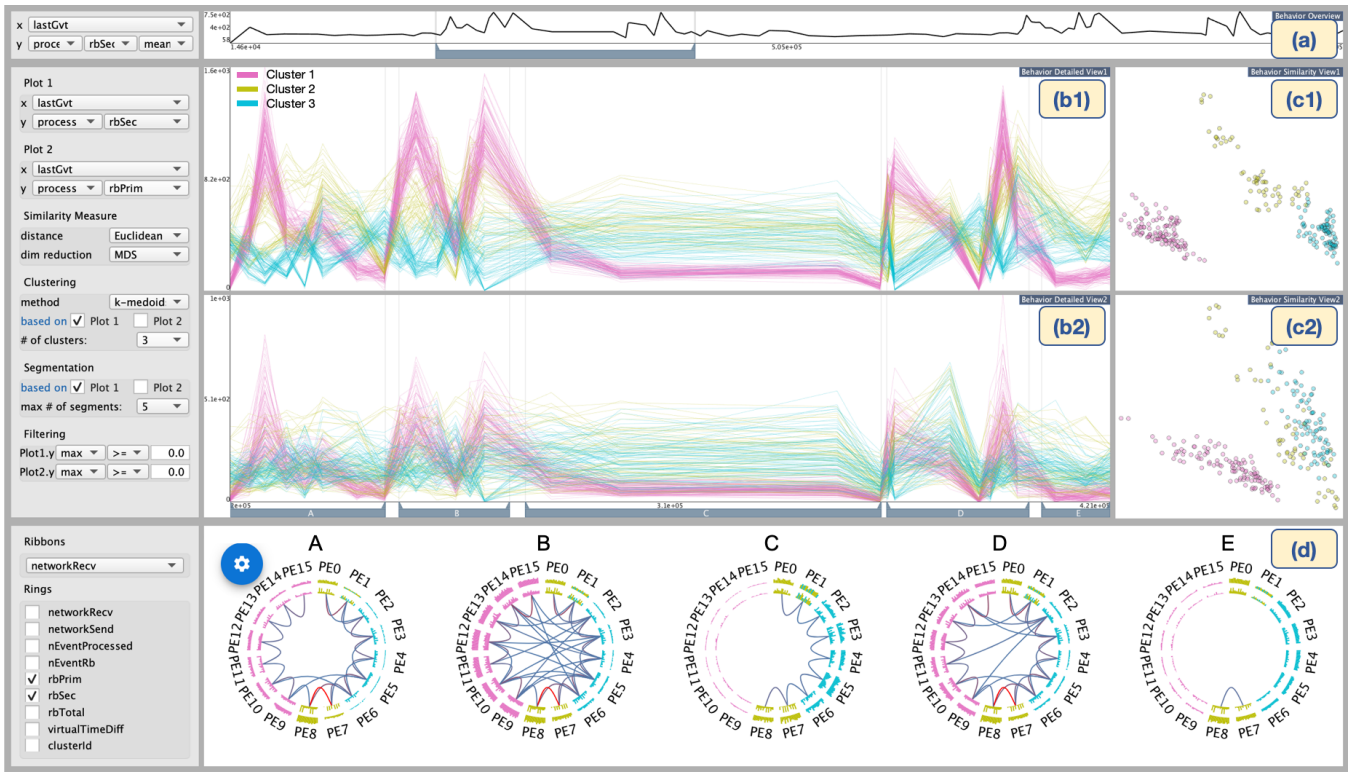


Figure 7: The user interface of the system, which contains four components: (a) the behavior overview, (b1, b2) the behavior detailed views, (c1, c2) the behavior similarity views, and (d) the communication network views. This example shows the secondary and primary rollback behaviors obtained in Sect. 5.2. (a) shows an overview (the mean in this example) of the number of secondary rollbacks over time. (b1) and (b2) show details of temporal changes of the numbers of secondary and primary rollbacks in the selected time range in (a), respectively. (c1) and (c2) show the similarity of each time-series shown in (b1) and (b2), by using the DR method. In (d), summaries of the metrics and communications of each simulation entity in the selected time ranges.

performance in supercomputers [13]. The analysis workflow and our visual analytics system are shown in Fig. 6 and Fig. 7, respectively. Our system clearly shows temporal behaviors of the performance metrics (e.g., primary, secondary rollbacks, network sends, etc) by coupling with unsupervised machine learning methods. Also, our system supports visual comparisons of not only temporal behaviors of multiple performance metrics but also communication patterns between the simulation entities (e.g., PEs and KPs) across multiple time intervals.

