Eurographics Symposium on Parallel Graphics and Visualization (2018)
H. Childs, F. Cucchietti (Editors)

La VALSE: Scalable Log Visualization for Fault Characterization in Supercomputers

Hanqi Guo, Sheng Di, Rinku Gupta, Tom Peterka, and Franck Cappello†

Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA

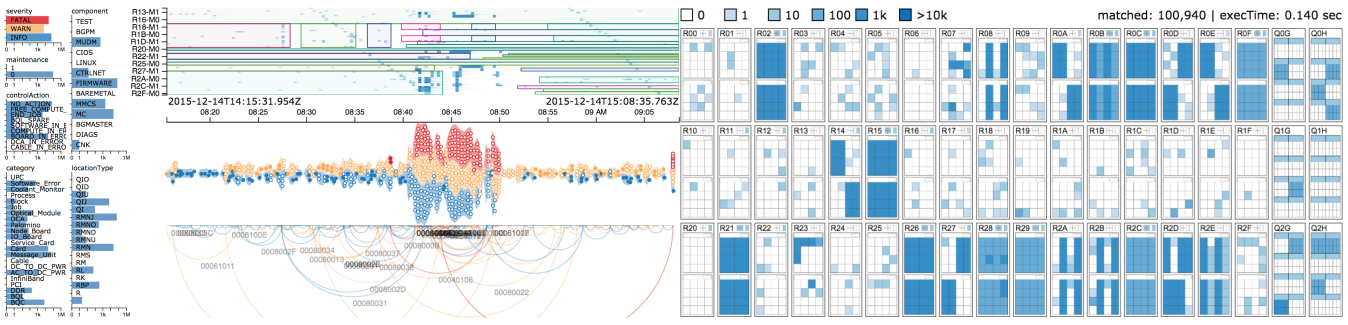

(a) (b) (c)

Figure 1: *Main user interface of* La VALSE*: (a) multidimensional view, (b) timeline view that features scalable ThemeRiver and arc diagram, (c) physical view.*

Abstract

We design and implement La VALSE*—a scalable visualization tool to explore tens of millions of records of reliability, availability, and serviceability (RAS) logs—for IBM Blue Gene/Q systems. Our tool is designed to meet various analysis requirements, including tracing causes of failure events and investigating correlations from the redundant and noisy RAS messages.* La VALSE *consists of multiple linked views to visualize RAS logs; each log message has a time stamp, physical location, network address, and multiple categorical dimensions such as severity and category. The timeline view features the scalable ThemeRiver and arc diagrams that enables interactive exploration of tens of millions of log messages. The spatial view visualizes the occurrences of RAS messages on hundreds of thousands of elements of Mira—compute cards, node boards, midplanes, and racks—with view-dependent level-of-detail rendering. The multidimensional view enables interactive filtering of different categorical dimensions of RAS messages. To achieve interactivity, we develop an efficient and scalable online data cube engine that can query 55 million RAS logs in less than one second. We present several case studies on Mira, a top supercomputer at Argonne National Laboratory. The case studies demonstrate that* La VALSE *can help users quickly identify the sources of failure events and analyze spatiotemporal correlations of RAS messages in different scales.*

1. Introduction

Resilience—the most difficult and underaddressed problem in today's high performance computing—becomes a significant issue as the systems approach exascale. For example, in a typical petascale supercomputer, a failure event that causes job termination could happen every several days. Such failures arise from hardware, system software, file systems, power, and even the cooling system. Large-scale scientific simulations and data analysis jobs are vulnerable to the errors, because fatal system errors may lead to the unexpected termination and job failures during execution. To this end, researchers have been investigating systems and software that are resilient to failures. Exploring the properties and correlations of the failure events is nontrivial; however, because of the large-scale, complicated system architecture that involves hundreds of

† {hguo,sdi1,rgupta,tpeterka,cappello}@anl.gov

DOI: 10.2312/pgv.20181099

thousands of various types of modules and components, such as processors, memory units, network modules, and power supplies.

One of the most important ways to study resilience is to perform posthoc analysis of the logs that record the error, warning, and informational messages generated by different components in supercomputers. These logs provide the key information that can be used to understand the features of failure events and eventually help resilience researchers improve checkpoint/restart mechanisms and system software designs.

In this paper, we present a scalable visualization framework—*La VALSE*—to help users explore and understand log data of supercomputers. We specifically study Mira, which is a 10-petaflops IBM Blue Gene/Q supercomputer at Argonne National Laboratory. The system consists of 786,432 processors, 768 TB of memory, and 24 PB of parallel file system. The interconnection of Mira is a 5D torus network. In our visualizations, we mainly incorporate the reliability, availability, and serviceability (RAS) logs. The RAS logs accumulated over five years have 55 million entries.

The objectives of *La VALSE* are derived from the needs of two target user groups, resilience researchers and system administrators: (1) interactively exploring tens of millions of logs and (2) correlating errors that occur in different categories, locations, and times. First, users need to trace the causes of failures and understand error propagation through interactive exploration. With *La VALSE*'s scalable visualizations, users can explore the causes of failures with their domain knowledge. For example, an unexpected memory error could lead to a bit flipping in the code segment of a user application. The bit flipping may cause an instruction error, resulting in failures in functions, threads, processes, and jobs. Second, it is important to correlate different types of errors in order to understand failures. In addition to the obvious correlations such as memory error and instruction error in the previous example, we must be able to comprehensively identify all correlations of different types of errors that happen in different spatiotemporal locations.

Major challenges in designing *La VALSE* include (1) visual representation of noisy and heterogeneous logs and (2) scalability of handling tens of millions of log records for interactive visualization and analysis.

We must design new visual representations for logs that are noisy, heterogeneous, and with hierarchical and high-dimensional network topologies. First, the RAS logs are so noisy that key messages can be easily obscured by using traditional visualizations. In general, most of the RAS records are warnings and informational messages, and only 1% of the RAS records ($\sim$100K messages per year) are fatal errors. In our observation, the majority of messages are insignificant, random, or duplicated during a burst. Second, there are heterogeneous sources of logs incorporating jobs and RAS. These logs have distinct data structures and reflect different aspects of the system. Third, the errors may be the reason for other components in a high-dimensional, hierarchical network, introducing a huge challenge when exploring their correlations. In Blue Gene/Q systems, for example, compute nodes are installed on node boards, which are located in different midplanes and racks; compute nodes are also connected in a 5D torus network.

