The Journal of Supercomputing
https://doi.org/10.1007/s11227-021-03724-5

HiperView: real-time monitoring of dynamic behaviors of high-performance computing centers

Tommy Dang[1] · **Ngan Nguyen**[1] · **Yong Chen**[1]

Accepted: 2 March 2021

Abstract

This paper presents *HiperView*, a visual analytics framework monitoring and characterizing the health status of high-performance computing systems through a RESTful interface in real time. The primary objectives of this visual analytical system are: (1) to provide a graphical interface for tracking the health status of a large number of data center hosts in real-time statistics, (2) to help users visually analyze unusual behavior of a series of events that may have temporal and spatial correlation, and (3) to assist in performing preliminary troubleshooting and maintenance with a visual layout that reflects the actual physical locations. Two use cases were analyzed in detail to assess the effectiveness of the *HiperView* on a medium-scale, *Redfish*-enabled production high-performance computing system with a total of 10 racks and 467 hosts. The visualization apparatus has been proven to offer the necessary support for system automation and control. Our framework's visual components and interfaces are designed to potentially handle a larger-scale data center of thousands of hosts with hundreds of various health services per host.

Keywords High-performance computing · Data center · RESTful API · Nagios Core · Redfish · Baseboard Management Controller (BMC) · HPC visualization · Radar charts · Scatterplot · Visual features · Heatmap · Boxplots · Time-series data analysis · Multidimensional data visualization

✉ Tommy Dang
tommy.dang@ttu.edu

Ngan Nguyen
ngan.v.t.nguyen@ttu.edu

Yong Chen
yong.chen@ttu.edu

[1] Texas Tech University, Lubbock, TX, USA

Published online: 29 March 2021

1 Introduction

The high-performance computing (HPC) system has shown its applicability in many areas [15], such as chemical simulations, physics simulations, social science, among other financial/scientific applications. Supercomputing experts reported that the difficulties of hierarchical data management, better power monitoring and control, and standard interfaces for monitoring should be addressed [1]. In response to these needs, the Intelligent Platform Management Interface (IPMI) has been developed and widely adopted to monitor HPC systems in many major manufacturers such as Dell, HP, IBM, and Lenovo. This protocol allows system administrators to control over remotely deployed servers. This permission, however, is exposed to vulnerability over a remote network attacking [32]. To tackle this issue, a standard specification and schema for server configuration, called *Redfish* [29], were proposed by the Distributed Management Task Force (DMTF) in 2015. DMTF's Redfish API is an open industry standard specification and schema designed to meet end-user expectations for simple, modern and secure management of scalable platform hardware. Specifically, Redfish is an embedded firmware web server that provides clients with simple, secure management by improving security and reliability to Baseboard Management Controller (BMC). Even though *Redfish* has been proven as a practical approach for system management, this embedded firmware web server is still challenging to use as the system administrators have to monitor hosts using Redfish API capabilities via command lines. Nagios Core [5] allows fetching data via Redfish API and provides basic listing views of the hosts and services. The simple interface makes it infeasible for the administrators to observe a holistic monitoring view (in terms of spatial and temporal perspectives) of the entire data center. Thus, a detailed investigation must be done manually (filtering an enormous number of records) by system admins.

Our research fulfills the gaps by providing a framework with visual encoding strategy can be extended to handle large data center of thousands of hosts and hundreds of computer health dimensions. Our contributions thus are:

- We propose the visual designs for high-dimensional health services of a large number of computers in HPC centers. For example, we create *Bundling Radar Chart* which groups hosts with similar patterns of health status to provide a high-level overview of the current system status and the significant clusters.
- We develop the web visual framework to enable real-time monitoring through the integration with Nagios Core, InfluxDB, and Redfish API.
- To show the effectiveness of our approach, we demonstrate our visualization on a high-performance computing system at Texas Tech University with a total of 10 racks and 467 hosts with ten health dimensions per host.

The rest of this paper is organized as follows: We summarize related research in Sect. 2 and then present the visual interface designs in Sect. 3. We discuss the characteristics of each visual component in Sect. 4, describe supported user

interactions in Sect. 5, and illustrate the feasibility of the visual interface on real-world use cases in Sect. 6. Finally, the conclusion and future research direction are presented in Sect. 8.

2 Related work

This section focuses on the most related monitoring tools that are used in high-performance computing systems. PARMON [8], a lightweight monitoring system, was built to monitor and control Solaris and Linux-based clusters in the late 1990s. The GUI-based client allows end-users to capture data visually in real time. However, admins were not able to perform in-depth analysis among multiple hosts regarding spatial and temporal perspectives. Ganglia [24] is an open-source PHP-based web front-end interface proposed by Massie et al. to help administrators to assemble data employing dynamic, powerful web pages. The authors utilized comparative charts to give an insight into CPU usage, memory usage, disk usage, network statistics, and concurrent running processes. The broadly used XML advances are leveraged for information representation. The upside of ongoing responsiveness on the web front-end, however, leads to high latency because of the extent of the XML tree. Accordingly, Ganglia is not reasonable on the less powerful machines.

Nagios [5] is a commonly used industry tool for HPC infrastructure monitoring, including hosts and associated hardware components, networks, storage, services, and applications. However, there are some issues with traditional Nagios, including:

- Nagios requires human intervention for the definition and maintenance of remote hosts configurations in Nagios Core.
- Nagios requires Nagios Remote Plugin Executor on Nagios Server and each monitored remote host.
- Nagios mandates Nagios Service Check Acceptor (NSCA) on each monitored remote host.
- Nagios also requires checking specific agents (e.g., SNMP) on each monitored remote host.

In addition to Nagios Core, Nagios provides a basic web interface easily accessible to the core monitoring engine. To follow an issue, however, it is a time-consuming task for system administrators to navigate through pages of reports on hosts, services, and status [27]. Even though basic filtering operations are given, system status overview, which is valuable to explore correlation in terms of temporal and spatial issues, can be lost [3]. Figure 1 shows an example of 467 nodes listed circularly: green for non-critical health status, red for critical health status, and black for unknown. In our framework, nodes are listed in two-dimensional space to mimic the physical spatial layout of the HPC center. Moreover, we keep track of historical data and highlight a sudden change in all components' health status for system debugging.

