ViSRE: A Unified Visual Analysis Dashboard for Proactive Cloud Outage Management

Paula Kayongo
Northwestern University
Illinois, USA
paulakayongo2023@u.northwestern.edu

Jane Hoffswell
Adobe Research
Washington, USA
jhoffs@adobe.com

Shiv Saini
Adobe Research
Bangalore, India
shsaini@adobe.com

Shaddy Garg
Adobe Research
Bangalore, India
shadgarg@adobe.com

Eunyee Koh
Adobe Research
California, USA
eunyee@adobe.com

Haoliang Wang
Adobe Research
California, USA
hawang@adobe.com

Tom Jacobs
Adobe Research
California, USA
jacobs@adobe.com

***Abstract*—Efficient outage detection and remediation is crucial for effectively operating cloud computing systems. To remediate outages, system engineers must quickly identify the causal relationships between metrics and correlate events across multiple monitoring tools. In practice, this process largely remains reactive due to the complexity and general lack of interpretability within such monitoring environments. This work presents ViSRE: an integrated visual analytics system that integrates causal and predictive models with interactive visualizations to aid in proactive cloud outage management. We develop enhanced node representations for our causal graph representation to support system engineers in performing root cause analysis and reasoning about causality chains in multi-dimensional temporal data. We report the results of a quantitative assessment of the proposed predictive models, which show good performance guarantees. To evaluate and refine our system, we conduct a study with six cloud system engineers who verify that our proposed techniques can support proactive cloud maintenance by intuitively displaying temporal relationships between predicted and raw data. By correlating and presenting data from disparate sources, ViSRE also reduces context switching costs and reduces the time spent on manually correlating events during remediation of time-critical outages.**
Video URL: https://youtu.be/ctnYNJ63yxM

***Index Terms*—Cloud Outage Prediction, Root Cause Analysis, Software Visualization**

I. Introduction

Cloud systems provide computational services such as servers and databases over a network on-demand. Cloud applications are often comprised of multiple interdependent services with connected components that rely on compute nodes (servers or virtual machines) to provide storage, processing, networking, and memory resources. System reliability is essential as outages when a service is under-performing or non-operational can result in degraded user experience and satisfaction.

The process of outage remediation heavily relies on human intervention. To effectively remediate outages, Site Reliability Engineers (SREs) need to monitor systems and identify faults in a timely manner, correlate events, identify cause and effect, and find the root cause of outages using the data produced by monitoring tools. The main challenges related to effective outage remediation are non-timely detection of faults, the amount of information generated by cloud monitoring systems, and the need to constantly context switch between monitoring applications, which increases the cognitive load on SREs.

Traditional approaches to outage management in cloud systems are reactive, meaning that SREs must deal with outages once they have already occurred [1]. Typically, numerous system monitoring tools are deployed to monitor system health, availability, latency, and performance efficiency [2]. These tools tend to focus on specific functions such as application log monitoring [3], infrastructure metric monitoring [4], [5], application metric monitoring [6], and version control systems [7]. This specialization often creates operational and data silos that hamper effective fault remediation as SREs need to monitor multiple tools and manually correlate events across them. As the complexity of cloud systems increases, manual correlation of events quickly becomes intractable.

To help SREs better interpret the large quantity of data generated by these monitoring tools, a common approach is to leverage visualization in cloud monitoring dashboards to display and analyze data. Cloud monitoring dashboards present aggregate view visualizations of data with little to no analytical support. However, as the complexity of the system increases, it becomes difficult to distinguish differences and determine what visualized information is relevant to resolving an error. Furthermore, due to the specialized nature of monitoring tools, the process of remediation involves context switching between the different visualization dashboards utilized by each tool.

To address these challenges we introduce ViSRE: a unified interactive visual analysis system for outage monitoring and triaging that aggregates data from multiple sources to *proactively* monitor the system health and predict the probability of future outages. Our main research contributions are as follows:

Outage Detection and Remediation Models: Outages are proactively detected by monitoring metrics using an unsupervised outage prediction model. ViSRE also integrates a causal inference algorithm to model the relationship between metrics to support root cause analysis of outages for remediation.

Integrated Visualization Dashboard: We further contribute ViSRE: an integrated visualization dashboard that aggregates information from multiple monitoring tools to provide a more cohesive workflow that requires less context switching across applications when remediating outages. The dashboard combines three visual components to display raw and predicted values related to potential outages. First, to enable causal inference and support root cause analysis, the dashboard contains a directed acyclic graph that includes enhanced node glyphs representing the raw and predicted time-series data for each monitoring metric. Second, to support long-term trend analysis of system metrics, we visualize the values of each metric using visually scalable horizon charts. Finally, to provide a high-level summary of the system state over a longer period and support drill down, we display a heat map of the predicted likelihood of an outage. These visualizations connect the outage prediction result to raw system data providing greater interpretability of the prediction model results.

Preliminary Evaluation: We quantitatively evaluate our outage prediction algorithm and achieve a high AUC score for different outage prediction windows and user-defined thresholds. We also evaluate the utility of ViSRE with preliminary, semi-structured interviews with six system experts who provide feedback on the effectiveness of the prototype for addressing the current challenges faced in existing cloud monitoring tools.

II. Related Work

Our work leverages prior research on outage prediction, root cause analysis, and visualizing system behavior and causality.

A. Proactive Outage Prediction and Root Cause Analysis

Proactive outage prediction approaches identify patterns of irregular behavior likely to result in future outages, thus allowing enough time to resolve errors before they impact users. Prior work proposed proactive monitoring approaches that involve predicting outages in particular settings such as disk failure [8], [9], network failure [10], or failures due to changes made in Virtual Machines (VMs) [11], [12]. These approaches do not generalize effectively as they only detect failures in specific settings, and would therefore result in missing faults (False Negatives). Profiling approaches to proactive outage prediction involve the "run-time analysis of metrics" such as memory, CPU usage, and latency to identify patterns of irregular behavior using supervised and unsupervised learning models. Supervised models [13], [14] require a large number of labeled samples, which is an inherent drawback given the sparse nature of outages in practice. Unsupervised models [15], [16] estimate forecast distributions one period ahead of a metric or group of metrics. If the probability of a new data point is higher than a predefined threshold, it is considered an anomaly. In this work, we build upon the use of unsupervised models; to improve the learning of rare outage events without labels, we treat the outage prediction problem as a "rare or extreme" event probability forecasting task

In the event of an outage or predicted outage, it is beneficial to identify the root cause to enable efficient triaging and quick remediation. Many advanced fault localization techniques depend on the notion of causality. Bayesian networks, which model the causal structure among alerting signals and how these signals lead to outages, have been proposed to support root cause analysis in cloud systems. For example, AirAlert [13] is an intelligent outage management system that models dependencies between different metrics using co-occurring alerts. A supervised learning model predicts outages as a function of the alert graph and uses the alert graph to diagnose the root cause. Other recent papers have applied different methods for inferring the causal structure in multidimensional data [17], [18], [19]. However, these works do not focus on the distributed setting whereby a single application is composed of numerous microservices each running their own processes and communicating via lightweight APIs [20].

In a real-world system deployment, each instance of a microservice generates performance indicators (e.g., throughput, latency, utilization, etc.) and hence metrics of a microservice would include the individual metrics of each such instance. Moreover, the total number of instances varies over time based on auto-scalar decisions. Off-the-shelf causal structure learning algorithms cannot handle varying numbers of instances over time. Thus, naive modeling of the causal structure of such a system, where the performance indicator of a microservice is the average of the metrics of all its deployed instances, does not perform well. To estimate the causal relationship between the application-level and the component-level metric data, we used a modified version of Fast Greedy Equivalent search (fGES) [21], which is suitable for continuous data.