4.1 Visualization for Analyzing Temporal Changes

The first principal component of our system is to visualize the temporal changes in the simulation entities' performance metrics. As shown in Fig. 6, the analysis starts with an overview of the temporal changes in the behavior overview Fig. 7(a). From this overview, the user selects a time range and metrics of interest, and then reviews the details of the performance behavior in the behavior detailed view Fig. 7(b1) and the behavior similarity view Fig. 7(c1).

4.1.1 Statistical Summary of Performance Behaviors

To help the user find a performance metric and a time range of his/her interest, we provide a statistical summary of a selected performance metric for each time point across time in the behavior overview Fig. 7(a). The selected time metric (e.g., GVT, virtual time, or real time) is encoded in x-coordinates. As for y-coordinates, from the collected dataset, the user can select a performance metric (e.g., primary rollbacks and network sends) and a statistical measure (e.g., the maximum value, mean value, or standard deviation) as values for the y-direction. For example, in Fig. 7(a), GVT and the mean of secondary rollbacks of all KPs are selected for x- and y-coordinates, respectively. This view is also 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. 7(a), the time range where the mean of the number of secondary rollbacks is increasing is selected.

4.1.2 Detailed Visualization of Performance Behaviors

The performance behavior of each computing node or process in the selected time range from the behavior overview is visualized in the behavior detailed view, as shown in the Fig. 7(b1). Similarly with the behavior overview, x- and y-coordinates represent the selected time and performance metric values, respectively. For example, in Fig. 7(b1), the number of secondary rollbacks for each KP's is visualized. Because many polylines would be drawn (e.g., 256 polylines in Fig. 7(b1)), without adequate visual support, finding interesting patterns from these polylines is difficult. To address this, our system allows users to select a clustering method described in Sect. 3.1, with the number of clusters and a similarity measure from the settings, placed on the left-hand side of the behavior detailed views. We select categorical colors, each of which has enough saturation to recognize the differences of each color line with a narrow width. Furthermore, the behavior detailed view and similarity view share the color scheme to correspond the time-series with the clustering results.

4.1.3 Visualization of Similarities

While the behavior detailed view Fig. 7(b1) shows behaviors of performance metrics for each simulation entity, it is difficult to convey

the dissimilarity of each behavior in detail. By using DR methods, we visualize the dissimilarity of the behaviors in the behavior similarity view, as shown in Fig. 7(c1). The clustering methods, described in Sect. 4.1.2, are effective for grouping the behaviors in a macro sense; however, it would not be enough to find the patterns that occurred in the small set of simulation entities (e.g., outliers and anomaly behaviors). Also, while the hierarchical clustering method can inform the potential subgroups within each cluster if we visualize its clustering dendrogram, *k*-means and *k*-medoids clustering cannot provide the subgroup information. The DR-based visualization can help find these patterns (outliers, anomalies, and subgroups). The behavior similarity view in Fig. 7(c1) shows a result obtained by applying the DR for the behaviors visualized in the corresponding behavior detailed view Fig. 7(b1).

4.2 Visual Comparison of Multiple Performance Metrics

In Sect. 4.1, we describe how our system help the user analyze the temporal behavior in one selected performance metric. In addition to this univariate time-series analysis, understanding the relationships between multiple performance metrics is necessary to know how we can achieve better PDES performance. For example, to understand the rollbacks, we need to know why rollbacks happened (i.e., the cause of the rollbacks) and what occurs after the rollbacks (i.e., the effect of the rollbacks).