We must also visualize and handle log data with scalability. Mira, which consists of $\sim$100K components, has generated 55 million RAS messages in the past five years. Existing visualizations are not able to render so many messages at an interactive speed. ThemeRiver [HHN00, HHWN02] and arc diagrams [Wat02], for example, are well-known tools to visualize repetition patterns over time, but such diagrams are not scalable to tens of millions of log records. *La VALSE* scales the interactive arc diagram rendering by binning the messages over time and avoiding overplotting the machine components with level-of-detail rendering. We must also scale data handling to support interactive queries. Although existing online analytical processing (OLAP) tools such as Nanocubes [LKS13] and imMens [LJH13] can handle geospatial and multidimensional queries, they are not directly applicable to RAS logs that contain complex network information. Instead, we need to redesign scalable OLAP query engines to handle RAS log data.

La VALSE features several novel designs in both visualization representation and data handling. We propose scalable ThemeRiver, scalable arc diagrams, and a scalable physical view to enable interactive exploration of tens of millions of log records through web browsers. The scalable ThemeRiver magnifies and highlights a small volume of important log messages; the scalable arc diagrams reduce geometry primitives by binning logs for fast and scalable rendering; and the scalable physical view uses semantic zooming and level-of-detail rendering to visualize hundreds of thousands of components on Mira, such as compute cards, link chips, optical modules, and power modules. The implementation of the web-based user interface is based on d3.js, SVG, and HTML5 Canvas. More details on the scalable visualization designs are in Section 5.

La VALSE also features scalable querying designs. We developed a customized in-memory database to enable execution of data cube queries for RAS logs with high dimensionality, large volume, and complex physical and topological locations. Not only can the query engine can run in a single machine, but it also is scalable to distributed and parallel environments. We also use a sparse data structure to reduce interprocess communication for scalable querying. More details on scalable querying are in Section 6.

To the best of our knowledge, *La VALSE* is the first visualization framework that addresses the challenge of analyzing RAS logs. Overall, the contributions of this paper are threefold:

- A scalable log visualization framework for fault characterization in supercomputers
- A redesign of scalable ThemeRiver and scalable arc diagrams for visualizing tens of millions of log records
- A scalable data cube query engine for interactive queries of tens of millions of log records

The remainder of this paper is organized as follows. Section 2 summarizes the related work. Section 3 gives an overview of the system, followed by the description of the data. We then introduce our scalable visualization and querying techniques in Section 5 and Section 6, respectively. Cases studies and discussions are in Section 7 and Section 8, respectively, followed by the conclusions in Section 9.

2. Related Work

We review related work in three aspects: failure log analysis, performance visualization, and spatiotemporal visualization.

2.1. Failure Log Analysis

Log analysis helps understanding failures in large computer systems such as supercomputers, clusters, and clouds. The resilience community has developed many automatic log analysis tools, yet little has been done to understand root causes and correlations of failures with visualization.

Automatic log analysis methods include pattern mining, signal analyses, and correlation analyses. Event log mining (HELO) [GCTK11], for example, is a message pattern mining tool for extracting templates that describe event formats based on the log files generated by large scale supercomputers. Event Log Signal Analyser (ELSA) [GCK12] merges signal analysis concepts with data mining techniques and offers an adaptive, efficient event prediction module. An adaptive semantic filter [LZXS07] efficiently filters the duplicated messages in a log by a semantic analysis of the large amount of log messages. Many other log analysis methods such as co-analysis of RAS logs [ZYT*11], holistic approaches [SB15], dependency-driven analysis [MCAC17], LogMaster for clusters [FRZ*12], and LogDiver [MJK*15] do not provide any visualization methods either. An exception is LogAider [DGS*17], which mines the potential event correlations in the system logs. It explores the cross-field correlation by posterior probability analysis, searches the spatial correlations by a torus topology analysis with a midplane granularity, and mines the temporal correlation. The analysis data can be plotted in LogAider by using Gnuplot scripts, but it does not provide interactive visualization for users. By comparison, not only does *La VALSE* provide a rich set of visualization interfaces across different logs to analyze the events interactively, but it also provides much finer spatial correlation (such as node board) for users.

Limited visual exploration tools have been developed for understanding failure logs. For example, IBM provides Blue Gene Navigator [LK13], which is designed for system administrators to monitor the system through basic visualizations of log statistics. RAVEN [PHGG10] is another attempt to visually map various types of RAS logs on a physical system map for Titan, a Cray machine at Oak Ridge National Laboratory. However, none of the existing tools supports scalable and interactive visual analysis of RAS logs.

2.2. Performance Visualization

Our study is related to but different from performance visualization, which aims to help high performance computing (HPC) developers profile their software. A comprehensive review of performance visualization can be found in [IGJ*14].

The performance visualization community has developed methods to visualize communication traffic in different network topologies, which is critical for HPC developers and administrators for studying communication patterns [IBJ*14, WZYK14], investigating network latencies [IGB*16], and scaling parallel applications. For example, Isaacs et al. developed Boxfish [ILG*12],which can visualize the performance data in 2D and 3D torus networks. McCarthy et al. [MIB*14] further generalized Boxfish to 5D torus networks. Ring layout [BGI*12] provides another way to project high dimensional networks, and Cheng et al. [CDJM14] further combine parallel coordinates and ring layout for performance visualization. In our study, we also use parallel coordinates to help users filter compute nodes in torus networks, as explained in the following sections.

2.3. Spatiotemporal Data Visualization

Our study is related to spatiotemporal data visualization, which is a well-studied topic in geographic information systems and visual analytic systems. Typical examples include visualization of trajectory and movement data [AA13, GWY*11], and social media visualization [CYW*16]. A comprehensive review of spatiotemporal visualization can be found in [AAG03].

La VALSE uses multidimensional and spatiotemporal aggregation, or OLAP data cubes, to help users understand correlations between attributes of failure events. Traditionally, data cubes are implemented in and supported by relational databases, which precompute every possible combination of aggregation and require many storage and computing resources. Recently, the visualization community has developed various light-weight data cube query engines for interactive visualization, such as Nanocubes [LKS13] and imMens [LJH13].Hierarchical data structures and screen space approximations are used to reduce the cost and to boost the performance. In order to further scale the aggregation on large datasets, distributed and parallel query engines are developed to support interactive queries [BIM*].

In this work, we redesign a scalable data cube query engine for *La VALSE* for three reasons. First, compared with geospatial visualizations, the spatial dimensions, including hierarchical physical layout of the machine and torus network addresses, are much more complicated. The spatial dimension must be customized to meet our analysis requirements. Second, the dimensionality in our data (tens of variables) is much higher than the existing records (a handful of variables) in previous literature. Our engine can handle all dimensions in RAS logs. Third, we need a distributed and parallel solution to handle the large data and complex queries for interactive exploration. More details on the scalable data cube query engine design are in Section 6.

3. System Overview

The main user interface and the system design of *La VALSE* are illustrated in Figure 1 and Figure 2, respectively. The main user interface of *La VALSE* comprises multiple linked views including the timeline view, physical view, multidimensional view, and correlation view. Through the user interface, users can investigate the log data from different perspectives, different spatiotemporal regions, and different query criteria.