B. Visualizing Program and Cloud Computing Behavior

To handle the massive amount of multivariate temporal data commonly produced by cloud systems, SREs require more analytical support in monitoring tools to increase efficiency. Recent visual analytics (VA) tools for the management of cloud computing systems have focused on integrating analysis algorithms. Xu et al. [22] leverage an anomaly detection algorithm to monitor the performance of cloud computing systems. Their system visualizes detailed performance data associated with anomalous metric results using horizon charts and line charts to support correlation analysis. Similarly, Lv et al. [23] propose a visual analytics system that uses different anomaly detection algorithms to enable domain experts to extract anomalous evolution partners in cloud system monitoring data.

While these approaches integrate analysis algorithms, they are limited to anomaly detecting and do not compute the impact of anomalies on the system behavior, i.e., will the anomalous behavior result in an outage? They are also limited to correlation analysis and do not support models of causal inference necessary for root cause analysis. Our visualization dashboard, ViSRE, integrates a predictive model to support the proactive management of cloud systems. The model quantifies the impact of anomalous behavior on the system by computing the likelihood of an outage due to the behavior. To aid in causal inference and provide greater interpretability to the predictive model, the system also integrates a causal model.

Similar to cloud monitoring approaches, current visualization tools tend to focus on specific data, i.e., logs [24], [25], metrics [22], and code change data [26], thus requiring SREs to repeatedly switch contexts while debugging. Research into cloud monitoring tools has proposed more integrated monitoring systems to minimize data redundancy and system overhead [27]. While these tools provide a unified approach by connecting to different data sources, they either propose simple aggregation approaches based on timestamps and custom queries that do not significantly reduce data complexity [5], or new architectures that can be challenging to implement [27]. In this work, we develop an integrated visual analysis dashboard, ViSRE, that enables monitoring of the cloud system within a single view by aggregating data from pre-existing monitoring systems. ViSRE uses various models to provide a prioritized visualization experience of cloud system activities; the models filter out superfluous information and focus on issues requiring attention, such as code changes that contribute to outages.

C. Visualizing Causality Chains

Visualizing causal relationships of cloud metrics is crucial to outage remediation as the process of performing root cause analysis is inherently chaining cause and effect. Judea Pearl proposed visually depicting causal model structures using directed acyclic graphs (DAGs) [28]–[30], where the nodes represent variables and the directed edges represent the relationships between them. In recent work, DAGs have gained popularity as a way of visualizing causality [31]–[34]. Furthermore, visualization research proposes the use of interactivity and animations [35]–[37] to model temporal aspects of data. However, frame-by-frame visualizations and animations often create a high cognitive load [38], which is undesirable in time-critical tasks, such as outage remediation characterized by high volume multivariate data. To visualize causality chains, Elmquist et al. proposed the use of Growing Polygons [35], which enhance the node representations on DAGs to depict the flow of information in a system using age rings in the polygon interiors. This approach assigns specific colors and sectors to each process within the polygon. Although the visualization depicts the temporal flow of causal data, it relies on a closed system where one process directly sends a message to another. However, in cloud systems, metrics and their dependencies are continuously updated. Our work similarly proposes a novel enhanced node representation for DAGs by directly encoding the time series for raw and predicted values into the nodes. The node glyph captures the intuitive temporal relationship between the raw metric data and the model predictions. Using this enhanced node representation, a user can quickly infer various aspects of the temporal causality (i.e., lags).

III. System Design Requirements

We conducted this work in collaboration with SREs at a SaaS (Software-as-a-Service)-based company that operates several large, cloud-based businesses with interdependent services and complex infrastructure. SREs monitor system availability, latency, and performance efficiency, and are primarily responsible for identifying and mitigating system errors before they severely impact critical client services or result in outages.

We conducted formative interviews with two SREs who described their general process to manage and remediate outages. The SREs stated that the current monitoring pipeline (1) is reactive, meaning that current monitoring systems often issue an alert once an error has already occurred, (2) uses manual alerting that requires domain knowledge, and (3) often results in missing faults (False Negatives) or generating too many alerts (False Positives). The SREs also noted that, in the event of an outage, the main operational challenges for root cause analysis were the data complexity, which required SREs to (1) correlate numerous metrics with complicated dependencies and (2) frequently context switch between various monitoring dashboards. Based on these interviews, we identified a set of five design requirements (**R1**-**R5**) for the proposed system to address the operational challenges faced by SREs.

R1 Proactive: Support real-time monitoring of cloud metrics for proactive outage detection. The SREs noted that current alerting approaches for outage management are reactive, thus requiring SREs to fix failures after they have started to impact users. SREs sought a more proactive approach to allow for the resolution of potential outages before they lead to a degraded user experience. To support proactive maintenance and error remediation, the system should fetch real-time system health metrics from multiple monitoring databases and integrate algorithms that proactively monitor the metrics for anomalous behavior to predict the likelihood of an outage before it occurs, allowing for sufficient remediation time.

R2 Unobtrusive: Have a low false-positive rate for outage detection. To increase the system's effectiveness and improve user trust, the systems should have a low false positive rate to reduce the level of disruption from unnecessary alerts.

R3 Unified: Reduce context switching during error remediation. To facilitate the timely remediation of errors and minimize context switching, the system should aggregate data from different data sources into a coherent interface.

R4 Insightful: Provide interpretable & actionable insights. The system should provide interpretable insights by relating model predictions to the raw metric values. The system should also relate outage forecasting data to auxiliary information such as code change data to enable more actionable insights in the event of a predicted outage. Individual probabilistic outage predictions from the the predictive model do not provide sufficiently actionable information and do not simplify the dependency-tracing task necessary for root cause analysis and remediation. Thus, for the system to be insightful, the SRE needs to be able to relate the predictions to their root causes.

R5 Interactive: Enable interactive exploration of the raw data, auxiliary data, and model. To enable users to effectively analyze the outage prediction and root cause analysis, the system should incorporate interactive visualizations that provide drill-down features and additional details on demand.

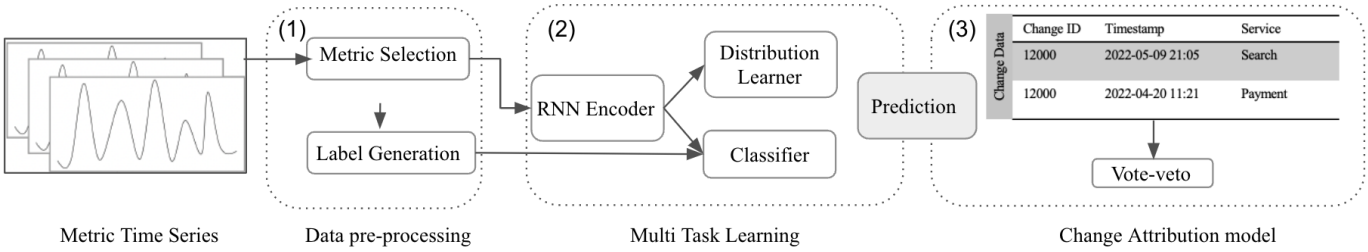

Fig. 1. An overview of the model pipeline for ViSRE: Given time series data for monitoring metrics, the system includes (1) a data processing and label generation step, (2) a Multi Task Learning prediction model using a classifier and mixture Gaussian network, and (3) a vote-veto change attribution model.