CHReME [26] provides a web-based interface for monitoring HPC resources that took non-expert away from conventional command lines. This tool, however,

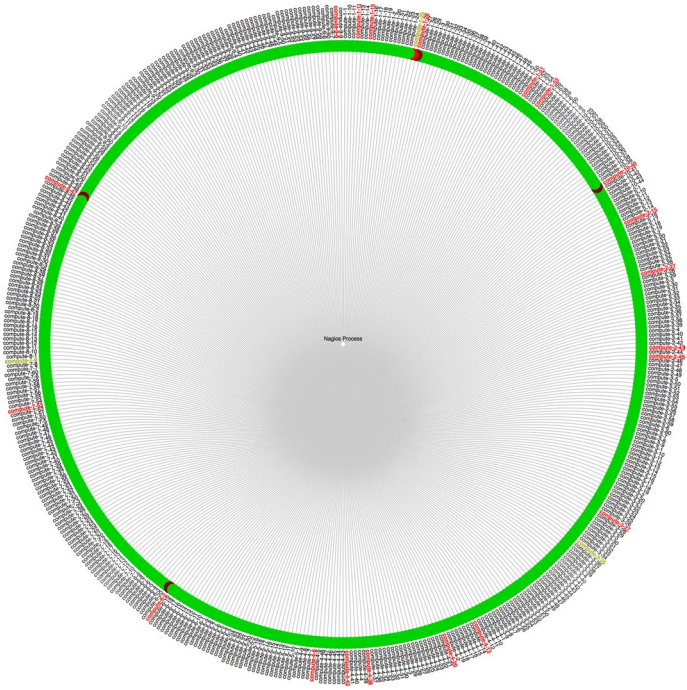

Fig. 1 The visual interface of Nagios Core where hosts are simply listed on a ring (no particular order). Green are healthy nodes, while red are computing nodes having some issues

focuses on basic tasks, which can also be found on the Nagios engine. The most similar approach to our visual analytic tool is Amazon CloudWatch [20], which gives end-users a web service to collect, view, and analyze pre-defined metrics in the form of logs, metrics, and events. Clients can define the threshold to alarms, visualize logs/metrics besides each other, and take automated actions. However, the apparatus's primary disadvantage is that it has a couple of measurements and is just relevant to Amazon cloud assets.

Splunk [9] is another software platform for mining and investigating log data for system analysts, whose most significant advantage is the capability to work with multiple data types (e.g., CSV, JSON, or other formats) in real time. It has been used and shown consistent performance in the study [36, 40]. However, Greenberg and Debardeleben [19] pointed out that Splunk was not feasible for searching a vast amount of data generated every day (e.g., hundreds of gigabytes of data) due to slow performance. Grafana [16] provides a vibrant interactive visualization dashboard that enables users to view metrics via a set of widgets (e.g., text, table, temporal data). Grafana defines a placeholder (i.e., arrays) that automatically generates widgets based on their values. This is also a limitation of Grafana: customized visualizations (such as parallel coordinates [30] and scatterplot matrices [38] for analyzing high-dimensional data) are not supported. This visualization package has been used in [6, 19] due to its multiple datastore features.

These tools mentioned above are helpful to some degree, contingent upon the necessities of a client. As pointed out by William Allcock [1], the administration monitoring tools often give far more usefulness than client's needs, or even worse, it powers clients to utilize the tools in a way that it was not intended to. Therefore, it is important to formulate research questions and enable investigators to systematically identify and examine a generic set of dimensions of the research problem [21]. The next sections discuss our research motivations and goals in monitoring HPC centers.

3 The *HiperView* approach

The primary objective of *HiperView* is to provide a visual web interface aiming to address the research goals, which were repeatedly refined in a weekly meeting with the HPC domain-experts:

- Display the real-time and historical data hosts and their health services (e.g., temperature, fan speed, memory usage). The visualization should be able to handle a large number of hosts and long time series.
- Highlight correlations between HPC health variables such as temperature vs. power consumption [13].
- Detect unusual behaviors (such as a sudden increase in temperatures). In particular, a threshold for a particular health status can be set so that an alarm can be triggered if the limit is reached [11].
- User control: Allows users to select the hosts, health services, and thresholds to define subjects/features of interest for in-depth investigation [39]. User control and analytics are especially important when the number of health dimensions and the size of the HPC center increase. This allows users to define and narrow down a subset of metrics/hosts/time intervals for system debugging.

Overall, our visual interface design includes three main components: data processing, data visualization, and user interaction.

- **Processing input data**: The input data for the *HiperView* are retrieved from the RESTful API web interface [23] as shown in Fig. 2. End-users can manually make a *http request* to the system with predefined syntax or through a visual *http query* interface (*jsonquery.html*). All parameters of the *http request* are listed on the left panel shown in Fig. 2, the corresponding response results in json format is displayed on the right panel, along with query url on the top. Our visualization tool uses this generated *http request* to retrieve data from the system on-the-fly. The blue arrow points to the essential information (memory usage) of the requested host (the *compute-5-11*).
- **Visualization interface**: Our visualization tool contains four inter-related components as depicted in Fig. 3. The detailed description of each component will be presented in the next section.
- **User interactions**: *HiperView* supports a full range of user interactions such as brushing and linking, filtering, and zooming.

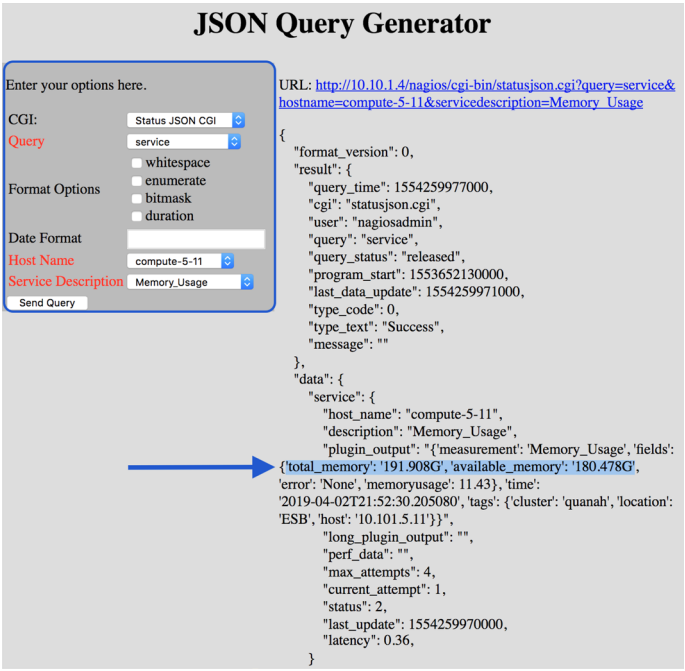

Fig. 2 Example of the RESTful API: (left) Input query for memory usage of *compute-5-11* (right) API result

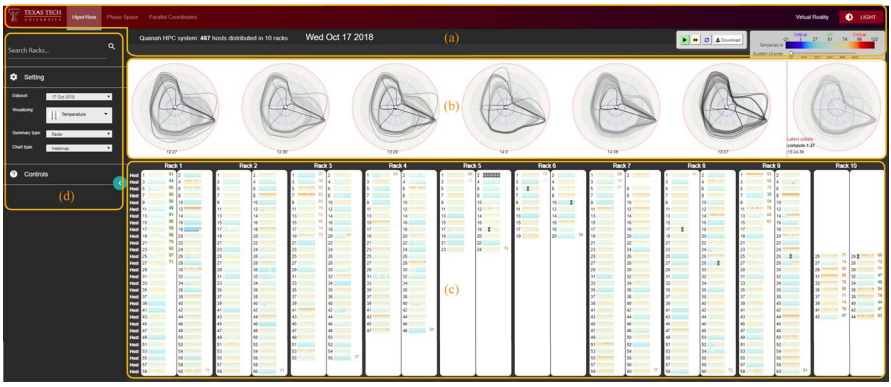

Fig. 3 Main interface of our *HiperView* visualization: **a** top panel, **b** summary view via radar charts, **c** main view using heatmaps, and **d** control panel