The timeline view, which features the scalable ThemeRiver and arc diagram designs, visualizes the distributions of RAS logs and job logs over time. All the subviews align with the time axis in the

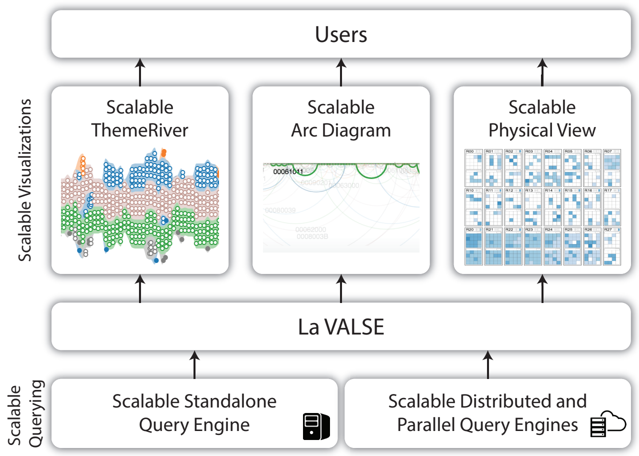


Figure 2: *Overview of* La VALSE.

horizontal direction. The job visualization on the top of the timeline has two layers, and each layer aligns with the y-axis that encodes the midplanes (from R00-M0 to R2F-M1). The bottom layer is a heatmap that shows the distributions of RAS messages over different midplanes; the top layer visualizes machine allocations of jobs as semitransparent rectangles. The middle of the timeline view is the scalable ThemeRiver with sampled messages embedded in the rivers. The bottom of the timeline view is the scalable arc diagram that helps users understand time correlations of different RAS messages. Users can browse any time period by zooming in and brushing. More details on the scalable ThemeRiver and arc diagram designs are in Sections 5.2 and 5.1, respectively.

The physical view visualizes the spatial distribution of RAS messages in the time period of interest. The physical view provides a scalable level-of-detail rendering and semantic zooming mechanism to enable interactive exploration of RAS messages on more than 100K components in Mira. The physical view also allows users to quickly explore the interconnections between compute nodes. More details on the physical view are in Section 5.3.

The multidimensional views visualize the statistics of RAS messages with bar charts. Bar charts show how many RAS messages exist for each value in the current data cube query. For example, we show the number of INFO, WARN, and FATAL messages in the severity bar chart, while we simultaneously show the number of messages under different RAS log categories.

4. Data Description and Preprocessing

We briefly review the system topology of Blue Gene/Q systems, describe the data structure of log data, and introduce the data preprocessing for further visualization and analysis.

4.1. Mira System Topology

As shown in Figure 1(c), Mira consists of 48 compute racks (R) and 6 I/O racks (Q). The racks are organized in three rows and 18

Table 1: *Attributes in RAS logs*

Attribute	Type	#Values	Example
Record ID	int	N/A	36212598
Message ID	int	86	0x0008000B
Time	time	N/A	2015-02-15T22:23:08.432574Z
Severity	string	3	FATAL
Component	string	22	FIRMWARE
Category	string	18	BQC (Blue Gene/Q Compute Card)
Description	string	86	A hardware eror has been detected by the Level 1 Prefetch unit of a BQC.
Location	string	N/A	R0B-M1-N15-J28
Job ID	int	N/A	4713291
Job Partition†	string	N/A	MIR-40000-73FF1-8192
Message	string	N/A	L1P Machine Check:...

columns; the first 16 columns (0 to F) are compute racks, and the last two columns (G and H) are I/O racks.

Compute racks Figure 3 illustrates the hierarchical structure of a compute rack, which consists of two midplanes (M0 and M1). Each midplane has 16 node boards (N00 to N15) and one service card (S). A node board further contains 32 nodes (so-called compute cards). Each node connects to its neighbor nodes in the 5D torus network. The locations of different elements are encoded in a hierarchical manner. For example, R1A-M0-N03-J05 is the sixth compute card on the fourth node board in the first midplane of rack R1A.

I/O racks As shown in Figure 1(c), I/O racks are also organized hierarchically. Unlike compute racks, each I/O rack contains 8 I/O drawers (I). Each I/O drawer has 8 compute cards, which are also encoded in a hierarchical manner.

Torus network The interconnect of the nodes (compute cards in compute racks) on Mira is a 5D torus. The dimensionality of the torus is $8 \times 12 \times 16 \times 16 \times 2$, and each node can be mapped uniquely to a torus coordinate.

4.2. RAS Logs

The RAS infrastructure on Blue Gene/Q systems provides the means to record both hardware and software events occurring in the system. Since the start of production use of Mira, there are 55 million messages totaling 21 GB.

Table 1 illustrates that each RAS message has a unique record ID, message ID, time, location, job ID, job partition, and textual message. Among the variables, message ID is the most important; it determines the severity, component, category, and control actions of the message. Blue Gene/Q systems have 822 distinct types (86 types in our data) of message IDs, which can be stored as a dictionary called the RAS book. The following describes other important variables that we consider.

Severity is informational (INFO), warning (WARN), or fatal (FATAL). The percentage of INFO, WARN, and FATAL is 29.7%, 69.4%, and 1%, respectively. In previous studies on RAS logs [ZYT*11, DGS*17], only FATAL messages were processed. We instead provide a comprehensive solution to analyze and visualize messages at all severity levels.

Category has of 22 different values. The most significant values

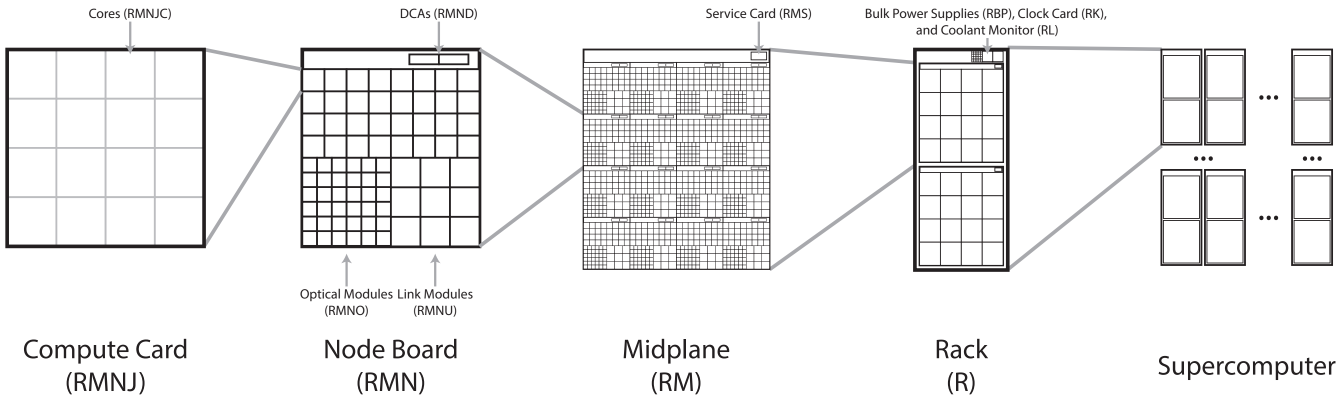


Figure 3: *Hierarchy of IBM Blue Gene/Q systems.*

include software errors, double data rate (DDR) memory, compute cards (BQC), and link modules (BQL).