To satisfy these system requirements, we propose a unified visual analysis dashboard for proactive cloud outage management: ViSRE. ViSRE includes a data fetching component, a data processing component, a multi-task learning (MTL) component, a change blame component, and a visual analysis dashboard (Fig. 1). ViSRE fetches real-time system health metrics from multiple monitoring tools and databases (**R3: Unified**). This raw data is pre-processed before being passed to a Bi-LSTM model for outage prediction (**R1: Proactive**). To reduce the rate of false positive alerts, the model computes outage probabilities based on user-defined outage thresholds (**R2: Unobtrusive**): the more conservative the estimates the fewer the alerts. In the event of a predicted outage, ViSRE performs initial diagnostics by relating the outage event to auxiliary code change information; such diagnostics provide insights that the SREs can then use to take action such as rolling back the code change (**R4: Insightful**). The model predictions and raw data are then fed into a visualization dashboard that provides visual displays of the raw metric and model data in a manner that supports analysis by the SREs (**R5: Interactive**). In addition, we augment the predictive information generated by the model with the output of a causal model to support root cause analysis. To design ViSRE, we leveraged an iterative design process, and considered various design alternatives for displaying the causal and predictive model output as well as the raw metric data; the design of the ViSRE dashboard and our design alternatives are discussed in Section V.

IV. Modeling

Effective outage prediction aims to determine whether an outage will occur in the future based on the current state of the monitoring metrics for the system. Once an outage has been predicted or observed, SREs perform root cause analysis (RCA) to identify the fault that caused the outage [1]. Effective *proactive* fault management systems must combine both outage prediction models to preemptively identify possible outages and diagnostic tools that facilitate immediate RCA based on the current and predicted state of the system. To support these two tasks, our proposed system, ViSRE, integrates an outage prediction model for proactive fault management (**R1: Proactive**), as well as causal and code change blame models for diagnostics and insight generation (**R4: Insightful**) into a single fault management system.

A. Bi-LSTM Model for Outage Prediction

The outage prediction model aims to predict the probability that an outage will occur at time t, given the time series of N monitoring metrics: $Pr(O_t|\mathbb{M}_t)$. In this notation, $\mathbb{M} \in \mathbb{R}^{N*T}$ is a multivariate time series representing N metrics for T time steps. O_t is the outage sequence where each binary value indicates whether an outage occurred at time t.

In most operational cloud systems, outages rarely take place; thus, there are limited positive outage labels in existing cloud system data. To reliably model tail risk to predict rare outage events, we treat the outage prediction as a rare and extreme event probability forecasting task. The model input consists of metrics that define system health for a service (e.g., outage alert metrics and key performance monitoring metrics). The model learns the distribution of the metrics to forecast the future probability of an outage based on user-defined thresholds. We leverage the domain knowledge of SREs to identify the relevant system health monitoring metrics from all the metrics defined for a service. We then train the model using data samples from metric monitoring tools over three months, sampled in five-minute intervals. We pre-process the data into time series by performing linear imputation for missing values and removing rows with missing data. The data we collect can predict outages at either an instance-level or a service-level.

However, instance-level data raises several implementation challenges as instances have a variable life span ranging from one day to more than a month, resulting in dimensional inconsistency in the data. To address the dimensional inconsistency, we perform service-level aggregation by computing averages across the instances for each metric at each time step. This aggregation relies on the assumption that the majority of the instances at a particular time can be assumed to be random samples drawn from the distribution.

An additional data challenge occurs because outage event sparsity leads to a data imbalance problem since there are few to no positive labels for training. Current manual intervention approaches further suppress the outage tail events either by preventing them from happening or quickly resolving them before they have a disproportionate impact. To suppress the influence of the manual intervention on the training data, we generate training labels during the data preprocessing step using quantile-based thresholds that act as proxies for tail events. The system recommended threshold is the 95th

percentile of values; however, this threshold can be adjusted based on expert knowledge of Service Level Agreements.

We utilize a Bi-LSTM to model the tail distributions for outage prediction. RNN models like LSTMs and Bi-LSTMs have been shown to outperform conventional methods in time series modeling [39]. The model is comprised of a Bi-LSTM sequence model followed by a Classification layer and a Mixture Density Network (MDN) layer. The classification layer uses the binary classification labels generated during pre-processing, and optimizes a Binary Cross-Entropy (BCE) loss [40] and a variation of BCE loss for extreme event forecasting called Extreme Value Loss (EVL) [41], [42]. In this notation, N is the size of the batch, $y \in 0, 1$ is the true value of the label and $\hat{y}_i$ is the value predicted by our model:

$$\mathbb{L}_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i * \hat{y}_i + (1 - y_i) * log(1 - \hat{y}_i) \quad (1)$$

For the EVL, β_0 is the proportion of normal events in the batch and β_1 is the proportion of extreme events in the dataset. We use $\gamma = 2$ for the experiments:

$$\begin{aligned}\mathbb{L}_{EVL} = -\frac{1}{N}\sum_{i=1}^{N} \beta_0 * \Big[1 - \frac{y_i}{\gamma}\Big]^{\gamma} \hat{y}_i \log y_i \\ + \beta_1 * \Big[1 - \frac{1 - y_i}{\gamma}\Big]^{\gamma} (1 - \hat{y}_i) \log(1 - y_i)\end{aligned} \quad (2)$$

The MDN combines a Deep Neural Network (DNN) and a mixture of distributions. The DNN is used to learn the mixture parameters (μ, σ) and the mixing coefficient α. The Gaussian mixture model is capable of modeling any arbitrary probability distribution [43]. Formally, c denotes the index of the corresponding mixture component, α is the mixing parameter, D is the distribution to be used (in our case Gaussian), and λ denotes the parameters of the distribution $\mathbb{D}$; $\mu(x)$, $\sigma(x)$ in our case.

$$p(y|x) = \sum_{c=1}^{C} \alpha_c(x)\mathbb{D}(y|\lambda_{1,c}(x), \lambda_{2,c}(x), ...) \quad (3)$$

The MDN layer utilizes the forecast values of the metrics as labels and optimizes the mean negative log-likelihood of y given the mixture parameters, where $\mathbb{R}$ corresponds to the realm of possibilities:

$$\arg\min_{\theta} l(\theta) = -\frac{1}{|\mathbb{R}|}\sum_{x,y \in \mathbb{R}} \log p(y|x) \quad (4)$$

During training, losses are optimized simultaneously to predict the distribution of metrics k time steps ahead. The model achieves multitask learning where the MDN loss optimization enables learning of predicted distributions; the binary optimization enables learning of the tail and non-tail classification. Using the forecasted distribution of the metrics, we can compute the probability of the outage based on any threshold queried by the user. The probabilistic outage prediction provides greater interpretability than the binary labels.

B. Bayesian Causal Network for Root Cause Analysis

Outages in cloud systems are often associated with multiple alerting signals. In the event of an outage, the primary inference task performed by SREs is root cause analysis (RCA). The goal of RCA is to establish the causal relationship between various alerting signals in order to infer the proximate and ultimate causes of the outage. Typically this inference task is performed manually by correlating the time-series information of the alerting signals. However, as the number of alerts and dependencies increases, manually correlating the time series becomes intractable. Within the proactive outage management pipeline, the probabilistic prediction is insufficient for causal inference and RCA. To aid in inference during root cause analysis, we use a Bayesian causal network to model the cause and effect relationships between alerting signals.