Visual analytics should allow system administrators to investigate the correlations among a series of events with and without a clear trace [4, 10]. Moreover, many research questions arise during the HPC data exploration process, such as what may cause a host to be overheating, why fan speed/power consumption is suddenly

increased, which in turn would allow admins to make more efficient use of the system or give a proper caution to a particular host/health services, along with their correlations. Andrienko et al. [3] suggested that information visualization systems should enable users to perform regular and analytical tasks concurrently. The recommended visual design poses a challenge due to the engagement and scarcity of end-users. Our approach was to collect all requirements, identify tasks, and then disperse them into different categories.

Thus, the *HiperView* implements five low-level visualization tasks, listed as follows:

- *Overview (T1)* Display a summary of health status [22] of all hosts over time, taking into account spatial and temporal perspectives.
- *Details-On-Demand (T2)* including details on historical events and resource usages [35].
- *Critical detection (T3)* Highlight an alarm and the critical threshold on a given measurement (e.g., *CPUs temperature*) on a selected host [2].
- *Detect the correlations (T4)* Summarize the relationships of system parameters such as CPU load, fan health, and memory usage [28].
- *Investigate the dependency across hosts (T5)* Represent the dependency of system parameters among neighboring hosts resided in racks.

4 The *HiperView* architecture

4.1 Data collection component

Our data are collected through the MonSTer framework [23] we have developed. It utilizes out-of-band measurements retrieved via Baseboard Management Controllers (BMCs) and in-band measurements accessed through the job scheduler (Univa Grid Engine, for example). The collected data were managed and stored in a time-series data, InfluxDB. We organize the collected metrics by data sources and category. For example, node-level power usage is stored in *Power* table; CPU temperature, inlet temperature, etc., are stored in *Thermal* table.

The data collection interval is restricted by the response of the BMC and the job scheduler's state update time, which are 55 s and 40 s on our platform, respectively. Therefore, our current implementation collects the data at a reasonable interval of 60 s to ensure that BMC metrics are retrieved even encountering network fluctuations and to collect job accounting information whenever possible.

4.2 Data visualization component

Figure 3 shows a schematic overview of our visual framework. Box (a) contains the system information and color legend, box (b) displays the summary of all hosts and available health services in the system, box (c) shows the detailed view of the 467-node Quanah cluster, Texas Tech University, at 12 PM on Wednesday, September

26, 2018, and box (d) contains interface settings (such as health service selection, visual encodings). The following sections are dedicated to describing each component in detail.

4.3 The top panel

The top panel contains HPC system information such as the number of hosts, number of racks, and the current timestamp. As replaying is a desirable feature for system debugging, *HiperView* provides historical simulation capability via play, forward, and refresh buttons. The color legends on the right of Fig. 3b is defined based on the critical thresholds of the selected health service. Below the color scale, users can filter out some abnormal behavior of health data by setting the "Sudden change" slider (for the selected health service). The range of each HPC health services is defined as follows:

1. **CPU temperature**
   - Lower threshold *Critical*: 3 °C or 37.4 °F
   - Lower threshold *Non-Critical*: 8 °C or 46.4 °F
   - Upper threshold *Critical*: 98° C or 208.4 °F
   - Upper threshold *Fatal*: 98 °C or 208.4 °F
   - Upper threshold *Non-Critical*: 93 °C or 199.4 °F
2. **Fan speed** in Rounds Per Minute (RPM):
   - Lower threshold *Critical*: 1050 RPM
   - Lower threshold *Fatal*: 1050 RPM
   - Upper threshold *Critical*: 17850 RPM
   - Upper threshold *Fatal*: 17850 RPM
3. **Inlet temperature**:
   - Lower threshold *Critical*: 0 °C or 32.0 °F
   - Lower threshold *Fatal*: 0 °C or 32.0 °F
   - Lower threshold *Non-Critical*: 3 °C or 37.4 °F
   - Upper threshold *Critical*: 47 °C or 116.6 °F
   - Upper threshold *Fatal*: 47 °C or 116.6 °F
   - Upper threshold *Non-Critical*: 42 °C or 107.6 °F
4. **Memory usage**:
   - *OK*: Memory usage $\leq 95\%$
   - *Warning*: $95\% <$ Memory usage $< 99\%$
   - *Critical*: Memory usage $\geq 99\%$
5. **Power consumption** (in Watts): .
   - *OK*: Power consumption $\leq$ 200 W
   - *Warning*: 200 W <Power consumption < 250 W
   - *Critical*: Power consumption $\geq$ 250 W

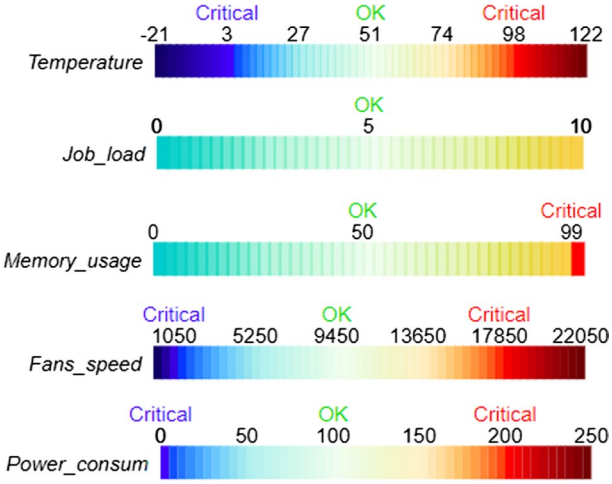

Fig. 4 Customized color scales and critical thresholds for different HPC health metrics

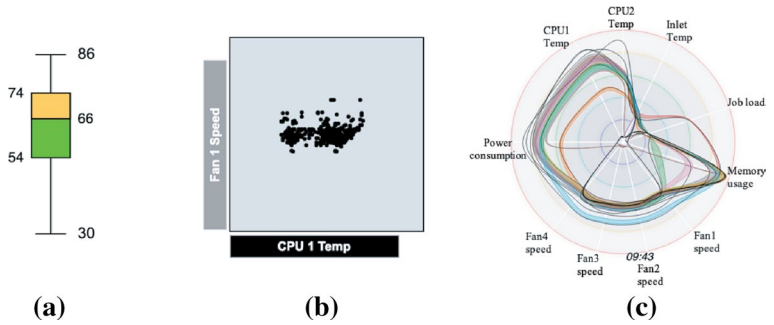

Fig. 5 The example overview of our *HiperView* for: **a** univariate analysis (boxplot), **b** bivariate analysis (scatterplot) and **c** multivariate analysis (radar plot)

Color spectra for different health status data are depicted in Fig. 4. Notice that various color transfer functions (from blue to red) are applied for different HPC health dimensions. For example, the *Memory usage* does not have a lower limit. The only critical threshold is on the upper limit at 99%. This explains the sudden turning from yellow to red at 99% on the *Memory usage*.