Component records the RAS software component that detects and reports the messages. The values include service related facilities (BAREMETAL), control system (MMCS), processes that monitor the control system (BGMASTER), common I/O service (CIOS), compute node kernel (CNK), diagnosis functions (DIAGS), machine controller (MC), firmware (FIRMWARE), control net (CTRLNET), memory unit (MDUM), and Linux (LINUX).

Location records the location code of the RAS message. The location code is in the hierarchical format (e.g. R0B-M1-N15-J28) except that in rare cases (0.05%) the location is undefined. We further derive two additional variables based on the location code, including the location type and torus coordinates. Details on the derived attributes are in the following sections.

4.3. Definition of Levels of Detail

We define levels of detail (LOD) for elements in Mira for both visualization and analysis purposes. For the visualization of message distributions on the machine, we must adapt the levels of detail to the available pixels on the screen, and we must also reduce the burden of human perception by aggregating the results into fewer elements in the visualization results. For the analysis of message propagation, we also need to study the patterns in different levels of detail, such as the midplane level and the node board level.

We have defined four levels of details (L0 to L3). L0 is the finest level and contains more than 120K elements including compute cards, link modules, optical modules, and direct current assemblies. L1 and L2 are the node board level and midplane level, respectively. L3 is the rack level, which contains only compute racks and I/O racks. Notice that the lower level always contains the elements in the higher levels. For example, the L3 element R1C is also a member of the L0 elements.

5. Scalable RAS Log Visualization

The key to achieving our analysis goals is the scalable visualization of log data. We have identified at least three major visual components that are not scalable with respect to the number of RAS messages. We describe their redesign in the following subsections.

5.1. Scalable ThemeRiver

We redesign ThemeRiver [HHN00, HHWN02]—a tool that visualizes trends of different input volumes—to scale to the dynamic range of the input volumes and to highlight individual messages. We define the dynamic range as the maximum value over the minimum value for a given input volume. For example, if the number of INFO, WARN, and FATAL messages is 10,000, 5000, and 1, respectively, the dynamic range is 10,000.

Our improvement is based on two observations of visualizing RAS logs with ThemeRiver. First, the dynamic range is limited in ThemeRiver. In the above example, if we faithfully create a ThemeRiver with a linear scale, the single important FATAL message can hardly be seen, as shown in Figure 4(a). Second, individual messages are not visible in traditional ThemeRiver. Both resilience researchers and system administrators need to know the specific RAS messages that lead to the failure. Thus, we need to redesign ThemeRiver, as described below.

First, we redesign the mapping from the number of occurrences to the river width, in order to magnify the messages with fewer occurrences. In RAS log analysis, because the machine may generate a burst of duplicated and similar messages, the number of messages is less important than the occurrence of individual messages. That is, for each time bin in ThemeRiver, users care more about whether a message occurs under a certain category than how many messages occurred under this category. We thus use a logarithmic scale to map the number of occurrences to river width W:

$$W(m) = \begin{cases} 0 & m = 0 \\ 1 + \log_{10} m & m \geq 1 \end{cases}, \qquad (1)$$

where m is the number of messages. Based on this mapping, the scalability with respect to the dynamic range is improved. As shown in Figure 4(b), the FATAL message appears more prominently than in the linear mapping in Figure 4(a).

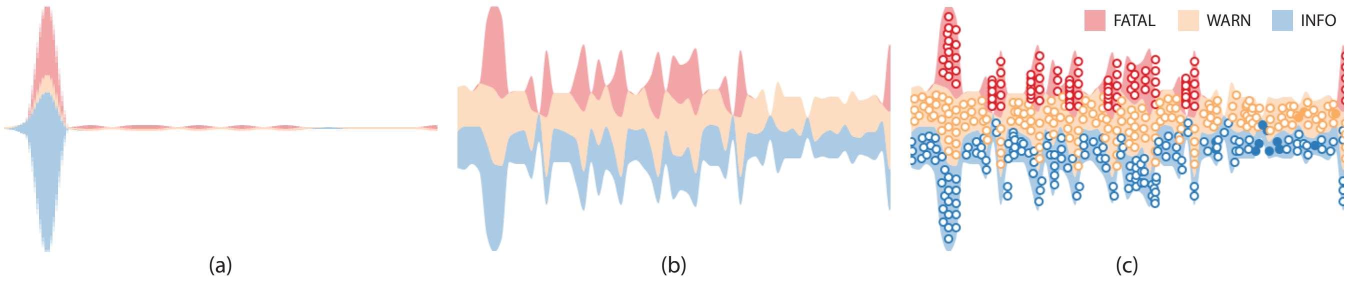


Figure 4: *ThemeRiver designs in* La VALSE*: (a) the original ThemeRiver, (b) scalable ThemeRiver, and (c) scalable ThemeRiver with sampled RAS messages embedded.*

Second, we embed sampled messages in the river. In the investigation of root causes of failures, as shown in Figure 4(a) and 4(b), users cannot directly access the individual RAS messages with ThemeRiver, because only message statistics are visualized. In our design, for each time bin, we randomly sample the messages from each category in the ThemeRiver and then plot the messages as glyphs overlaid on the rivers. The number of sampled messages is $\min\{m, \alpha W(m)\}$, where α is a constant related to the height of the ThemeRiver chart. If $m \le \alpha W(m)$, all messages in the specific time bin and category will be shown as solid circles; otherwise only $\alpha W(m)$ messages are sampled and visualized as hollow circles. Users can further investigate the message by moving the mouse over the glyph. Compared with Figure 4(b), the embedded glyphs can show detailed messages, given available spaces in the rivers.

We regard our design as a scalable visualization, not only because it scales well with large numbers of messages, but also because it can emphasize details when the number of messages is small. Based on the overview provided by the scalable ThemeRiver, users can further zoom in on the timeline to explore the messages until the most important messages are identified.

Although the logarithmic scale is nonlinear and does heavily distort the number of messages, the distortion is acceptable and suitable for our application. Based on discussions with the users, the number of messages is usually not critical to the analysis, because the logs usually have many duplicated noisy messages. For example, a burst of failure messages could be generated due to a single root cause. On the one hand, the users care mainly about the first failure message that led to the problem, and care less about how many follow-up messages are there. On the other hand, important messages with few occurrences are magnified and emphasized by the distortion.