Our system can be useful for visually comparing between the temporal behaviors of multiple performance metrics, shown in Fig. 7 where the detailed performance behaviors (b1 and b2) and their similarities (c1 and c2) are presented together. To make the comparison between two different metrics easier, the same color that represents the cluster label is used for the corresponding simulation entities (i.e., lines in (b1) and (b2) or points in (c1) and (c2)). The user can select which behavior detailed view will be used for clustering using the settings panel, placed on the left-hand side of the behavior detailed views. Additionally, if the user wants to cluster the network behaviors based on multiple metrics (e.g., the numbers of primary and secondary rollbacks), the clustering methods and similarity measures described in Sect. 4.1.2 can be used to support multivariate time-series data. In this case, the system processes all metrics on a scale between 0 and 1.

Moreover, to support comparison within a subset of the simulation entities, our system provides multiple selection 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 a right-mouse click. Additionally, in the behavior detailed views, the system provides a freeform selection that selects intersected lines with the freeform drawn by the user. An example of the freeform selection can be seen in Fig. 9. For the behavior similarity views, a lasso selection is available to select a subset of points. After these selections, the user can filter out the unselected lines or points. The filtered out simulation entities in one view will also be filtered out from the other views at the same time.

4.3 Visual Comparison of Communication Patterns

While all the views described above depict the performance behaviors from their time-varying aspects, we visualize a summary of temporal changes in communication patterns as the last system component.

By obtaining time segments with the change-point detection method describe in Sect. 3.3, our system visualizes the summaries of behaviors. For each time segment, we calculate mean values for each metric (e.g., network receives and primary rollbacks) for each processor (e.g., PE, KP, and LP), then depict them with the circular visualization method described in Sect. 3.4. The user can also adjust these time segments based on their observations or background information. Fig. 7(d) shows an example of the visualized result. The circular visualization results are placed from the left in the order of the five segments (indicated with the alphabets from A to E), as shown in Fig. 7(b2). This example shows the network receives between PEs as ribbons drawn in the center. Also, the KPs' primary rollbacks and secondary rollbacks are visualized in the inner and outer rings, respectively. To allow the user to compare the changes in each metric across time, for each metric, the range of the height or the heatmap used in the rings is shared across different segments. The user also can filter out the ribbons based on the metric value from the blue setting menu placed on the top left of the view.

5 Case Studies

To demonstrate the effectiveness of our data analytics and visualization methods, we use our visual analytics system to evaluate the efficiency of ROSS. ROSS can run a large-scale PDES that processes up to billions of events. Interactive analysis and visualization of the instrumented data are necessary for understanding performance issues and removing bottlenecks in order to achieve the highest possible efficiency. Here, we first provide details about the setup of our experiments, then we present three cases for showing the effectiveness of our methods for analyzing various factors that affect the performance of ROSS in simulating next-generation supercomputers.

5.1 Experiment Setup

For our experiments, we use ROSS with the Dragonfly [23] network simulation model provided by the CODES simulation framework [29]. The Dragonfly configuration that we use models a system similar to the Theta Cray XC supercomputer [3] at Argonne National Laboratory with 864 routers and 3,456 compute nodes. Each router is represented by a single LP that handles all router functionality, while each compute node is represented by two LPs—one for generating the workload and one handling packet send and receive functionality. This setup results in a total of 7,776 LPs. The workload replayed over the Dragonfly network is an MPI trace from the DoE Design Forward program of the Algebraic Multigrid (AMG) solver for unstructured mesh physics packages with 1,728 MPI ranks [11].

5.2 Analysis of PDES Performance

In this case study, we use our visual analytics system to analyze a ROSS simulation with 16 PEs, with each MPI rank associated to a PE with 16 KPs. The result is shown in Fig. 7. The result provides summaries of the simulation as well as useful insights for identifying and removing performance bottlenecks. The behavior detailed views(Fig. 7(b1) and (b2)) show the secondary and primary rollbacks, respectively. Change point detection is performed to automatically select the five salient time intervals (A to E). For each of these five time intervals, the system automatically generates a hierarchical circular visualization to show the communication patterns between PEs. Also, the numbers of primary and second rollbacks of the KPs across the PEs as the inner and outer rings of the circular visualization, respectively. The colors in the circular visualizations show the clusters of KPs from the time-series clustering results. This allows us to see the distributions of KPs with different similarities among the PEs.