The Bayesian causal graph for a microservice is a directed graph where nodes are metrics, and a directed edge between two metrics represents the causal relation. In the event of an outage, the Bayesian model can be used to identify the root cause directly by calculating the conditional dependence between an alerting signal and an outage. However, in ViSRE, we do not directly compute the conditional probabilities; instead, we use the inferred causal model to support visual analysis of the causal relationships between metrics during predicted, as opposed to observed, outages.

To infer an instance-level Bayesian causal graph we modify the Fast Greedy Equivalence Search (fGES) algorithm [21]. fGES is a modification of the Greedy Equivalent Search (GES) algorithm designed for discovering directed acyclic graphs on random variables from sample values [44], [45]. GES heuristically searches the space of all causal Bayesian networks and returns the model with the highest Bayesian score. The GES computation consists of two phases starting from an empty graph: a forward and backward phase. In the *forward phase*, directed edges are added iteratively until no addition increases the score. In the *backward phase*, edges are removed until no edge removal can increase the score. fGES is a parallelized version of GES, where the scores for each fragment of the causal graph are cached to optimize the computation. Building a causal graph at the instance level can be tricky due to the changing number of instances, so we use domain knowledge to reduce this complexity. Since a load balancer distributes the workload evenly to all the instances of a service we studied, we assume that instances are IID conditional on service level workload. This assumption reduces complexity because we can consider data generated by different instances as random draws from the same distribution. We use a service call graph to create a list of prohibited edges which is used in fGES. We use penalized Bayesian Information Criterion (BIC) as the score function for graph structure detection. We modify the BIC-based score by running a pooled regression model of x_{ijt} on parent metrics, $\mathcal{P}(x_{ijt})$. Formally, the score function is defined as follows, where, L is the likelihood function, ρ is a penalty term for the graph complexity, k number of parameters in f_i, and n_i is the number of observations in the pooled data.

We found $\rho = 2$ to be optimal in our experiments:

$$Score(x_{ijt}, \mathcal{P}(x_{ijt})) = \\ -2\sum_{j,t} \log\left(L(f_i(\mathcal{P}(x_{ijt}))|x_{ijt}, \mathcal{P}(x_{ijt}))\right) + \rho k \log n_i \quad (5)$$

C. Change Blame Scoring

One of the most common causes of outages in cloud systems is code changes [46]. Typically, to identify which code change is responsible for an outage, engineers manually correlate changes from a version control system to trends in the system health metrics. However, cloud system change histories are often comprised of a large number of entries, thus making it difficult and time-consuming to identify the code change that is most relevant to an outage during debugging.

To address this challenge, our modeling pipeline uses predicted outages and deployment event data to find the code change most responsible for the outage. Based on the approach proposed by Li et al. [11], the model uses a vote-veto approach to establish the top-k changes responsible for an outage alert. The model uses four different time windows (1 hour, 24 hours, 72 hours, and the difference between deployments and the latest data points). The model relates the predicted outages to change events deployed in the time windows by performing a vote-veto computation. Any predictions made after a deployment event votes for the deployment event, while predictions before the deployment event vetoes it. The vote-veto scores are aggregated based on the equations below, where a represents the outage alert, c represents the change, WD represents the four different time windows, P_i represents the vote score for the ith time window, B represents the veto score, k refers to the different outage alerts predicted over the time window, and w_i refers to the weight of the time widows which are weighted exponentially in decreasing order from w_1 to w_4. The top three scored changes from the lists of changes blamed are made available to the users.

$$\begin{aligned} P_i &= \sum_k V(a, c|WD_i) \\ B &= \sum_k V(a, c|WD_{-1}) \\ Score(a, c) &= \sum_{i\in[1,4]} w_i log\Big(\frac{P_i - B + 1}{B + 1}\Big) \end{aligned} \quad (6)$$

V. ViSRE: A Prototype Dashboard for Proactive Cloud Computing and Fault Management

The user interface is comprised of three functional views: the Parameter Interaction View (Section 5A), the Temporal View (Section 5B), and the Auxiliary Data View (Section 5C). We followed an iterative design process, and considered various design alternatives for displaying the causal and predictive model output, as well as the raw metric data which we updated based on feedback from SREs. We detail the design alternatives considered in the temporal view section (Section 5B).

A. Parameter Interaction View

The *Paramemter Interaction View* (Fig. 2B) allows users to select a threshold value for each metric. The model (Section 4B) uses these thresholds to compute the likelihood of an outage. The selection of meaningful thresholds is critical to the effective operation of the outage prediction models. Setting thresholds that are too liberal would result in an increased false-positive rate; while, setting thresholds that are too conservative may result in a high false-negative rate. Additionally, threshold values that are too conservative would not allow for enough time to remedy errors. Typically, SREs rely on analysis of past trend data to set these thresholds.

To help SREs decide how to set the threshold values, we visualize boxplots in the configuration drawer that show the distribution of metric values based on past data (Fig 2B). The minimum and maximum value of each metric over the most recent two week data collection period are presented on either side of the box plot. This visualization enables the SREs to quickly infer the typical distribution of values for the metrics. The default values presented to the users before they adjust them are the current thresholds defined for the metrics in the SREs' existing monitoring tools. The threshold values can be adjusted using a slider directly below the boxplot. This slider is on the same scale as the boxplot to enable the SRE to reason about their chosen threshold value relative to the distribution of raw metric values. Updating the thresholds updates all visualizations in the Temporal Data View.

B. Temporal Data View

The *Temporal Data View* (Fig. 2C-F) consists of three visualizations of the temporal metric and model data: a directed acyclic graph (Fig. 2C) that visualizes the causal relationships between metrics with node enhancements (Fig. 2D) to show the raw and predicted values for that metric, a horizon chart (Fig. 2E) that visualizes the raw values for each metric, and a probability overview chart that visualizes the overall likelihood of an outage (Fig. 2F).

(1) Directed Acylic Graph: Root cause analysis requires reasoning about events in a sequential manner. To perform root cause analysis, typically, SREs manually correlate metric information using such time series visualizations and queries. Manual correlation is time-consuming and relies on expert knowledge of dependencies, which hampers debugging efforts. Although the space-efficient color-coded representations of horizon charts can make metric correlation easier to infer, the correlations are not sufficient to infer causality. A visualization that effectively supports root cause analysis should enable the quick identification of causal direction and dependencies.

Motivated by prior work [47], we use a top-down sequential ordering of a directed acyclic graph (DAG), as this visually represents the parent-child relationship between metrics with the relative node positions (Fig. 2C). During the initial design phases, we used a simple DAG to visualize the causal relationship. To conserve the display space, we directly encoded the results of the predictive model onto the metric nodes. Since