5.2. Scalable Arc Diagram

We propose a scalable solution to render arc diagrams to help users understand time correlations between different RAS messages. The arc diagram, which was proposed by Wattenberg [Wat02], is used to visualize repetition patterns in time sequences.

In *La VALSE*, the rationale of using an arc diagram is to make comparisons of repeated patterns between different types of RAS messages, which are encoded by message IDs. If two message IDs have similar occurrence patterns, their arc shapes are also similar in the arc diagram, and thus the two message IDs are highly correlated in time.

Direct drawing of arc diagrams, however, does not scale with tens of millions of messages. In the direct drawing algorithm, an arc connects two adjacent messages with the same message ID on the timeline. Both the storage and time complexity of the direct drawing algorithm is O(the number of messages), but keeping and rendering tens of millions of message in the client side are impossible.

We instead design a scalable way to render arc diagrams. First, we filter all messages that happen during the time period of interest and then sort the messages into n bins that uniformly partition the time, where n is the number of pixels along the timeline. For each message ID, we create a vector with n elements. Each element is 1 if at least one message with the message ID falls into the corresponding time bin; otherwise it is 0. Second, we draw a group of arcs for each message ID based on the vector. An arc connects the ith and jth bins if the value on i and j are 1 and all elements between the ith and jth bins are 0. With our scalable solution, the complexity of rendering is O(number of pixels n) instead of O(number of messages), where n $<<$ m.

Our scalable arc diagram view supports various user interactions for data exploration, including mouseover highlighting, brushing, zooming, and panning. Mouseover highlighting enables users to probe a message ID by moving the mouse pointer to the corresponding arc without clicking, which reduces the extra action for efficient exploration. The highlighted message ID will also be reflected in other views. Brushing allows users to select one or multiple message IDs for interactive queries. Users can also zoom in on and pan over the arc diagram to change the time period of interest.

5.3. Scalable Physical View

We design a scalable physical view that is capable of visualizing frequencies of RAS messages across more than 100K different components on Mira, with LOD rendering, semantic zooming and panning, and network topology probing. Figure 1(c) demonstrates the user interface of the physical view, whose layout follows the physical location of the components.

The spatial view supports semantic zooming and panning. As shown in Figure 5, the LOD changes with the viewport during the

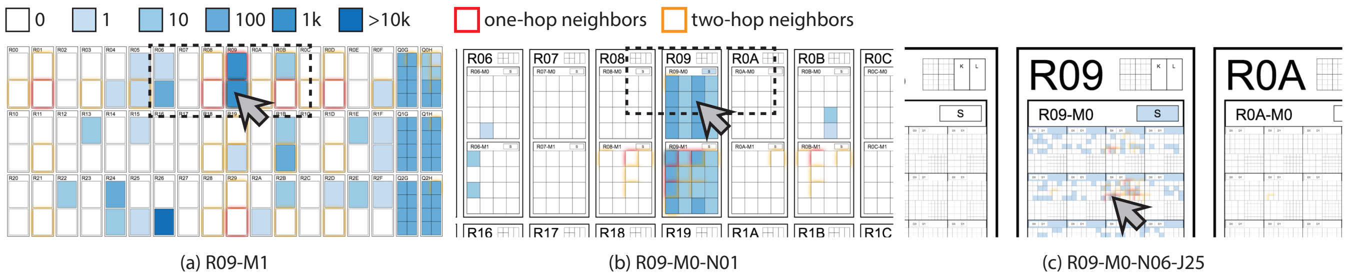


Figure 5: *View-dependent and level-of-detail rendering in the spatial view: (a) midplane level, (b) node board level, and (c) compute card level. The dashed box in (a) and (b) represents the region shown in (b) and (c), respectively. The color encodes the number of RAS messages on each element. The colored shadows indicate the neighbors of the current element under the mouse cursor.*

user interaction. We map the current zoom factor into the corresponding level of detail; if the LOD changes, we update the rendering results by requesting a new query. The semantic zoom also keeps the context when zoomed in. For example, when the user zooms into the compute card level, the upper-level racks and midplanes remain visible.

The spatial view also allows users to interactively probe neighbors in the torus network in different levels of detail, in order to help users investigate spatial correlations and how RAS messages propagate over the network. For example, in Figure 5(a), in the midplane level, when the mouse pointer moves over the midplane R09-M1, both one-hop and two-hop neighbors, including both midplanes on computing racks and I/O boards on I/O racks, are highlighted with halos. Likewise in Figure 5(b) and 5(c), neighbors in the node board and compute card levels are highlighted.

The rationale for the semantic zooming is threefold. First, it enables users to “overview first, zoom and filter, then details-on-demand" [Shn96] in different LODs. For example, users can first explore how RAS messages are distributed in different midplanes and then drill down to the node board level or compute card level. Users can investigate how messages propagate between midplanes, node boards, and compute cards. Second, it reduces the burden of perception. Users have difficulty seeing the patterns, given hundreds of thousands of elements. Different levels of abstraction and aggregation are needed for users to explore the data. Third, the semantic zooming avoids overflowing the rendering pipeline, because only a small number of elements are rendered at once.

Because the LOD rendering of more than 100K geometries is not currently supported by SVG, we use HTML5 canvas to customize the rendering pipeline in the machine view. A hierarchical data structure is built to manage the rectangles that represent different elements in the machine. Upon zooming in and panning, a culling test is performed to filter all elements that intersect the viewport; only the elements that pass the culling test are rendered.

6. Scalable RAS Log Queriying

In addition to scalable visualizations, we must also scale the handling and querying of tens of millions of RAS messages. We have developed both standalone and parallel versions of the query engines for users with different amounts of resources, as described below.

The key to achieving interactive exploration of tens of millions of RAS logs is a customized in-memory database implemented with C++. The in-memory database consists of an offline data converter and online query engine.

The offline data converter encodes all chronological and categorical variables in the RAS data into a compact and binary format, which is loaded by the query engine for fast online queries. The data conversion of categorical variables is based on the enumeration of all possible values during preprocessing. For example, locations require 10 bits, because there are at most 127,357 possible locations in the data. We use 16-bit integers for ease of access. Likewise, users, projects, and queues are also encoded with 16-bit integers. Variables such as severity, component, and category are not stored because they can be directly derived from the message ID on the fly. The storage size of the converted binary data is only 480 MB per year, which is memory efficient for even commodity laptops.

The online engine supports fast in-memory queries, even though it checks every single entry to determine whether the entry matches the query. The engine is also in charge of sampling the RAS data and returning the record IDs of the results for visualizing individual logs in ThemeRiver, in order to further accelerate the process with multithreading. First, we subdivide the RAS data into equal number of entries for each thread. Second, we compute the query results with each thread in parallel. Third, we merge the query results from different threads into the final results.