From the behavior detailed views(Fig. 7(b1) and (b2)), we can see that the numbers of both primary and secondary rollbacks rise and fall in time interval A, generate two peaks in time interval B, stay low in time interval C, generate two more peaks in time interval D, and then drop to very low in the final time interval. In Fig. 7(d), the communication patterns for each of these five time intervals are visualized. We can see that the number of communication events increases as the number of rollbacks increases, and the number of communication events between PE 7 and PE 8 is significantly higher in all the time intervals with a high number of rollbacks (A, B, and D). This suggests that the communication between PE 7 and PE 8

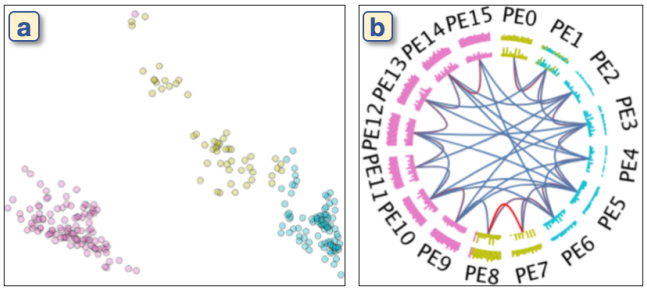


Figure 8: DR and clustering results (a) are visualized in communication view (b) to analyze the similarities of the KPs in each PE.

might have caused many of the secondary rollbacks, which indicates this is a potential performance bottleneck.

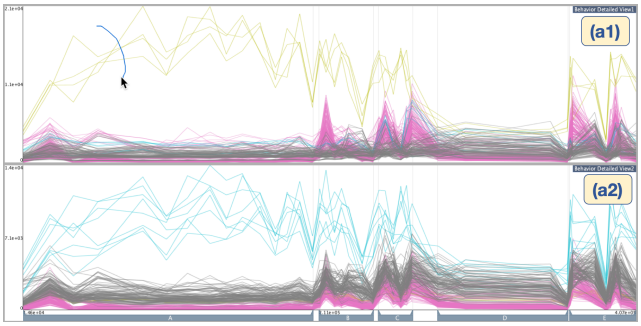
5.3 Proximity and Communication Patterns

For parallel and distributed computing, assigning processes with very different behaviors to the same computing node might result in performance degradation. In order for PDES to achieve good efficiency, the KPs in the same PE should have similar behaviors to reduce the number of rollbacks. In addition, the KPs that are communicating intensively should have similar behaviors as well, otherwise they can cause rollbacks to each other. With the time-series clustering results providing a measure of similarity for each KP over time, encoding this information in the hierarchical circular visualization allow users to visually correlate the communication patterns to the temporal behaviors based on similarity, which can provide more insights for optimizing performance. Fig. 8 shows the two views from Fig. 7 for informing users the similarity among the KPs and the communication view with color encoding the classification of the KPs, respectively. As the hierarchical circular visualization shows, the KPs in each PE have similar temporal behavior, which indicate a good assignment and mapping of KPs to the PEs. However, there are also many communication events occurred between KPs in different clusters, which can increase the number of rollbacks.