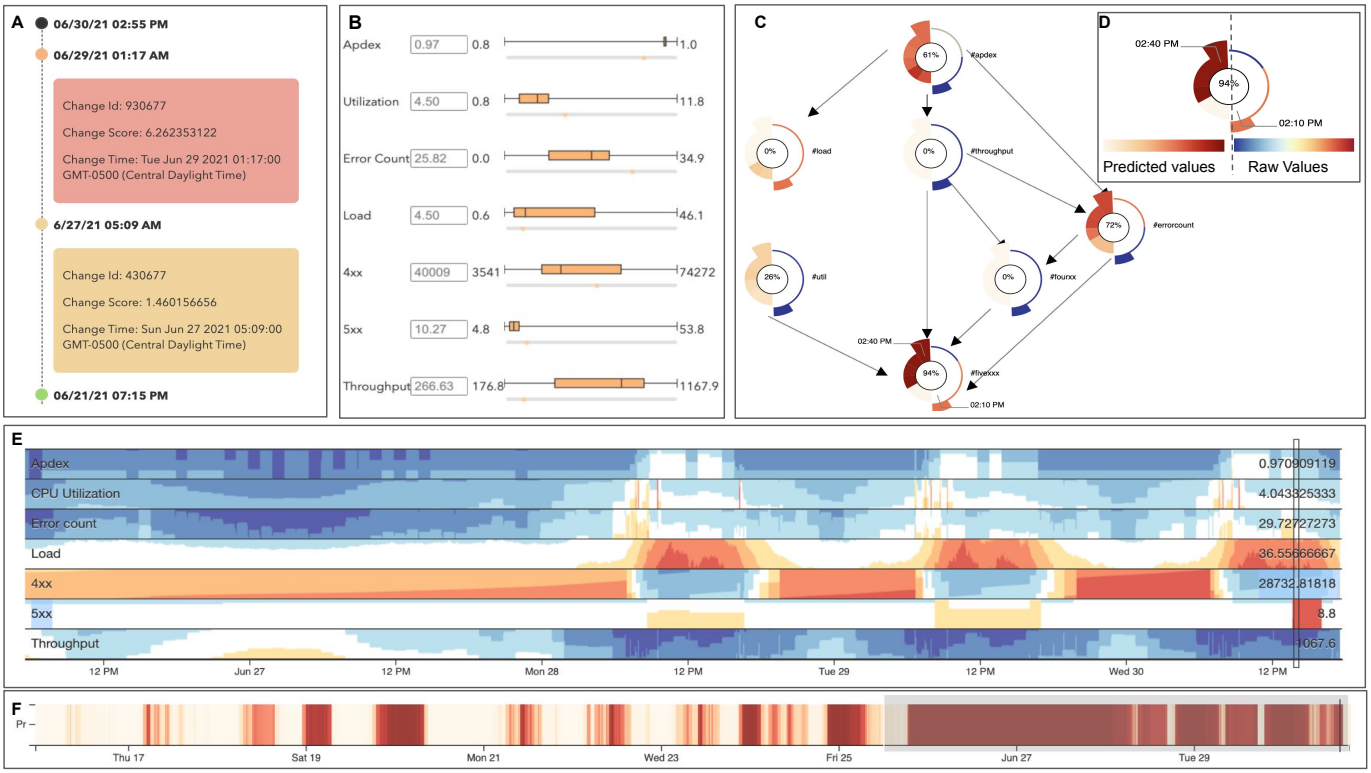

Fig. 2. The ViSRE interface includes five components: (A) The code change timeline; (B) The threshold configuration drawer; (C) The causal metric directed acyclic graph with (D) enhanced node glyphs showing the raw and predicted values in two half arcs; (E) The horizon charts of the raw metric values with an interactive bounding box corresponding to the region visualized in the causal graph; (F) The probability overview chart for long-range values with an interactive bounding box corresponding to the region visualized in the horizon chart. In this example, metric glyphs on the causal graph show increased load and error count and reduced user satisfaction (`#apdex`) in a 30-minute analysis interval. At the beginning of the analysis period, the grey color in the left half of the `#apdex` node glyph indicates an anomalous metric value. The outage prediction value on the right half of the `#apdex` node indicates that an outage is likely within 30 minutes. Hovering over a region of high outage probability in the glyph generates the code change timeline (A). Later in the analysis period, a spike in server errors (`#5xx`) results in an outage, visualized in the left arc of the `#5xx` glyph that transitions from blue to red. The probability overview chart (F) shows a period of high outage prediction around June 26th; brushing this region updates the horizon chart to visualize the corresponding metric values, which show extended periods of threshold violation for client errors (`#4xx`) interspersed with shorter periods for server errors (`#5xx`).

the predictive model values have a temporal component, we used a frame-by-frame animation to display the values along with a timeline to highlight periods of high outage prediction probabilities. However, SREs noted that keeping track of the metric incidents in individual frames was challenging; therefore, encoding the temporal component directly into the causal graph would be beneficial. Based on this feedback from SREs, we designed an enhanced node representation for the DAG to support (1) reasoning about the temporal aspects of the data, i.e., lags and causality chains with minimal frame changes, and (2) reasoning about the context between the predicted and the raw values. We designed a donut-based glyph that directly encodes the temporal aspect of the raw and predicted values onto the DAG nodes (Fig. 2D).

Implementation: To render the DAG we used the Sugiyama Layout algorithm [48] implemented in D3, which assigns each node to a vertical position with edges directed downwards. To provide the raw data context for the model predictions, the node glyph directly encodes both metric time series data and model-predicted data by visualizing them in the two half arcs surrounding the node. The most recent prediction value is presented in the text at the center of the node, additional past data is provided in the arcs, and time labels are provided at the start and the end of each arc to communicate the time intervals represented by each. Each half arc contains smaller equal-angled arcs that represent equal-sized time intervals. The size of each arc can be adjusted to support the display of data at varying levels of time granularity. Reading the arcs clockwise the first half arc represents the raw data for the metrics over a t_1 to t_k time interval (i.e., in Fig. 3A from 3:00pm to 3:25pm). The color scale for the raw data arc is the same as the color scale on the horizon chart. For each time interval t, the model predicts the likelihood of an outage 30 minutes into the future at time $t+30$, thus every raw value at time t corresponds to a predicted value at time $t+30$. The second half arc corresponds to the predicted values for the outage in the upcoming t_1+30 minutes to t_k+30 minutes time interval (i.e., in Fig. 3A from 3:30pm to 3:55pm). The color scale for the predicted values arc is the same as the color scale for the probability overview chart. Each predicted value in the second half arc corresponds to the diametrically opposed raw value in the first arc.

Interaction: Hovering over the smaller arcs in the predicted value arc animates the corresponding raw value by increasing its size relative to the other values (Fig. 3). Additionally, hov-

ering over the arcs updates the text at the node center to show the predicted value for the hovered arc. To support inference into how the raw values and predicted values are related across the metrics all nodes in the graph are linked; thus, if a user hovers over a region in one node, the corresponding predicted and raw values in all other nodes are animated (Fig. 2C). The default view of the glyph is the lowest level of drill-down which shows a 30-minute raw data interval and the corresponding 30-minute prediction interval that corresponds to the predictive window of the model (Fig. 3A). However, if users are interested in observing longer time periods the glyph supports varying degrees of time granularity (Fig. 3B-C).

(2) Horizon Chart: One of the most important factors for understanding the overall health of the system is to monitor the raw values of multiple metrics simultaneously. During this process, SREs leverage their knowledge of the standard system behavior to specify acceptable thresholds for the metric values. When designing this visualization, we considered four design alternatives to display the metric data: individual plots, small multiple-line charts, multi-line plots, and horizon charts.SREs primarily use metric visualizations to (1) establish the periods during which metric values exceeded their respective thresholds and (2) correlate different metric values to establish a precedence relationship. Thus, SREs require visualizations that support the quick identification of threshold breaches and enable faster metric correlation.

We took these requirements into account when considering our design alternatives. While the level of detail in the large, individual metric plots supported anomalous value inference, the visualizations took up a lot of space; thus, they did not effectively support metric correlation as the SREs needed to scroll back and forth to compare the metric values. Although more space-efficient than individual plots, multi-line plots did not enable either the quick inference of threshold breaches or metric correlation as the graph was cluttered and difficult to read. Additionally, the order of magnitude variation in the metric values led to scaling issues that made it difficult to decipher the individual values of some metrics. Small multiple plots, although space-efficient, did not support the inference of threshold breaches due to their diminished size and resulting lack of detail. We therefore chose horizon charts [49] for their space-efficient characteristics and improved distinguishability of values, in order to easily represent all metric values in a single interface without scrolling (**R3: Unified**). Furthermore, we utilize the user-defined thresholds to compute the color encoding in the horizon chart; this color encoding enables users to quickly identify time periods with anomalous values.