The performance of the query engine can achieve interactive speed for various queries. On a 2014 MacBook Pro laptop with a 2.5 GHz Intel Core i7 CPU and 16 GB of RAM, the engine can always return results in less than one second. Compared with MongoDB with optimal settings, our engine typically runs 10 to 100 times faster. Because the performance varies for different queries, a comprehensive performance study is not available. We instead conducted benchmarks for several arbitrary queries to verify the conclusion. For example, for the simple query {location=R0B-M1-N06-J00, component=FIRMWARE}, MongoDB returns the results in 31.210 seconds, while our search engine returns the full results in 0.78 seconds.

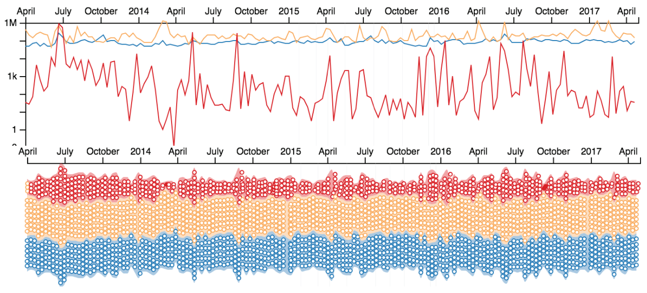


Figure 6: *Demonstration of daily message count based on 5-year MIRA RAS log.*

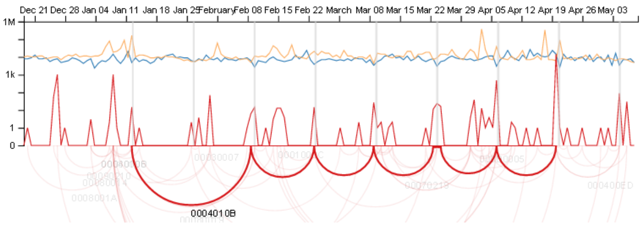


Figure 7: *Demonstration of daily message count based on half-year MIRA RAS log and periodic occurrence of particular fatal messages.*

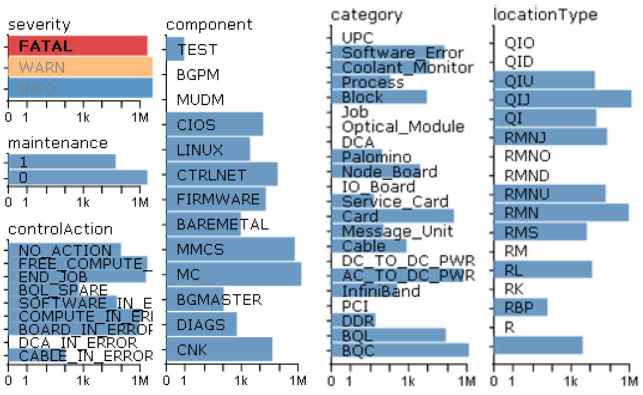


Figure 8: *Demonstration of components, categories, and location types with FATAL messages.*

To further improve the user experience for interactive queries, we implemented a distributed and parallel query engine to boost the performance. We used the MPI programming model to implement the query engine: RAS data are evenly partitioned into different processes, and then each process queries the corresponding data independently. After the independent queries, query results are summed up, gathered, and stored in the root process with parallel reduction operations.

7. Example

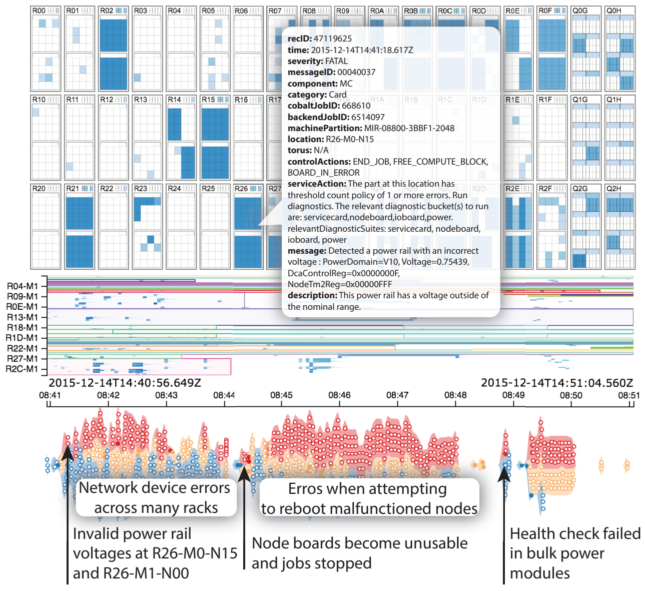
We demonstrate three failure examples that are identified and characterized with *La VALSE*.

Overview of five-year data Figure 6 presents an overview of the daily message count based on MIRA RAS log through the five-year period. One can clearly observe a typical zigzag pattern about the daily number of FATAL messages in the five years, and the daily FATAL message count spans a large range—from 0 to 1 million. By contrast, the daily amounts of INFO and WARN messages do not exhibit such a clear zigzag pattern because a large number of such messages (tens of thousands of messages or even more) are always generated every day. One can also learn from the figure that the daily count of FATAL messages generally is not correlated with the daily count of other types of severity messages (i.e., INFO messages and WARN messages). This observation implies that the FATAL events may not be closely related to other severity messages.

Exploring periodical patterns As shown in Figure 7, fatal messages with the event ID 0004010B may occur every 14 days for a certain period starting from February 8, 2015. One can also observe that this event ID likely appears in the period with peak fatal message count. All these observations help understand the occurrence of fatal events and the periodicity.

Identifying significant attributes involved with fatal severity *LA VALSE* can also present the distribution of fatal messages across different values of fields such as components, categories, location types, based on the user's interactive choices customized online. Specifically, when a user selects one or more particular severity levels, the visualization page will immediately show the distribution of the corresponding messages on different fields. In Figure 8, for example, the user clicked on the FATAL severity with the entire 5-year period, so the component panel presents the volumes of the FATAL messages on different values overall. Based on this visualization, one can clearly see that the FATAL messages may never happen to some components such as Blue Gene Performance Monitor (BGPM) and Memory Unit (MUDM). We can also observe that MC and MMCS are the most error-prone components and the FATAL messages fall into many different categories. All such observations can help system administrators narrow the range of the diagnosis of errors.