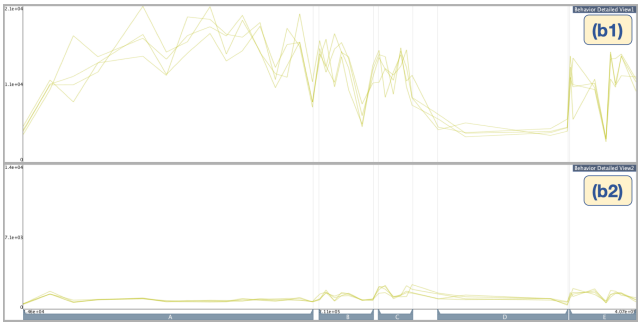
While the classification provided by the time-series clustering method only shows a fixed number of clusters (e.g. three clusters in this case), the behavior detail view (Fig. 8a) can reveal the similarity within each cluster. As we can see here, the KPs in the yellow cluster have relatively low similarity when comparing to the other two clusters. We can also see from the hierarchical circular visualization that PE 8 have higher number of rollbacks than PE 0 and 7, and having intense communication with PE8 within the same cluster can also cause performance bottlenecks. This case study demonstrates the usefulness of our visual analytics framework that includes time-series clustering result in the hierarchical circular visualization for visually correlating between temporal performance behaviors and communication patterns.

5.4 Interactive Analysis

Analyzing the time-series data of PDES simulations can help understand and correlate the performance behaviors over time, which can gain useful insights for removing bottlenecks and improving performance. Our time-series clustering methods and the user interface of our visual analytics system provide effective ways to analyze, compare, and correlate the temporal behavior of PDES. To analyze the temporal behaviors of the simulation with 16 PEs, we can compare two performance metrics over time. Fig. 9a shows time-series clustering (PAM with Euclidean distance) results for the number of network sends and receives over time, with each line representing a KP. The line charts show the peaks of the network sends and receives occur in the middle and near the end of the simulation. The colors of the lines are based on the time-series clustering results of the first



(a) Before selection



(b) After selection

Figure 9: Visual comparison of the network sends (a1) and network receives (a2) with the behavior detailed views. PAM clustering with Euclidean distance is applied based on both network receives and sends. In (a), the entities which have high network sends are selected with the freeform selection, as indicated with a blue-curved line. In (b), only the selected entities' network sends (b1) and network receives (b2) are visualized in the behavior detailed views.

selected metric (network sends), showing the subgroups of KPs with similar temporal behaviors.

Users can select any two different performance metrics for comparison and analysis. In addition, our system also supports data filtering for interactive visual analysis of details on demand, which allows a better correlation of performance metrics and temporal behaviors. From Fig. 9(a1), we can select a subset of KPs with a large number of network sends (yellow lines) and show these in Fig. 9(b1). Fig. 9(b2) shows the number of network receives for those selected KPs. We can see that KPs with a large amount of network sends in most of the simulation time have very few network receives.

Encoding time-series clustering results in the hierarchical circular visualizations also helps users to make selections for interactive analysis. As we see from the result in Fig. 8, the KPs in the yellow cluster have more intense communication and relatively low similarity comparing to other clusters. Base on this result, we can make select the yellow cluster on the behavior views (line charts) to further investigate and verify our findings. By allowing interactive analysis of different aspects of PDES or other parallel and distributed applications, more insights for optimizing performance can be provided to the users. More important, the analysis and visualization methods in our framework can support users to make selection to better facilitate interactive visual analysis.

6 Discussion and Future Work

Our visual analytics framework is designed for reasoning and interpreting multivariate time-series and communication data collected from HPC applications. As our current effort is just an initial step to develop a full framework for building visual analytics system to analyze HPC datasets, several possible extensions can be added to our framework.

First, more visualization and interaction methods can be added. Currently, we only use hierarchically circular visualizations for showing the communication patterns and correlation of performance metrics. Other visualization methods, such as matrix plot and node-link diagram, can be used with additional user interactions. For example, matrix plot can be used to create overviews of the communication network data by showing all the communication between each pair of processes. On the other hand, a subset of computing nodes or processes can be selected and visualized using node-link diagram to provide more detailed information. For visualizing the time-series clustering results and temporal behaviors, stacked area charts and stream graphs can be used instead of line charts.