4.4 Summary view

The summary panel provides an overview of all computing nodes at various time steps. Figure 5 shows the three chart types that users can select to display in the summary view. The charts convey a different number of dimensions from univariate, bivariate, to multivariate analysis (the visualization task **T4**).

Boxplot Boxplot is a standardized way of displaying the distribution of data based on a five-number summary: minimum, first quartile, median, third quartile, and

maximum. It also displays the outliers which are not in the range from minimum to maximum. For each timestamp, the selected service (such as *power consumption*) of all hosts in the HPC center is consolidated and dispersed through the use of boxplot. This summary representation allows administrators to grasp the overall range of the selected variable of the system at the current timestamp.

Scatterplot To analyze the pairwise correlation, scatterplots and their visual features [14] are applied to highlight unusual distributions. Figure 5b shows an example of 2D scatterplot of *CPU1 temperature* vs. *Fan1 speed*, each data point is a host in the HPC center. Scatterplots are designed to handle the types of doubly multivariate data [12] and to unveil unusual pairwise distributions, such as linear correlations [17], clusters [34], and outliers [37].

Radar Chart Within this chart, the multidimensional status of each host is represented by a closed curve traveling through the corresponding values on each dimension. As the number of hosts grows, similar hosts are grouped and presented as a closed band. Outliers are represented as a single black curved. The main purpose of this chart is to summarize typical groups of hosts in the system (which as similar multivariate behavior) as well as highlight abnormal behaviors (black curves). This design can be extended to handle the larger number of health dimensions by merely adding more coordinates into the circular layout [25].

4.5 Main view

Figure 6 shows the spatial distribution of 467 hosts resided in 10 racks (the visualization task **T1**), mimicking the physical location of hosts at the HPC center, Texas Tech University. In Fig. 6a, the *CPUs temperature* on each host (each row) is represented by a series of color-coded cells as defined on the top panel in Fig. 4. Users can select to change to a different health measurement. The color scale will be updated accordingly.

The two visual options that users can select to convey time series data in this main view, namely heatmap and area chart, are depicted in Fig. 6. While heatmap enables system administrators to compare the time series across different hosts easily, the area chart allows plotting many time series and highlighting changes at a glance. Notice that in Fig. 6(b), CPU temperature is plotted with regard to the baseline of 55 °F (the HPC normal room temperature), and the color spectrum is embedded directly into the area charts. These design choices allow subtle changes easily detected on a small screen area and, therefore, can be adapted to monitor large HPC systems.

The Detailed Panel A pop-up window is introduced on the mouse over to show the details of a particular host (visualization task **T2**). While the main view shows an overall temporal signature of all hosts, the detail panel allows users to unveil more detailed information of a given host by a set of line graphs. The vertical axis represents the CPU temperature (from 20 to 100 °F), while the horizontal axis represents the time. As depicted in Fig. 7, a sudden drop on both *CPUs temperature* occurs at 12 pm. We can set up an alarm to detect these sudden changes and notify system administrators for timely investigations.

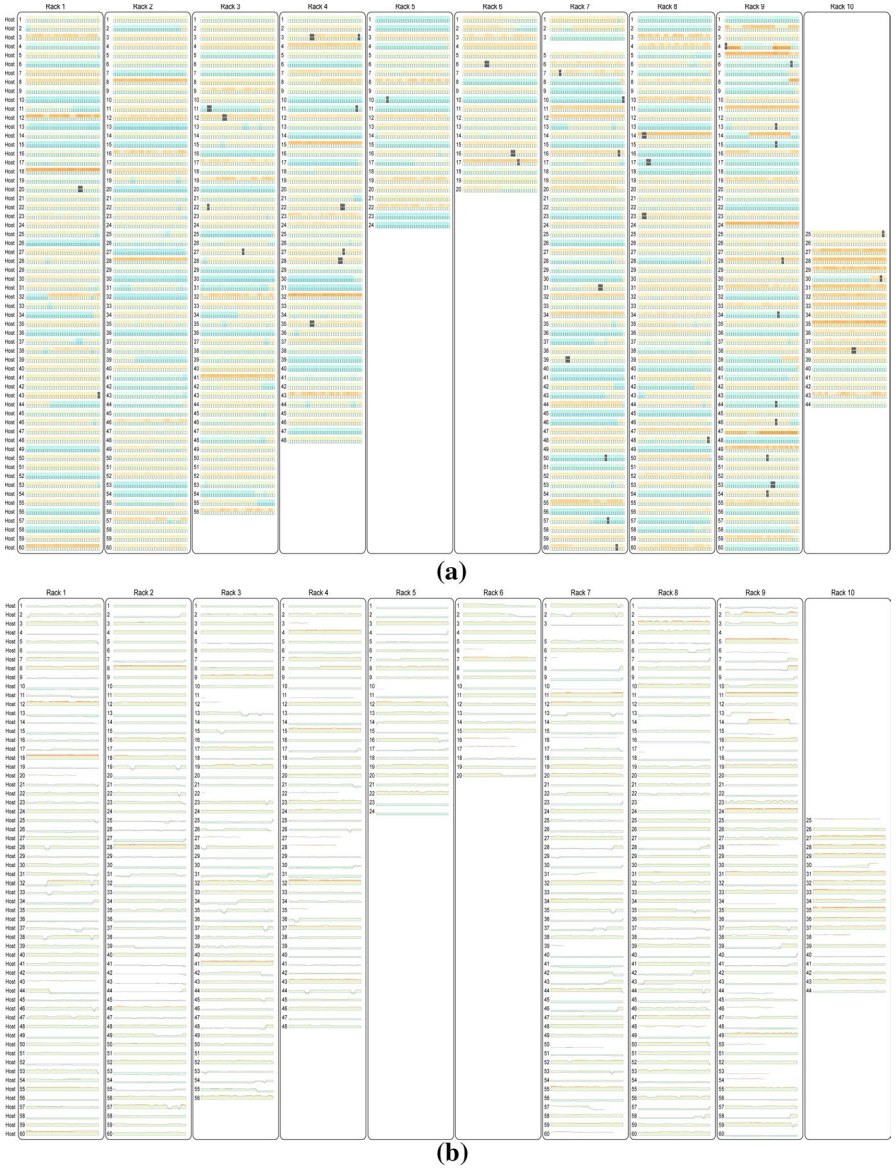

Fig. 6 Two types of visual layout for visualizing CPU temperature readings in 1 day: **a** the heat map; **b** the area chart