Investigating the root cause of a burst of network errors One of the failure events that we explored with *La VALSE* is a burst of network errors that happened on December 14, 2015. As shown in Figure 9, a user uses the multiple linked views to iteratively zoom on the details of a possible failure event and find the root cause. From the overview depicted by the ThemeRiver view, an outbreak of FATAL messages can be easily identified in December 2015. When we zoom into details at the minute level, from the multidimensional view we can find out that most of the FATAL messages belong to network device issues, such as receiver/sender link errors. From the physical view, we can see that the errors are distributed in a few racks that are interconnected. The timeline view also demonstrates the how the errors are distributed over racks and time. However, we cannot conclude that the root cause is the network failure. After further zooming in on the timeline, we note several FATAL errors at the beginning of this event, indicating that the node board R26-M0-N15 has one or more invalid power rail voltages. Since no other RAS messages could lead to this voltage error,


R04-M1
R09-M1
R0E-M1
R13-M1
R18-M1
R1D-M1
R22-M1
R27-M1
R2C-M1
2015-12-14T14:40:56.649Z
2015-12-14T14:51:04.560Z
08:41
08:42
08:43
08:44
08:45
08:46
08:47
08:48
08:49
08:50
08:51
Network device errors across many racks
Erros when attempting to reboot malfunctioned nodes
Invalid power rail voltages at R26-M0-N15 and R26-M1-N00
Node boards become unusable and jobs stopped
Health check failed in bulk power modules

recID: 47119625
time: 2015-12-14T14:41:18.617Z
severity: FATAL
messageID: 00040037
component: MC
category: Card
cobaltJobID: 668610
backendJobID: 6514097
machinePartition: MIR-08800-3BBF1-2048
location: R26-M0-N15
torus: N/A
controlActions: END_JOB, FREE_COMPUTE_BLOCK, BOARD_IN_ERROR
serviceAction: The part at this location has threshold count policy of 1 or more errors. Run diagnostics. The relevant diagnostic bucket(s) to run are: servicecard,nodeboard,ioboard,power. relevantDiagnosticSuites: servicecard, nodeboard, ioboard, power
message: Detected a power rail with an incorrect voltage : PowerDomain=V10, Voltage=0.75439, DcaControlReg=0x0000000F, NodeTm2Reg=0x00000FFF
description: This power rail has a voltage outside of the nominal range.

Figure 9: *Burst of network errors caused by a power failure on December 14, 2015.*

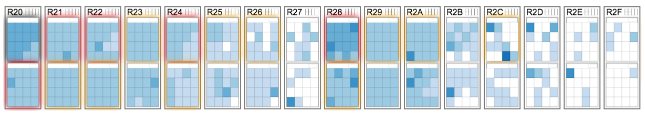


Figure 10: *Example of message propagation over torus network. Red halo: one-hop neighbors of R20-M0, orange halo: two-hop neighbors of R20-M0.*

one can conclude that the root cause for the burst of network errors is the voltage error on R26-M0-N15.

Discovering spatial correlations As shown in Figure 10, when the user hovers on one midplane, the physical view panel will immediately show various colors on the edges of the close midplanes in the torus network. Red-edged midplanes indicate that they have only one hop difference from the selected midplane, and orange-edged midplanes mean two hops in the network. Through the visualizations, we can clearly see the correlations between message occurrences and the number of hops from the selected midplane. We can also see that the error seems to propagate from the selected midplane through the torus network, such that the system administrators can infer that the source of the errors is likely related to the network component.

8. Discussion

Although the current implementation of *La VALSE* is tailored for Mira, the scalable design of *La VALSE* has generality and thus can be extended to analyze logs in other systems, with modifications depending on the architecture.

For IBM Blue Gene systems, the extension is straightforward and requires the least effort. First, the system hierarchy (e.g., rack, midplane, node board, and compute card hierarchy) of different IBM Blue Gene system is similar or even identical. Users can simply write down the specifications of a Blue Gene systems in the configuration file, such as the number of rows and columns of both compute and I/O racks in the system. Second, RAS logs of different Blue Gene systems follow almost the same protocol. As long as the system specification is given, users can directly import the RAS log data into *La VALSE* query engine without any additional effort.

For other systems, additional efforts are necessary in order to define the system hierarchy and the properties of logs. The mapping between individual components and geometries in the physical view must be defined in a CSV file, as exemplified in Table 2. The logs also need to be categorized by severity, location type, and other properties for the interactive exploration with data cubes. Such information is directly available in mainstream supercomputing systems. For example, Cray systems have similar types of RAS logs to that of Blue Gene systems. One can redefine the data dimensions in the query engine, and then import the logs for interactive queries.

Table 2: *An example CSV file that defines the physical layout of a system. For Blue Gene systems, the CSV file can be automatically generated from a template by specifying the number of columns and rows. For other systems, users need to configure the LOD and geometry of each individual component of the target system, in order to explore the distribution of logs with* La VALSE.

ID	LOD	x	y	width	height	text
...	...	...	...	...	...	...
R00-M0	2	4	12	30	41	R00-M0
R00-M0-S	1	25	13	8	3	S
R00-M0-N00	1	4	17	7.5	9	N00

9. Conclusions and Future Work

The objective of *La VALSE* is to provide a scalable user interface for exploring large amounts of log data in supercomputers. The scalability is reflected in two aspects: visualization and querying. In the visualization aspect, we designed scalable versions of ThemeRiver, the arc diagram, and the physical view. In the querying aspect, we developed scalable query engines to support interactive exploration. We demonstrated interactive exploration results of 55 million RAS logs of the IBM Blue Gene/Q supercomputer Mira. Resilience researchers and system administrators can benefit from *La VALSE* to locate root causes of failures and discover correlations between different log messages. We provide several use cases that are conducted by users.

In the future, we would like to generalize *La VALSE* to visualize logs for other supercomputers and clusters, such as Cray systems and Linux clusters. In addition to RAS and job logs, we plan to add I/O logs, communication logs, and performance logs to support comprehensive analysis over the entire machine. We also plan to incorporate new criteria to show representative messages in the ThemeRiver. We would also like to incorporate automatic log analyses into the workflow of *La VALSE*.

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357. This work is also supported by the U.S. Department of Energy, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program.

References

[AA13] Andrienko N. V., Andrienko G. L.: Visual analytics of movement: An overview of methods, tools and procedures. *Information Visualization 12*, 1 (2013), 3–24. 3

[AAG03] Andrienko N. V., Andrienko G. L., Gatalsky P.: Exploratory spatio-temporal visualization: an analytical review. *J. Vis. Lang. Comput. 14*, 6 (2003), 503–541. 3

[BGI*12] Bhatele A., Gamblin T., Isaacs K. E., Gunney B. T. N., Schulz M., Bremer P., Hamann B.: Novel views of performance data to analyze large-scale adaptive applications. In *SC'12: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (2012), p. 31. 3

[BIM*] Budiu M., Isaacs R., Murray D., Plotkin G., Barham P., Al-Kiswany S., Boshmaf Y., Luo Q., Andoni A.: Interacting with large distributed datasets using sketch. In *EGPGV'16: Proceedings of Eurographics Symposium on Parallel Graphics and Visualization*, Bethel W., Gobbetti E., (Eds.), The Eurographics Association, pp. 31–43. 3