Implementation: Values are represented using a diverging blue-white-red color scale to differentiate values that diverge from the user-defined threshold. While the values for anomalous behavior relative to the threshold varies for each metric, in general, the horizon chart can be read as follows: the midpoint of the color scale (white) corresponds to the threshold value, blue indicates normal patterns and red indicates anomalous patterns; the greater the color salience the greater the degree to which the pattern violates or conforms to a threshold condition. We leverage D3's cubism.js plugin [50] in order to properly handle real-time streaming data in cases where the user is not actively performing analysis on past data.

Interaction: A black bounding box is overlaid on the horizon chart to show users the time period of data that is visualized on the causal graph (Fig. 2C). Users can interactively move this bounding box to update the time period on the causal graph.

(3) Probability Overview Chart: SREs emphasized the need for visualizations at varying levels of granularity to support drill down (**R5: Interactive**). A visualization at the highest level of granularity was required to (1) provide a summary of the state of the system, (2) support inference of longer-term predictive model trends, and (3) facilitate the quick identification of the temporal distribution of outage prediction values. To aid in these tasks we designed a probability overview chart (Fig. 2F) that displays the aggregated distribution of the prediction values. During the initial phases of design, the probability chart was visualized using a simple line graph. However, due to space constraints that limited the size of the graph and the corresponding axis, the line graph did not adequately support the discovery of distinctive patterns and outliers in the predicted values. We updated the probability graph to a binned heat map visualization, where the color encoding can better enable the user to quickly identify periods with high predicted probability of an outage.

Implementation: The probability overview is implemented as a binned heat map [51], [52] and visualizes aggregated prediction values as fixed-width, vertical ticks, the size of which can be increased or decreased to show a smaller or larger time frame. To support the display of a longer time frame relative to the horizon charts, we set the tick width to accommodate two weeks' worth of data by default. The hue of each tick encodes the probability value. Ticks are colored using a sequential single-hue color scale. Low probability values correspond to lighter colors, while high probability values map to darker colors. We aggregate the probability values represented in the chart by selecting the maximum value of the predicted outage across the seven system health metrics visualized in the system at five-minute time intervals.

Interactions: The probability overview displays two weeks of data. The horizon chart and causal graph are visually scalable to support varying levels of drill down and enable the identification of lags in the causality chains. By default, the horizon chart displays four days of metric data, but can display up to two weeks by adjusting the brush on the probability overview. The causal graph can visualize up to six hours of metric and prediction data, but displays 30 minutes by default; the bounding box on the horizon chart shows the time period visualized on the causal graph and can be updated interactively.

C. Code Change Timeline

In the event of a predicted outage, the modeling pipeline uses a vote-veto mechanism to identify the code changes that most likely caused an outage. For each time period where there

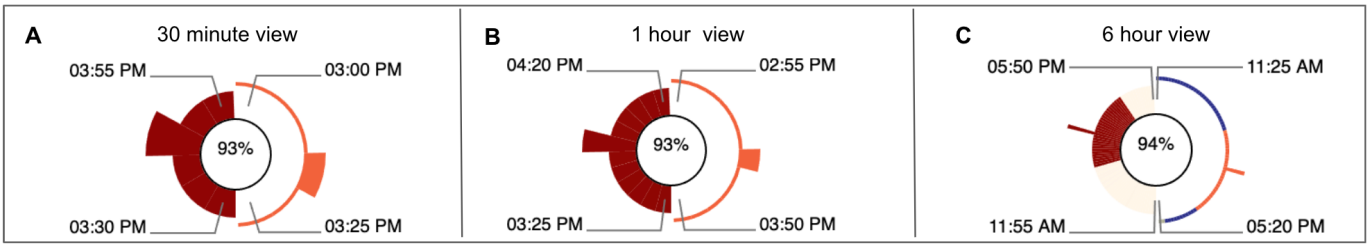

Fig. 3. Enhanced node glyph visualizing multiple time periods. (A) The default 30-minute view where 30 minutes corresponding to the raw data values (on the right), and 30 minutes corresponding to the predicted metric values (on the left); this view can be read similar to a clock. (B) one hour or (C) six hours.

is a high predicted probability of an outage based on the value of each metric, users can view the timeline detailing the related code change by hovering over the high prediction region in the DAG nodes. The code change timeline view contains the timeline visualization of the code change data. To aid users with remediation, a tooltip with detailed information on the code change such as the timestamp, the code change ID from the version control system (VCS), and the model assigned change score are also visualized on the timeline. The timeline visualization is a nonlinear timeline where values are spaced out evenly, instead of a real-time timeline to support a more compact representation. A similar approach was used by [26] to visualize the code change history. We color the change marker on the timeline and the tooltip based on the model assigned change score, which represents the degree to which the outages were likely caused by the code change. Red indicates that the code change is most likely responsible for the outage, orange indicates it was somewhat likely responsible, and green indicates it was the least likely responsible.

VI. Evaluation

We evaluated the proposed algorithmic solutions quantitatively and performed user studies on the ViSRE UI with six SREs.

A. Data Preliminaries

We used performance data from a cloud microservice to train and evaluate the models and perform user studies. The outage prediction model and the causal model were trained on three months of data. The prediction model was tested on one month of data. We used two weeks of the test data for the user studies.

B. Quantitative Evaluation

We evaluate the performance of our proposed outage prediction algorithm through three experiments. In the first experiment, we compare the accuracy of our proposed approach, Bi-LSTM, relative to different Recurrent Neural Network model architectures: simple LSTM, stacked LSTM, and stacked Bi-LSTM. In the second experiment, we compare the proposed multitask learning model, a Bi-LSTM with MDN and Classification Layer trained with either Binary Cross Entropy loss (Bi-LSTM MTL + BCE) or Extreme Value loss (Bi-LSTM MTL + EVL), to Bi-LSTM models with either a Mixture Density Layer (Bi-LSTM MDN) or a classification layer (Bi-LSTM Classifier). In the third experiment, we show the proposed model trained on 95 percentile thresholds generalized to 97 and 99 percentile thresholds, indicating that the distribution learned by the MDN can be used for an arbitrary user threshold. Detailed AUC scores for each model can be found in Table 1 with summaries of the results presented below.

TABLE I
The AUC scores of the proposed outage prediction model (Bi-LSTM MTL) across three quantitative experiments.

Experiment 1	5 minutes	15 minutes	30 minutes
Bi-LSTM	**0.974**	**0.968**	**0.959**
LSTM	0.950	0.950	0.948
Stacked LSTM	0.961	0.925	0.914
Stacked Bi-LSTM	0.956	0.933	0.918

Experiment 2	5 minutes	15 minutes	30 minutes
Bi-LSTM Classifier	0.909	0.930	0.927
Bi-LSTM MDN	0.967	0.956	0.951
Bi-LSTM MTL + BCE	**0.982**	**0.976**	**0.972**
Bi-LSTM MTL + EVL	**0.981**	**0.977**	**0.975**

Experiment 3	95	97	99
Bi-LSTM MTL + BCE	**0.982**	**0.972**	**0.962**
Bi-LSTM MTL + EVL	**0.981**	**0.974**	**0.960**

Experiment 1: Different RNN architectures. The bi-directional LSTM (Bi-LSTM) performs better than simple LSTM with 0.967 AUC as compared to 0.949 AUC, on average. The stacked LSTM and stacked Bi-LSTM models perform worse than Bi-LSTM model due to overfitting with an average AUC of 0.933 and 0.936 on average respectively.