At the bottom of this detailed view, a radar chart is also appended to show ten health dimensions of the given host (visualization task **T4**), including CPU temperature, fans speed, memory usage, and power consumption. Using the radar chart, users can investigate the correlation among multivariate observations as well as discern the outliers [33]. The order, space, and orientation of these ten health metrics

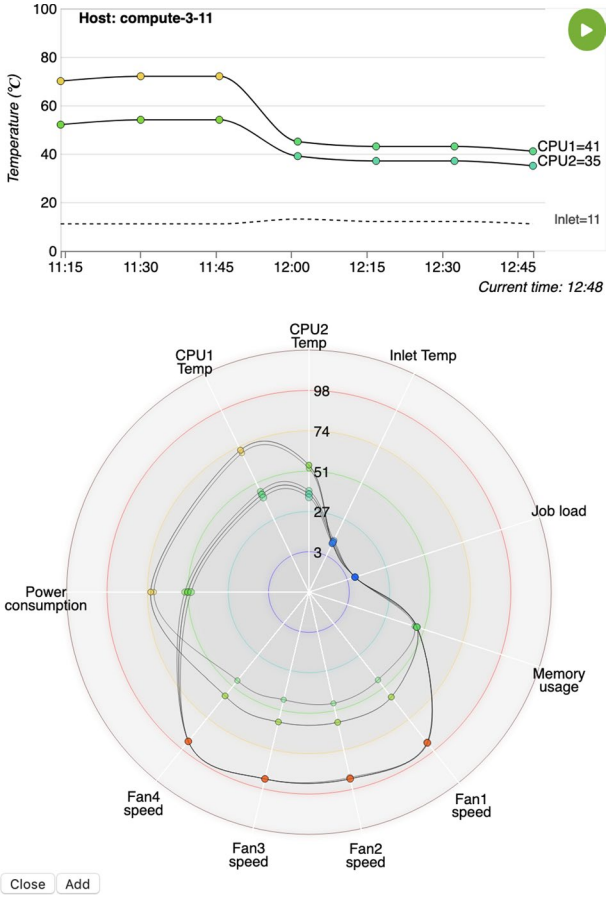

Fig. 7 The detailed view of a given host: (top) CPU temperature series (bottom) the radar chart for visualizing multidimensional heath status

on the radar chart are kept the same in this paper for simplification. However, the user can reorganize/customize the radar chart if needed. Different closed curves represent the multidimensional health status of the given host (*compute-3-11* in this example) at different timestamps. To animate the historical data over time, users can activate the *play back* button on the top right corner via a simple mouse click.

4.6 The control panel

The control panel in Fig. 3d enables system administrators to navigate through different types of HPC measurement, including *CPUs temperature*, *job load*, *memory usage*, *fans speed*, and *power consumption*. The default selection is set to a *CPUs temperature* as it is the vital metric for HPC systems. The layout of the main view can be switched between area chart and heat map by selecting "Chart type".

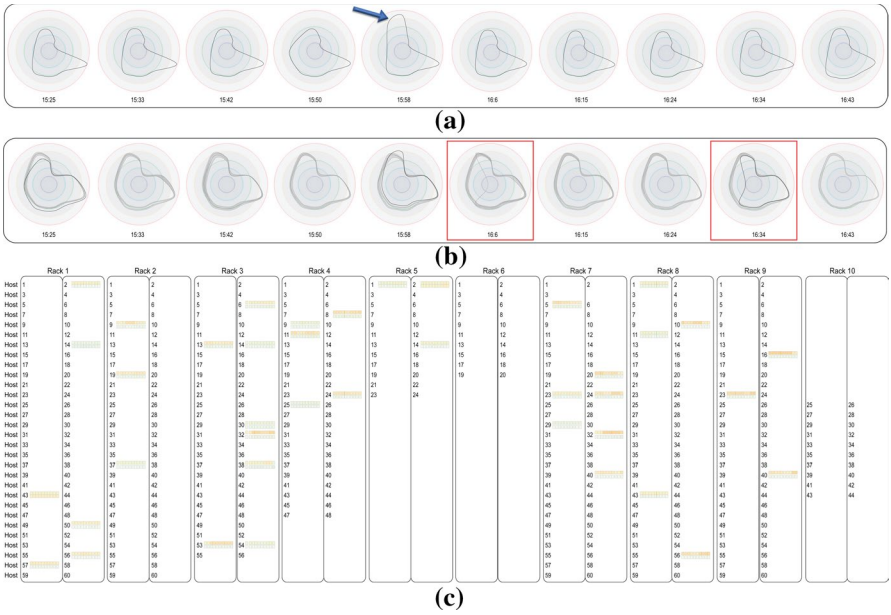

Fig. 8 Brushing and linking in our *HiperView*: **a** single-host selection, **b** multiple-host selection and **c** corresponding hosts are highlighted in the main view

Similarly, three system summary types (boxplot, radar chart, and scatterplot) can be made interchangeably via the "Summary type" drop-down list.

5 User interactions

HiperView supports a full range of user interactions [31]. **Host selection** reduces visual clutter by filtering out other hosts. For example, in Fig. 8(a), when the user selects an abnormal host from the summary radar chart, the historical data of that host on the previous cycle were also highlighted. The health status of this host changes significantly over time, reflecting through the various shapes at different timestamps. Especially at 11:58, the CPU temperatures are very high, close to the *critical* threshold (the red ring) at the blue arrow.

Cluster selection Users can also select and highlight multiple hosts as depicted in Fig. 8b. Host health status can frequently change due to several circumstances (e.g., newly assigned users, newly scheduled jobs). *HiperView* groups hosts with similar behaviors into the same cluster and summarizes the cluster behaviors by multidimensional bands. In Fig. 8b, we select the cluster on the rightmost radar chart (at 16:43) and visualize the cluster dynamics in previous snapshots. In particular, the hosts in this cluster might represent very different behavior in the past hour. For example, we can easily detect a host in this cluster that experienced low *power consumption* at the highlighted timestamps (16:06 and 16:34 at the red boxes). Users can visualize the temporal behaviors of individual hosts in the spatial distribution in

Fig. 8c, as other hosts are faded out. *HiperView* supports holding the current selection via mouse click. Users can also filter the HPC health data by using sliders or defining a subset of metrics/hosts to be focused on.

6 Use cases

HiperView is developed using JavaScript, a web browser programming language, and in particular, using the D3.js library [7]. To demonstrate the feasibility of the *HiperView* visualization, we conduct two use cases on the Quanah cluster at Texas Tech University. The Quanah cluster is comprised of 467 nodes, with Intel XEON processors providing 36 cores per node. Quanah has a total of 16,812 cores with a benchmarked total computing power of 485 Teraflops/s. The cluster is based on Dell EMC PowerEdge^{TM} C6320 servers, which are equipped with the integrated Dell Remote Access Controller providing Redfish API for accessing the monitoring metrics.