[CDJM14] Cheng S., De P., Jiang S. H., Mueller K.: TorusVis$^{\text{ND}}$: unraveling high-dimensional torus networks for network traffic visualizations. In *VPA'14: Proceedings of Workshop on Visual Performance Analysis* (2014), pp. 9–16. 3

[CYW*16] Chen S., Yuan X., Wang Z., Guo C., Liang J., Wang Z., Zhang X. L., Zhang J.: Interactive visual discovering of movement patterns from sparsely sampled geo-tagged social media data. *IEEE Trans. Vis. Comput. Graph. 22*, 1 (2016), 270–279. 3

[DGS*17] Di S., Gupta R., Snir M., Pershey E., Cappello F.: LogAider: A tool for mining potential correlations in HPC log events. In *CCGrid 2017: Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (2017), pp. 442–451. 3, 4

[FRZ*12] Fu X., Ren R., Zhan J., Zhou W., Jia Z., Lu G.: Log-Master: Mining event correlations in logs of large-scale cluster systems. In *SDRS'12: Proceedings of IEEE Symposium on Reliable Distributed Systems* (2012), pp. 71–80. 3

[GCK12] Gainaru A., Cappello F., Kramer W.: Taming of the shrew: Modeling the normal and faulty behaviour of large-scale HPC systems. In *IPDPS'12: Proceedings of IEEE International Parallel and Distributed Processing Symposium* (2012), pp. 1168–1179. 3

[GCTK11] Gainaru A., Cappello F., Trausan-Matu S., Kramer B.: Event log mining tool for large scale HPC systems. In *EuroPar'11: Proceedings of International Conference on Parallel and Distributed Computing* (2011), pp. 52–64. 3

[GWY*11] Guo H., Wang Z., Yu B., Zhao H., Yuan X.: TripVista: Triple perspective visual trajectory analytics and its application on microscopic traffic data at a road intersection. In *Proceedings of IEEE Pacific Visualization Symposium 2011* (2011), pp. 163–170. 3

[HHN00] Havre S., Hetzler E. G., Nowell L. T.: ThemeRiver: Visualizing theme changes over time. In *Proceedings of IEEE Symposium on Information Visualization 2000* (2000), pp. 115–123. 2, 5

[HHWN02] Havre S., Hetzler E. G., Whitney P., Nowell L. T.: ThemeRiver: Visualizing thematic changes in large document collections. *IEEE Trans. Vis. Comput. Graph. 8*, 1 (2002), 9–20. 2, 5

[IBJ*14] Isaacs K. E., Bremer P., Jusufi I., Gamblin T., Bhatele A., Schulz M., Hamann B.: Combing the communication hairball: Visualizing parallel execution traces using logical time. *IEEE Trans. Vis. Comput. Graph. 20*, 12 (2014), 2349–2358. 3

[IGB*16] Isaacs K. E., Gamblin T., Bhatele A., Schulz M., Hamann B., Bremer P.: Ordering traces logically to identify lateness in message passing programs. *IEEE Trans. Parallel Distrib. Syst. 27*, 3 (2016), 829–840. 3

[IGJ*14] Isaacs K. E., Giménez A., Jusufi I., Gamblin T., Bhatele A., Schulz M., Hamann B., Bremer P.-T.: State of the art of performance visualization. In *EuroVis - STARs* (2014), Borgo R., Maciejewski R., Viola I., (Eds.), The Eurographics Association. 3

[ILG*12] Isaacs K. E., Landge A. G., Gamblin T., Bremer P., Pascucci V., Hamann B.: Exploring performance data with Boxfish. In *VPA'12: Workshop on Visual Performance Analysis* (2012), pp. 1380–1381. 3

[LJH13] Liu Z., Jiang B., Heer J.: imMens: Real-time visual querying of big data. *Comput. Graph. Forum 32*, 3 (2013), 421–430. 2, 3

[LK13] Lakner G., Knudson B.: *IBM System Blue Gene Solution: Blue Gene/Q System Administration*. International Business Machines Corporation, 2013. 3

[LKS13] Lins L. D., Klosowski J. T., Scheidegger C. E.: Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Trans. Vis. Comput. Graph. 19*, 12 (2013), 2456–2465. 2, 3

[LZXS07] Liang Y., Zhang Y., Xiong H., Sahoo R. K.: An adaptive semantic filter for Blue Gene/L failure log analysis. In *IPDPS'07: Proceedings of International Parallel and Distributed Processing Symposium* (2007), pp. 1–8. 3

[MCAC17] Mavlyutov R., Curino C., Asipov B., Cudré-Mauroux P.: Dependency-driven analytics: A compass for uncharted data oceans. In *CIDR'17, Proceedings of Biennial Conference on Innovative Data Systems Research* (2017). 3

[MIB*14] McCarthy C. M., Isaacs K. E., Bhatele A., Bremer P., Hamann B.: Visualizing the five-dimensional torus network of the IBM Blue Gene/Q. In *VPA'14: Proceedings of the Workshop on Visual Performance Analysis* (2014), pp. 24–27. 3

[MJK*15] Martino C. D., Jha S., Kramer W., Kalbarczyk Z. T., Iyer R. K.: LogDiver: A tool for measuring resilience of extreme-scale systems and applications. In *FTXS'15: Proceedings of the 5th Workshop on Fault Tolerance for HPC at eXtreme Scale* (2015), pp. 11–18. 3

[PHGG10] Park B. H., Heo J., Guruprasad K., Geist A.: RAVEN: RAS data analysis through visually enhanced navigation. In *CUG'10: Proceedings of Cray User Group Conference* (2010). 3

[SB15] Sirbu A., Babaoglu O.: A holistic approach to log data analysis in high-performance computing systems: The case of IBM Blue Gene/Q. In *EuroPar'15: Proceedings of International Conference on Parallel and Distributed Computing* (2015), Springer, Springer. 3

[Shn96] Shneiderman B.: The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of 1996 IEEE Symposium on Visual Languages, Boulder, Colorado, USA, September 3-6, 1996* (1996), pp. 336–343. 7

[Wat02] Wattenberg M.: Arc diagrams: Visualizing structure in strings. In *Proceedings of IEEE Symposium on Information Visualization 2002* (2002), pp. 110–116. 2, 6

[WZYK14] Wu J., Zeng J., Yu H., Kenny J. P.: CommGram: a new visual analytics tool for large communication trace data. In *VPA'14: Proceedings of Workshop on Visual Performance Analysis* (2014), pp. 28–35. 3

[ZYT*11] Zheng Z., Yu L., Tang W., Lan Z., Gupta R., Desai N., Coghlan S., Buettner D.: Co-analysis of RAS log and job log on Blue Gene/P. In *Proceedings of IEEE International Symposium on Parallel and Distributed Processing* (2011), pp. 840–851. 3, 4