Experiment 2: Different Models. The addition of a Mixture Density Layer to the classifier improves the performance from an average AUC of 0.922 to 0.977. The addition of a classifier layer to the MDN model helps learn the tail of the distribution, thus improving average AUC from 0.958 to 0.977. Models trained using EVL loss and BCE perform similarly.

Experiment 3: Different Percentile Thresholds The model trained on a 95 percentile threshold adapts to higher percentile thresholds of 97 and 99, achieving an AUC of 0.972 and 0.962 respectively for the model trained with BCE loss, and 0.974 and 0.960 for the model trained with EVL loss function.

C. Qualitative Evaluation

We performed semi-structured user studies with six SREs (**E1-E6**) to evaluate the effectiveness of ViSRE for proactive outage prediction and root cause analysis. In place of a longer follow-up study using live outage data, we performed this preliminary user evaluation by asking participants to walk through a usage scenario that depicted data from a prior outage event (Fig 2). Each session lasted approximately 45 minutes. The first 15 minutes were used to describe the system, during which time we provided a high-level overview of the models (purpose, input, and output), the data (seven system monitoring metrics of a cloud microservice), and an explanation of the visualizations. In the remaining 30 minutes, participants were provided with a link to ViSRE and were asked to perform tasks such as identifying causality chains, anomalous metric values, and the likelihood of an outage. We selected the task questions to reflect a variety of analysis tasks that might arise during outage remediation. Once participants finished responding to the structured questions, they were asked to provide general feedback on the usability of the system.

(1) Overall System Utility: Several SREs noted that the model predictions were particularly helpful; for example, E3 noted that *"Predictions can allow for future planning and course correction in the event of an error before they negatively impact users"* (**R1: Proactive**). The SREs also unanimously noted that the code change integration was useful for reducing context switching (**R3: Unified**) and to cut back on labor spent in identifying problematic code changes (**R4: Insightful**). **E4** explained that *"If it can directly tell the commit, we can just go and roll back that commit and the outage is over in a minute. It can also save man-hours because only the developer who made the change can go and check the code and not every developer who has made recent changes to the system"*. **E1** noted that metric aggregation was also helpful in reducing context switching: *"First I have to go check 4xx and 5xx in Splunk then I have to go check apdex in New Relic, so having those metrics in one place is kinda cool"* (**R3: Unified**).

(2) Visualization Dashboard: The SREs generally found the visualizations in the dashboard were well-rationalized and satisfied the design requirements outlined in Section III.

Parameter View: E5 noted that the threshold setting mechanism helped them infer the behavior of the model. **E6** stated that the box plot and corresponding slider positioning helped set meaningful thresholds that balance the trade-off between too many false positives and false negatives (**R2: Unobtrusive**), while allowing for enough time for error remediation.

Probability Overview Chart: SREs used the probability chart to brush regions of high predicted outage probability in order to view the associated horizon chart, which provided more granular details on the metrics (**R5: Interactive**). **E5** felt that the *"Most helpful [visualization] is the probability timeline"* as it can enable high-level inference of all model predictions and metrics in the past to identify the *"problematic durations"*.

Horizon Chart: Overall, four of six participants noted that the horizon chart was helpful and intuitive. These SREs found that the horizon charts *"provide a clear indication of the general health of the system"* (**E2**) and are useful for identifying metrics that violated the user-defined thresholds. **E1** noted that *"using the red marks only I can identify a region of anomalous behavior"* (**R4: Insightful**). However, some experts (**E4** and **E6**) noted that including support for a more typical timeline visualization (i.e., a line chart) in addition to the horizon chart would have been helpful. While this design was initially considered, it was discarded due to reduced metric comparability, but may be useful to include due to the familiarity of the chart type.

Causal Graph: Using the causal graph and the interactions that it supports, SREs were able to quickly identify the causal relationships and causality chains between metrics in the instances where the model predicted a high outage probability; for example, in the time period represented in Fig 2b, both **E1** and **E3** identified the Apdex, 4xx, and 5xx causality chain. Using the two half arcs on the node, the SREs were able to identify that high values of predicted outages corresponded to violations or likely violations of thresholds in the raw data.

VII. Discussion, Limitations, and Future Work

Overall, the SREs' feedback on ViSRE was positive, but there remain several areas for improvement. For example, **E3** noted that the model-inferred causal structure might be missing some relationships, and suggested the ability to manually update the edges in the causal graph. To this end, future work aims to incorporate graph algorithms that support layout changes and aim to minimize computational overhead and edge crossings.

Although all SREs could use the system following a short introduction, some noted that a novice user might struggle to interpret the visualizations thus ViSRE should include a tutorial or on-demand tool tips to guide novice users. In the event of an outage, several SREs noted that it would be beneficial for ViSRE to infer exactly which service instance was causing anomalous values. However, our prediction model performs service-level aggregation to address the dimensional inconsistency at an instance level. Due to this aggregation, the model cannot identify which instance is causing anomalous values, thus limiting its inferential power. Future work aims to enhance the model to enable instance-level predictions; this enhancement can be done, for example, by training the model on the maximum value of instances as opposed to the average.

In the current remediation pipeline, information is spread across multiple tools; thus, we cannot make one-to-one comparisons of ViSRE to a single system. As noted in Sections II and IV, outages are rare events in healthy operational cloud systems such as the one we study. Thus, a longer follow-up study is required to collect sufficient quantitative data to meaningfully observe user behavior. We aim to present the results of a long-term quantitative study in future work.

REFERENCES

[1] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," ACM Computing Surveys (CSUR), vol. 42, no. 3, pp. 1–42, 2010.

[2] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," Computer Networks, vol. 57, no. 9, pp. 2093–2115, 2013.

[3] Splunk. [Online]. Available: https://www.splunk.com/

[4] Promethues. [Online]. Available: https://prometheus.io/

[5] Grafana. [Online]. Available: https://grafana.com/

[6] Newrelic. [Online]. Available: https://newrelic.com/

[7] Newrelic. [Online]. Available: github.com

[8] J. Xiao, Z. Xiong, S. Wu, Y. Yi, H. Jin, and K. Hu, "Disk failure prediction in data centers via online learning," in Proceedings of the 47th International Conference on Parallel Processing, 2018, pp. 1–10.

[9] S. Lu, B. Luo, T. Patel, Y. Yao, D. Tiwari, and W. Shi, "Making disk failure predictions smarter!" in 18th {USENIX} Conference on File and Storage Technologies ({FAST} 20), 2020, pp. 151–167.

[10] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in Proceedings of the ACM SIGCOMM 2011 Conference, 2011, pp. 350–361.

[11] Z. Li, Q. Cheng, K. Hsieh, Y. Dang, P. Huang, P. Singh, X. Yang, Q. Lin, Y. Wu, S. Levy et al., "Gandalf: An intelligent, end-to-end analytics service for safe deployment in large-scale cloud infrastructure," in 17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20), 2020, pp. 389–402.

[12] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, Z. Zang, X. Jing, and M. Feng, "Funnel: Assessing software changes in web-based services," IEEE Transactions on Services Computing, vol. 11, no. 1, pp. 34–48, 2016.

[13] Y. Chen, X. Yang, Q. Lin, H. Zhang, F. Gao, Z. Xu, Y. Dang, D. Zhang, H. Dong, Y. Xu et al., "Outage prediction and diagnosis for cloud service systems," in The World Wide Web Conference, 2019, pp. 2659–2665.