The purpose of this study was to assess whether our proposed visual interface would be able to extract and convey useful information from the data. The expected results include the ability to detect correlations and anomalies of the HPC health dimensions.

6.1 Use case 1: September 26, 2018

Figure 9 provides a snapshot of the event on September 26, 2018, at the HPC center at Texas Tech University that the *HiperView* revealed during the investigation process. As shown in Fig. 9a, the *CPU2 temperature* on *compute-3-13* (host 13 in rack 3) passed the *critical* threshold. By performing further analysis on this host, there was a clear temperature gap between CPU1 and CPU2 shown in Fig. 9c: *CPU1 temperature* is low, whereas *CPU2 temperature* almost reached 100 °F. This detailed view also shows that the temperatures of both CPUs were not reported (unreachable) by our monitoring system from 11:15 AM to 12:00 AM as there is no connecting line in this period. Similarly, unreachable status was also represented as gray cells in Fig. 9a. Other information related to this host can be analyzed via a radar chart (Fig. 9d) along with the high *CPU2 temperature* such as *fans speed* and *power consumption*. We can quickly notice that all the fans also worked to their maximal capacity (close to the red line threshold as shown in Fig. 9b). One of the explanations to this phenomenon is that the thermal issue on *compute-3-13* led to a temperature increase in the surrounding environment. As a consequence, fans of neighboring hosts had to speed up as they sense the heat. The abnormal behavior of this host can also be captured by looking at the outlier curve of the bundling radar chart in the analysis component. Figure 9d highlights the multidimensional behavior of *compute-3-13*. It was noted that *compute-3-13* was not reachable from 11:02 to 11:18 AM (as there is no services or no curves in the first two radar charts shown in Fig. 9d), which is consistent with the detailed view in Fig. 9c.

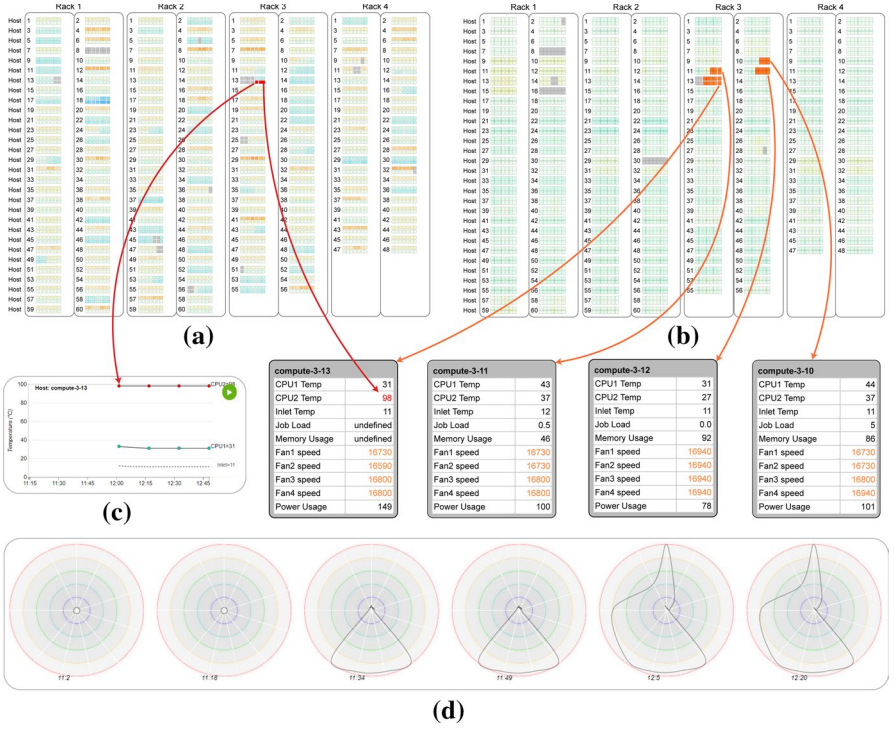

Fig. 9 The event on September 26, 2019, for the 467-node Quanah cluster, HPC center, Texas Tech University: **a** heat map of CPUs temperature, **b** heat map of Fans speed, **c** detailed view of the *compute-3-13*, **d** real-time CPUs temperature of host data *compute-3-13*, and **e** radar charts of *compute-3-13* at consecutive timestamps from 11:02 AM to 12:20 PM

We further investigate the causal relationship and discover the patterns of propagation. Specifically in Fig. 9b, in addition to the high fan speed of *compute-3-13*, other neighboring nodes (*compute-3-10*, *compute-3-11*, and *compute-3-12*) are also reported a sudden increase on *fans speed* at the very same time. The detail on the health services of these hosts is plotted at the orange arrows. In this use case, the spatial layout, together with temporal representation, provided system administrators a more detailed investigation of the health status for a subset of HPC nodes.

6.2 Use case 2: March 21, 2019

The HPC center at Texas Tech University uses chilled water to cool down the systems. At 3:20 PM on Thursday, March 21, 2019, the chill system starts delivering higher temperature water which was not enough chilled to cool down the *CPUs temperature* and finally at 4:00 PM, *CPUs temperature* reaches their critical stage. Consequently, the whole cluster was shut down by the system admins. Figure 10a shows a summary of the situation on March 21, 2019, where significant events are annotated on top of the temperature time series charts. A more detailed timeline of the situation is listed below:

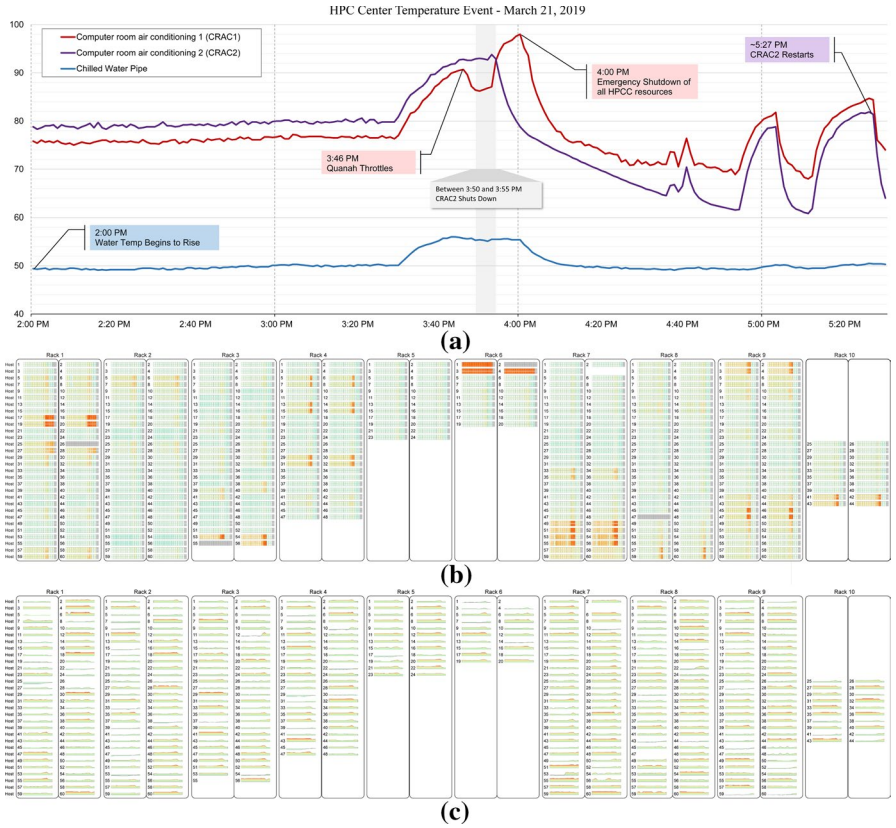

Fig. 10 The situation on March 21, 2019, for the 467-node Quanah cluster, HPC center, Texas Tech University: **a** summary of HPC chiller system and HPC cluster temperatures. **b** The *Fans speed* heatmap of our *HiperView* visualization **c** the *CPUs temperature* area charts of our *HiperView* visualization. Time expands from 3:00 to 4:10 PM of the same day

- At 2:00 PM, chilled water temperatures began to rise steadily.
- At 3:23 PM, chilled water temperatures began to rise suddenly.
- At 3:40 PM, chilled water temperature rose above 55°F (a key metric for the HPC center).
- At 3:46 PM, chilled water temperature detected above 55°F for over 5 straight minutes, which triggered the "chilled water alarm script" and notified HPCC staff by email. They then put the cluster into a low power state.
- Between 3:50 PM and 3:55 PM, CRAC[1] went into high-temperature alarm. CRAC received a remote shutdown command from Operations' monitoring software. CRAC went offline. This caused a sudden spike in temperatures.
- At 4:00 PM, the whole cluster was shut down.

[1] A computer room air conditioning (CRAC) unit is a device that monitors and maintains the temperature, air distribution, and humidity in a network room or data center.

The last two panels shown in Fig. 10 show how our *HiperView* visualization captures the entire event on March 21, 2019. In particular, Fig. 10b shows the heatmap of *Fans speed* for the time span from 3:00 to 4:10 PM. We can see that on many hosts, the fans were creasing their speeds toward 4:00 PM, returning regular rotations after that, and suddenly stopped at around 4:10 PM (represented by gray cells that the end of each series). Figure 10b shows the area charts of *CPUs temperature* during the same time interval. Sudden appearances and drops of red areas occurred around 3:15–4:00 PM. It took 45 min for *CPUs temperature* to reach a critical threshold. As a result, having sensors on the water chiller can provide some preventive actions which might avoid downtime. It was also discussed whether the monitoring framework could predict an unexpected shutdown incident with rationales.

7 Discussions and limitations

7.1 Overhead

The temperature, fan speed, and power consumption are collected through the Baseboard Management Controller, which utilizes an independent network to communicate with the data collection component. This out-of-band measurement avoids interfering with ongoing computation fabric. The job load and memory usage are collected from the scheduler accounting information, which is done only between the host running the data collection component and the cluster head node running the job scheduler. Our measurements show that the network bandwidth consumed for transmitting the monitoring data (in-band) for each node is 0.32KB/s, which is negligible compared to the management network's speed.

7.2 Data volume and security

In our project, the data volume for 1-year monitoring data of 467 nodes is about 25 GB. For larger clusters, the data volume will expand. However, since our visualization tool focuses on the visualization of real-time data (although it supports visualization of historical data), we do not have to keep all historical data and can keep the data in a reasonable size by setting the retention policies in the database.

As we mentioned in the data collection section, the data are collected through the BMC and the job scheduler, which have dedicated authentication methods. In addition, the collected data are managed by a time-series database, InfluxDB, which also has an authentication process. In addition, *HiperView* is intended to be used by system administrators, not by regular HPC users, and the data access can be restricted at the HPC center. Therefore, there are no major security concerns regarding data collection and visualization.

7.3 Limitations

HiperView heavily relies on the collected data, especially relies on the existing infrastructures. Therefore, it cannot be directly used in other HPC centers that do not have such out-of-band measurements. In order to be adopted by other HPC centers, we have to make extra efforts to adapt the interface to other existing monitoring infrastructures. Besides, the monitoring metrics cannot be retrieved within seconds, and for some relatively short duration state changes, *HiperView* cannot capture them. The *HiperView* source codes are hosted on the GitHub page of our project, located at https://idatavisualizationlab.github.io/HPCC/HiperView/demo.html.

Our *HiperView* visualization component is limited by the screen sizes and resolutions, as depicted in Fig. 6a. As the numbers of computes and time steps increase, the heatmap cells' overlapping becomes more significant. Moreover, the rendering time is another consideration as an average laptop can draw around 10K rectangles with a JavaScript Canvas at 60 frames per second (FPS) on the web browsers. This is not an issue with the stream graphs in Fig. 6b. However, a sudden event on the time series (only for a particular time step) might become less visible due to the space allocation for a time step is small. We can mitigate the visual limitations by allowing users to define and narrow down a subset of computers/time interval of interest rather than plotting the entire time series data.

8 Conclusion and future work

In this paper, we presented *HiperView* visual interface for tracking the dynamic behavior of the data center. The *HiperView* introduces multivariate analysis for HPC health services (such as using bundling radar chart), which provides a complete picture of health status for interpreting high-dimensional data and helps users explain the phenomenon of an unexpected event. More importantly, the *HiperView* can help system administrators analyze, monitor, and debug HPC system health status in real time based on the industry-standard Redfish protocol. *HiperView* supports a full range of user interactions such as brushing and linking, filtering, and zooming, as demonstrated through visual examples. The visual framework is validated on two use cases where unexpected events occurred at the HPC center, Texas Tech University.

In the future, this work can be extended by incorporating machine learning frameworks to predict the system health status. The model will be trained on historical data, make real-time predictions, and raise the alarm to the system administrator for timely actions. Our long-term research vision is to provide a holistic monitoring and controlling framework with rich visual interfaces to automate the management of highly sophisticated data centers focusing on user analytics.

Acknowledgements The authors acknowledge the High-Performance Computing Center (HPCC) at Texas Tech University [18] in Lubbock for providing HPC resources and data that have contributed to the research results reported within this paper. The authors are thankful to the anonymous reviewers for their valuable feedback and suggestions that improved this paper significantly. This research is supported

in part by the National Science Foundation under grant CNS-1362134, OAC-1835892, and through the IUCRC-CAC (Cloud and Autonomic Computing) Dell Inc. membership contribution.