In addition to interactive visualizations, progressive visual analytics can be used to support analysis of large HPC datasets. Progressive visual analytics provides useful intermediate results within a reasonable latency even when the computational cost to complete entire calculations is too high. An advantage of using progressive visual analytics is that we can support the analysis and visualization of streaming data [33], which can be used to enable real-time monitoring and analysis. However, converting our existing workflow for streaming data is challenging, especially for multivariate time-series data. One of the major challenges is how we show important changes or meaningful patterns with a low visual cognitive load since available data is constantly updated [24]. To address such issues, we can use Approximated-tSNE [33] instead of Barnes-Hut t-SNE [41] for dimensionality reduction, and incremental time-series clustering methods [38] instead of conventional k-means clustering. However, PCA-based approaches still suffer from significant false alarms, as they are highly sensitive to changes in any features. The actual projection of points in DR view using incremental PCA methods is highly sensitive to changes in features (e.g., indeterminant sign flipping [7, 40]), thereby causing a lot of visual changes at every time step.

Furthermore, we also plan to leverage in situ techniques [27] for performing data analysis on the computing nodes running the HPC applications. For example, we can perform in situ data processing and visualization within the PDES process. With in situ data processing, the simulation only needs to stream the analysis results to the visualization system, which can significantly reduce the requirement of network bandwidth. For example, we can perform progressive change point detection [6] in situ and send only the data associated with important time intervals instead of logging data for every time step during the simulation. As PDES already leverages distributed computing, combining in situ data processing and progressive analytics can be a scalable solution for real-time monitoring and visualization of the large-scale PDES.

Our visual analytics framework is already useful for analyzing HPC applications, and we have demonstrated its effectiveness with case studies on PDES. Besides application-level data, we plan to add support for analyzing system or hardware-level data (e.g., CPU utilization and memory usage). We also plan to enable co-analysis of application and hardware-level data for providing better support for performance analysis and optimization. With these planned extensions, our framework can be more useful and robust for building visual analytics solutions for a large class of HPC applications.

Acknowledgments

This research has been sponsored in part by the U.S. Department of Energy through grants DE-SC0007443, DE-SC0012610, and DE-SC0014917.

References

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

[2] S. Aminikhanghahi and D. J. Cook. A survey of methods for time series change point detection. *Knowledge and Information Systems*, 51(2):339–367, 2017.

[3] Argonne Leadership Computing Facility. Theta. `https://www.alcf.anl.gov/theta`. Accessed: 2017-12-11.

[4] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, pages 359–370. AAAI Press, 1994.

[5] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer. Analyzing network health and congestion in dragonfly-based supercomputers. In *Proc. IEEE Parallel and Distributed Processing Symp.*, pages 93–102, 2016.

[6] A. Bifet, R. Gavaldà, G. Holmes, and B. Pfahringer. Dealing with change. In *Machine Learning for Data Streams: With Practical Examples in MOA*, pages 67–84. MIT Press, 2018.

[7] R. Bro, E. Acar, and T. G. Kolda. Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 22(2):135–140, 2008.

[8] C. Bryan, K.-L. Ma, and J. Woodring. Temporal summary images: An approach to narrative visualization via interactive annotation generation and placement. *IEEE Trans. on Visualization and Computer Graphics*, 23(1):511–520, 2017.

[9] C. D. Carothers, D. Bauer, and S. Pearce. Ross: A high-performance, low-memory, modular time warp system. *Journal of Parallel and Distributed Computing*, 62(11):1648–1669, 2002.

[10] C. D. Carothers, K. S. Perumalla, and R. M. Fujimoto. Efficient optimistic parallel simulations using reverse computation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 9(3):224–253, 1999.

[11] *Co-design at Lawrence Livermore National Laboratory*. *Algebraic Multigrid Solver (AMG)*. `https://computation.llnl.gov/projects/co-design/amg2013`. Accessed: 2019-3-8.

[12] T.-c. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[13] T. Fujiwara, J. K. Li, M. Mubarak, C. Ross, C. D. Carothers, R. B. Ross, and K.-L. Ma. A visual analytics system for optimizing the performance of large-scale networks in supercomputing systems. *Visual Informatics*, 2(1):98–110, 2018.

[14] 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. on Visual Analytics Science and Technology*, pages 59–70, 2017.

[15] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, 1996.