[14] P. Park, P. D. Marco, H. Shin, and J. Bang, "Fault detection and diagnosis using combined autoencoder and long short-term memory network," Sensors, vol. 19, no. 21, p. 4612, 2019.

[15] Q. Guan and S. Fu, "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures," in 2013 IEEE 32nd International Symposium on Reliable Distributed Systems. IEEE, 2013, pp. 205–214.

[16] X. Zhang, J. Kim, Q. Lin, K. Lim, S. O. Kanaujia, Y. Xu, K. Jamieson, A. Albarghouthi, S. Qin, M. J. Freedman et al., "Cross-dataset time series anomaly detection for cloud systems," in 2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19), 2019, pp. 1063–1076.

[17] P. Chen, Y. Qi, P. Zheng, and D. Hou, "Causeinfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems," in IEEE INFOCOM 2014-IEEE Conference on Computer Communications. IEEE, 2014, pp. 1887–1895.

[18] Y. Meng, S. Zhang, Y. Sun, R. Zhang, Z. Hu, Y. Zhang, C. Jia, Z. Wang, and D. Pei, "Localizing failure root causes in a microservice through causality inference," in 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS). IEEE, 2020, pp. 1–10.

[19] P. Wang, J. Xu, M. Ma, W. Lin, D. Pan, Y. Wang, and P. Chen, "Cloudranger: root cause identification for cloud native systems," in 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). IEEE, 2018, pp. 492–502.

[20] E. Wolff, Microservices: flexible software architecture. Addison-Wesley Professional, 2016.

[21] J. Ramsey, M. Glymour, R. Sanchez-Romero, and C. Glymour, "A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images," International journal of data science and analytics, vol. 3, no. 2, pp. 121–129, 2017.

[22] K. Xu, Y. Wang, L. Yang, Y. Wang, B. Qiao, S. Qin, Y. Xu, H. Zhang, and H. Qu, "Clouddet: Interactive visual analysis of anomalous performances in cloud computing systems," IEEE transactions on visualization and computer graphics, vol. 26, no. 1, pp. 1107–1117, 2019.

[23] C. Lv, K. Ren, H. Zhang, J. Fu, and Y. Lin, "Pevis: visual analytics of potential anomaly pattern evolution for temporal multivariate data," Journal of Visualization, pp. 1–17, 2021.

[24] C. Xie, W. Xu, and K. Mueller, "A visual analytics framework for the detection of anomalous call stack trees in high performance computing applications," IEEE transactions on visualization and computer graphics, vol. 25, no. 1, pp. 215–224, 2018.

[25] M. Sun, G. Convertino, and M. Detweiler, "Designing a unified cloud log analytics platform," in 2016 International Conference on Collaboration Technologies and Systems (CTS). IEEE, 2016, pp. 257–266.

[26] Y. Yoon, B. A. Myers, and S. Koo, "Visualization of fine-grained code change history," in 2013 IEEE Symposium on Visual Languages and Human Centric Computing. IEEE, 2013, pp. 119–126.

[27] J. Montes, A. Sánchez, B. Memishi, M. S. Pérez, and G. Antoniu, "Gmone: A complete approach to cloud monitoring," Future Generation Computer Systems, vol. 29, no. 8, pp. 2026–2040, 2013.

[28] J. Pearl, "Causal diagrams for empirical research," Biometrika, vol. 82, no. 4, pp. 669–688, 1995.

[29] ——, Causality. Cambridge university press, 2009.

[30] ——, "Causal inference," Causality: Objectives and Assessment, pp. 39–58, 2010.

[31] J. Wang and K. Mueller, "The visual causality analyst: An interactive interface for causal reasoning," IEEE transactions on visualization and computer graphics, vol. 22, no. 1, pp. 230–239, 2015.

[32] ——, "Visual causality analysis made practical," in 2017 IEEE Conference on Visual Analytics Science and Technology (VAST). IEEE, 2017, pp. 151–161.

[33] X. Xie, F. Du, and Y. Wu, "A visual analytics approach for exploratory causal analysis: Exploration, validation, and applications," IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 1448–1458, 2020.

[34] C.-H. E. Yen, A. Parameswaran, and W.-T. Fu, "An exploratory user study of visual causality analysis," in Computer Graphics Forum, vol. 38, no. 3. Wiley Online Library, 2019, pp. 173–184.

[35] N. Elmqvist and P. Tsigas, "Causality visualization using animated growing polygons," in IEEE Symposium on Information Visualization 2003 (IEEE Cat. No. 03TH8714). IEEE, 2003, pp. 189–196.

[36] Z. Jin, S. Guo, N. Chen, D. Weiskopf, D. Gotz, and N. Cao, "Visual causality analysis of event sequence data," IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 1343–1352, 2020.

[37] N. R. Kadaba, P. P. Irani, and J. Leboe, "Visualizing causal semantics using animations," IEEE Transactions on Visualization and Computer Graphics, vol. 13, no. 6, pp. 1254–1261, 2007.

[38] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko, "Effectiveness of animation in trend visualization," IEEE transactions on visualization and computer graphics, vol. 14, no. 6, pp. 1325–1332, 2008.

[39] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2018, pp. 1394–1401.

[40] K. P. Murphy, Machine learning: a probabilistic perspective. MIT press, 2012.

[41] D. Ding, M. Zhang, X. Pan, M. Yang, and X. He, "Modeling extreme events in time series prediction," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1114–1122.

[42] S. Coles, J. Bawa, L. Trenner, and P. Dorazio, An introduction to statistical modeling of extreme values. Springer, 2001, vol. 208.

[43] D. A. Reynolds, "Gaussian mixture models." Encyclopedia of biometrics, vol. 741, pp. 659–663, 2009.

[44] C. Meek, "Graphical models: Selecting causal and statistical models," Ph.D. dissertation, PhD thesis, Carnegie Mellon University, 1997.

[45] D. M. Chickering, "Learning equivalence classes of bayesian-network structures," The Journal of Machine Learning Research, vol. 2, pp. 445–498, 2002.

[46] H. S. Gunawi, M. Hao, R. O. Suminto, A. Laksono, A. D. Satria, J. Adityatama, and K. J. Eliazar, "Why does the cloud stop computing? lessons from hundreds of service outages," in Proceedings of the Seventh ACM Symposium on Cloud Computing, 2016, pp. 1–16.

[47] J. Bae, T. Helldin, and M. Riveiro, "Understanding indirect causal relationships in node-link graphs," in Computer graphics forum, vol. 36, no. 3. Wiley Online Library, 2017, pp. 411–421.

[48] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for visual understanding of hierarchical system structures," IEEE Transactions on Systems, Man, and Cybernetics, vol. 11, no. 2, pp. 109–125, 1981.

[49] J. Heer, N. Kong, and M. Agrawala, "Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations," in Proceedings of the SIGCHI conference on human factors in computing systems, 2009, pp. 1303–1312.

[50] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," IEEE transactions on visualization and computer graphics, vol. 17, no. 12, pp. 2301–2309, 2011.
[51] A. Sopan, M. Freire, M. Taieb-Maimon, J. Golbeck, B. Shneiderman, and B. Shneiderman, "Exploring distributions: design and evaluation," University of Maryland, Human-Computer Interaction Lab Tech Report HCIL-2010-01, 2010.
[52] F. Heimerl, C. Kralj, T. Moller, and M. Gleicher, "embcomp: Visual interactive comparison of vector embeddings," IEEE Transactions on Visualization and Computer Graphics, 2020.