References

1. Allcock W, Felix E, Lowe M, Rheinheimer R, Fullop J (2011) Challenges of hpc monitoring. In: SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, pp 1–6. https://doi.org/10.1145/2063348.2063378
2. Amar R, Eagan J, Stasko J (2005) Low-level components of analytic activity in information visualization. In: Proc. of the IEEE Symposium on Information Visualization, pp 15–24
3. Andrienko N, Andrienko G, Gatalsky P (2003) Exploratory spatio-temporal visualization: an analytical review. J Vis Lang Comput 14(6):503–541
4. Andrienko N, Lammarsch T, Andrienko G, Fuchs G, Keim D, Miksch S, Rind A (2018) Viewing visual analytics as model building. In: Computer graphics forum, vol 37. Wiley Online Library, pp 275–299
5. Barth W (2008) Nagios: system and network monitoring. No Starch Press, San Francisco
6. Betke E, Kunkel J (2017) Real-time i/o-monitoring of hpc applications with siox, elasticsearch, grafana and fuse. In: International Conference on High Performance Computing, pp 174–186. Springer
7. Bostock M, Ogievetsky V, Heer J (2011) D3 data-driven documents. IEEE Trans Vis Comput Graph 17(12):2301–2309
8. Buyya R (2000) Parmon: a portable and scalable monitoring system for clusters. Softw Pract Exp 30(7):723–739
9. Carasso D (2012) Exploring splunk. CITO Research, New York
10. Ceneda D, Gschwandtner T, May T, Miksch S, Schulz H, Streit M, Tominski C (2017) Characterizing guidance in visual analytics. IEEE Trans Vis Comput Graph 23(1):111–120. https://doi.org/10.1109/TVCG.2016.2598468
11. Dang T, Wilkinson L (2013) TimeExplorer: similarity search time series by their signatures. In: Proc. International Symp. on Visual Computing, pp 280–289
12. Dang TN, Anand A, Wilkinson L (2013) TimeSeer: scagnostics for high-dimensional time series. IEEE Trans Vis Comput Graph 19(3):470–483. https://doi.org/10.1109/TVCG.2012.128
13. Dang TN, Wilkinson L (2014) Scagexplorer: exploring scatterplots by their scagnostics. In: 2014 IEEE Pacific Visualization Symposium, pp 73–80. https://doi.org/10.1109/PacificVis.2014.42
14. Dang TN, Wilkinson L (2014) Transforming scagnostics to reveal hidden features. IEEE Trans Vis Comput Graph 20(12):1624–1632
15. Eurotech: Industry solutions (2019) https://www.eurotech.com/en/hpc/industry+solutions. Accessed 10 Oct 2020
16. Grafana: The open platform for beautiful analytics and monitoring (2019). https://grafana.com/. Accessed 10 Oct 2020
17. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3: 1157–1182. URl http://dl.acm.org/citation.cfm?id=944919.944968. Accessed 10 Oct 2020
18. HPCC: High Performance Computing Center (2021) http://www.depts.ttu.edu/hpcc/. Accessed 10 Oct 2020
19. Hugh Greenberg ND (2018) Tivan: a scalable data collection and analytics cluster (2018). In: The 2nd Industry/University Joint International Workshop on Data Center Automation, Analytics, and Control (DAAC)
20. Inc, A.: Amazon cloudwatch (2012) http://aws.amazon.com/cloudwatch/. Accessed 10 Oct 2020
21. Jia C, Cai Y, Yu YT, Tse T (2016) 5w+1h pattern: a perspective of systematic mapping studies and a case study on cloud software testing. J Syst Softw 116:206–219
22. Keim DA, Panse C, Sips M (2004) Information visualization: scope, techniques and opportunities for geovisualization. In: Dykes J (ed) Exploring geovisualization. Elsevier, Oxford, pp 1–17
23. Li J, Ali G, Nguyen N, Hass J, Sill A, Dang T, Chen Y (2020) Monster: an out-of-the-box monitoring tool for high performance computing systems. In: 2020 IEEE International Conference on Cluster Computing (CLUSTER), pp 119–129. IEEE

24. Massie ML, Chun BN, Culler DE (2004) The ganglia distributed monitoring system: design, implementation, and experience. Parallel Comput 30(7):817–840
25. Meyer M, Munzner T, Pfister H (2009) Mizbee: a multiscale synteny browser. IEEE Trans Vis Comput Graph 15(6):897–904. https://doi.org/10.1109/TVCG.2009.167
26. Misra G, Agrawal S, Kurkure N, Pawar S, Mathur K (2011) Chreme: a web based application execution tool for using hpc resources. In: International Conference on High Performance Computing, pp 12–14
27. Nguyen N, Dang T (2019) Hiperviz: Interactive visualization of CPU temperatures in high performance computing centers. In: Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning), PEARC '19. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3332186.3337959
28. Nguyen N, Hass J, Chen Y, Li J, Sill A, Dang T (2020) Radarviewer: visualizing the dynamics of multivariate data. In: Practice and Experience in Advanced Research Computing, PEARC '20, pp 555–556. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3311790.3404538
29. Organization SD (2013) Distributed management task force. https://www.dmtf.org/standards/redfish. Accessed 10 Oct 2020
30. Palmas G, Bachynskyi M, Oulasvirta A, Seidel HP, Weinkauf T (2014) An edge-bundling layout for interactive parallel coordinates. In: 2014 IEEE Pacific Visualization Symposium, pp 57–64. https://doi.org/10.1109/PacificVis.2014.40
31. Pike WA, Stasko J, Chang R, O'Connell TA (2009) The science of interaction. Inf Vis 8(4):263–274. https://doi.org/10.1057/ivs.2009.22
32. Roberts PF (2013) IPMI: the most dangerous protocol you've never heard of. https://www.computerworld.com/article/2708437/ipmi--the-most-dangerous-protocol-you-ve-never-heard-of.html. Retrieved 27 Mar 2021
33. Saary MJ (2008) Radar plots: a useful way for presenting multivariate health care data. J Clin Epidemiol 61(4):311–317
34. Seo J, Shneiderman B (2004) A rank-by-feature framework for unsupervised multidimensional data exploration using low dimensional projections. In: Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on, pp 65–72. IEEE
35. Shneiderman B (1996) The eyes have it: a task by data type taxonomy for information visualizations. In: Proceedings of the 1996 IEEE Symposium on Visual Languages, VL '96, p 336. IEEE Computer Society, Washington, DC, USA. URL http://dl.acm.org/citation.cfm?id=832277.834354
36. Stearley J, Corwell S, Lord K (2010) Bridging the gaps: Joining information sources with splunk. In: SLAML
37. Wilkinson L (2017) Visualizing big data outliers through distributed aggregation. IEEE Trans Vis Comput Graph 24(1):256–266
38. Wilkinson L, Anand A, Grossman R (2005) Graph-theoretic scagnostics. In: Proceedings of the IEEE Information Visualization 2005, pp 157–164. IEEE Computer Society Press
39. Wilkinson L, Anand A, Grossman R (2006) High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions. IEEE Trans Vis Comput Graph 12(6):1363–1372
40. Zadrozny P, Kodali R (2013) Big data analytics using Splunk: deriving operational intelligence from social media, machine data, existing data warehouses, and other real-time streaming sources. Apress, New York