[16] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[17] K. E. Isaacs, P.-T. Bremer, I. Jusufi, T. Gamblin, A. Bhatele, M. Schulz, and B. Hamann. Combing the communication hairball: Visualizing parallel execution traces using logical time. *IEEE Trans. on Visualization and Computer Graphics*, 20(12):2349–2358, 2014.

[18] K. E. Isaacs, A. Giménez, I. Jusufi, T. Gamblin, A. Bhatele, M. Schulz, B. Hamann, and P.-T. Bremer. State of the art of performance visualization. In *Proc. Eurographics Conf. on Visualization*, pages 141–160. Eurographics-European Association for Computer Graphics, 2014.

[19] N. A. James and D. S. Matteson. ecp: An R package for nonparametric multiple change point analysis of multivariate data. *Journal of Statistical Software*, 62(i07), 2015.

[20] D. R. Jefferson. Virtual time. *ACM Trans. on Programming Languages and Systems (TOPLAS)*, 7(3):404–425, 1985.

[21] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, volume 344. John Wiley & Sons, 2009.

[22] J. Kim, W. J. Dally, S. Scott, and D. Abts. Technology-driven, highly-scalable dragonfly topology. In *Proc. Int. Symp. on Computer Architecture*, pages 77–88. IEEE, 2008.

[23] J. Kim, W. J. Dally, S. Scott, and D. Abts. Technology-driven, highly-scalable Dragonfly topology. In *Proc. Int. Symp. on Computer Architecture*, pages 77–88, 2008.

[24] M. Krstajić and D. A. Keim. Visualization of streaming data: Observing

change and context in information visualization techniques. In *IEEE Int. Conf. on Big Data*, pages 41–47, 2013.

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

[26] 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. on Cluster Computing*, pages 193–203, 2017.

[27] K.-L. Ma, C. Wang, H. Yu, and A. Tikhonova. In-situ processing and visualization for ultrascale simulations. In *Journal of Physics: Conference Series*, volume 78, page 012043, 2007.

[28] P.-F. Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(2):306–318, 2009.

[29] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns. Enabling parallel simulation of large-scale hpc network systems. *IEEE Trans. on Parallel and Distributed Systems*, 28(1):87–100, 2017.

[30] C. Muelder, B. Zhu, W. Chen, H. Zhang, and K.-L. Ma. Visual analysis of cloud computing performance using behavioral lines. *IEEE Trans. on Visualization and Computer Graphics*, 22(6):1694–1704, 2016.

[31] W. E. Nagel, A. Arnold, M. Weber, H.-C. Hoppe, and K. Solchenbach. VAMPIR: Visualization and analysis of MPI resources. 1996.

[32] D. M. Nicol and J. Liu. Composite synchronization in parallel discrete-event simulation. *IEEE Trans. on Parallel and Distributed Systems*, 13(5):433–446, 2002.

[33] N. Pezzotti, B. P. Lelieveldt, L. van der Maaten, T. Höllt, E. Eisemann, and A. Vilanova. Approximated and user steerable tSNE for progressive visual analytics. *IEEE Trans. on Visualization and Computer Graphics*, 23(7):1739–1752, 2017.

[34] C. Ross, C. D. Carothers, M. Mubarak, P. Carns, R. Ross, J. K. Li, and K.-L. Ma. Visual data-analytics of large-scale parallel discrete-event simulations. In *Proc. Int. Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, pages 87–97. IEEE, 2016.

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

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

[37] C. Sigovan, C. W. Muelder, and K.-L. Ma. Visualizing large-scale parallel communication traces using a particle animation technique. *Computer Graphics Forum*, 32(3pt2):141–150, 2013.

[38] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. De Carvalho, and J. Gama. Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1):13, 2013.

[39] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[40] C. Turkay, E. Kaya, S. Balcisoy, and H. Hauser. Designing progressive and interactive analytics processes for high-dimensional data analysis. *IEEE Trans. on Visualization and Computer Graphics*, 23(1):131–140, 2017.

[41] L. van der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[42] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[43] R. Xu and D. C. Wunsch. Survey of clustering algorithms. *IEEE Trans. on Neural Networks*, 16(3):645, 2005.