Adaptive Anomaly Detection in Performance Metric Streams

Olumuyiwa Ibidunmoye, Ali-Reza Rezaie, and Erik Elmroth

***Abstract*—Continuous detection of performance anomalies such as service degradations has become critical in cloud and Internet services due to impact on quality of service and end-user experience. However, the volume and fast changing behavior of metric streams have rendered it a challenging task. Many diagnosis frameworks often rely on thresholding with stationarity or normality assumption, or on complex models requiring extensive offline training. Such techniques are known to be prone to spurious false-alarms in online settings as metric streams undergo rapid contextual changes from known baselines. Hence, we propose two unsupervised incremental techniques following a two-step strategy. First, we estimate an underlying temporal property of the stream via adaptive learning and, then we apply statistically robust control charts to recognize deviations. We evaluated our techniques by replaying over 40 time-series streams from the Yahoo! Webscope S5 datasets as well as four other traces of real Web service QoS and ISP traffic measurements. Our methods achieve high detection accuracy and few false-alarms, and better performance in general compared to an open-source package for time-series anomaly detection.**

***Index Terms*—Performance monitoring and measurement, computer network management, quality of service, time series analysis, anomaly detection, unsupervised learning.**

I. Introduction

PERFORMANCE anomalies in Internet services generally appear as significant deviations (e.g., dips or spikes) in continuous measurements of vital performance metrics or indicators (e.g., response time, CPU utilization, etc) [1] or significant performance regression in between software releases [2] that act as barriers to predictable performance. The occurrence of multiple anomalous data points over a given interval is often an indication of an underlying performance problem and may portend imminent impact on the quality of service (QoS). For instance, an additional one-second delay in page response can increase end-user attrition rate by nearly 11% with thousands of dollars in lost sales [3]. If additional cost and effort required to diagnose and restore services to good state is factored in, then it is clear to see the significant impact on service profitability, reliability, and end-user experience.

According to our previous survey [4], performance anomalies can be concrete manifestations of issues both internal or external to a service, such as, application-level bugs, workload spikes, and correlated systems faults (e.g., server crash or resource contention in cloud datacenters). This multi-dimensional nature thus makes their detection non-trivial if one considers the plurality of software services and the heterogeneous nature of IT infrastructures on which they run.

Anomaly detection at the service level has been mostly addressed from a white- or gray-box perspective where some knowledge of source codes, flow of user requests or transactions, and distributed components are known a priori [5]–[7]. This approach is generally not application or infrastructure agnostic since they often require intrusive source instrumentation [4] and hence hard to generalize for diverse services and use-cases [8], such as in the cloud computing scenario. This has led to renewed interest within the academic and industrial community in performance anomaly detection techniques that are both scalable and versatile [1], [2], [8]–[13]. However, advancement in service deployment, complex service workloads, and dynamism introduced in cloud computing environments, have led to interesting *contextual* behaviour in performance metric data that are hardly explored by existing systems. For example, some measurements that are anomalous at one point in time may be completely normal at another.

Moreover, the problem of anomaly detection is well studied in other domains [14] especially in the field of network monitoring and security where sophisticated techniques have been proposed for varieties of traffic anomalies [15]–[19]. However, in performance diagnosis, many techniques rely on the premise that statistical distribution of metric measurements is either known or static [12], [20], that monitoring data is available a priori for model selection and training [1], and that what constitute normal and/or abnormal behaviour is well delineated [4]. This somewhat limits their applicability in real-time scenarios where they may generate many false-alarms as baselines become obsolete [8]. This is not desirable since the cost of missing an anomaly is often very high. For instance, if a system consistently fails to detect anomalies in service latency, the application may quickly become unusable causing end-users to abandon it. It is also crucial to minimize number of cases where resources are wrongfully deployed to mediate a false

Manuscript received December 21, 2016; revised May 31, 2017; accepted August 24, 2017. Date of publication September 11, 2017; date of current version March 9, 2018. This work is supported by the Swedish Research Council (VR) under contract C0590801 for the Cloud Control project, the Swedish Strategic Research Program eSSENCE, and the EU Seventh Framework Programme under grant agreement 610711 (CACTOS). The associate editor coordinating the review of this paper and approving it for publication was P. Owezarski. *(Corresponding author: Olumuyiwa Ibidunmoye.)*

The authors are with the Computing Science Department, Umeå University, SE-901 87 Umeå, Sweden (email: muyi@cs.umu.se; ali.rezaie@cs.umu.se; elmroth@cs.umu.se).

Digital Object Identifier 10.1109/TNSM.2017.2750906

anomaly. The expectation, therefore, is to balance high detection rates with low false-alarm rates while adapting rapidly to changes in service performance.

The focus of this paper is therefore on the real-time detection of contextual anomalies such as deviation-based outliers or significant changes in dynamic metric streams. We consider a black-box scenario where there is limited knowledge about the nature of the metric stream or internals of the services or systems generating it. We assume that anomalies are indicative of actionable events in the underlying application (e.g., workload phase changes or bugs due to code revisions) or infrastructure (e.g., resource contention) that can be addressed by human operators, a resource controller, or an automated diagnosis and remediation system. The main research question that underscores our goal is as follows:

> "*How to adaptively detect deviations in continuous measurements of a service performance metric with limited a priori knowledge of the metric stream?*"

Addressing this question is useful in many domains such as network operations [16], resource management and orchestration of cloud services [21], integrated monitoring [22], software performance testing [2], [23], and event detection in Internet of Things (IoT) [24]. Our approach seeks to enhance reliable QoS through the following contributions:

1) Online identification of contextual anomalies in streams of a service metric based on a two-step approach:
   a) Adaptive estimation of an underlying statistical and temporal property of an input data stream namely; *conditional density* (Section III-A) and *conditional residual* (Section IV-A) using adaptive learning.
   b) Detecting deviations in sequential estimates of the property using statistically robust and adaptive control charts (Sections IV-B and III-B).
2) Evaluation of the approach in a trace-driven manner by replaying diverse real world and synthetic performance traces as if they were observed in real-time. The traces include data from Yahoo services, two Web service QoS, and two ISP Internet traffic datasets. Results are also compared with an open-source anomaly detection package (Section V).

Results from evaluations show that our approach has high efficacy. In particular, our online techniques offer a factor of 5-6 improvement in detection accuracy compared to a batch deployment of ADVec, an open-source R package for offline time-series anomaly detection. While an offline deployment of ADVec yields lower false-alarm rate of less than 1%, the detection accuracy of our online techniques is at least 12% better.

II. Overview

Consider a service S (e.g., an e-commerce website, a sensor in an Internet of Things environment, or a multimedia streaming service) whose performance is described by a QoS (e.g., latency, throughput, etc.) or resource utilization (e.g., CPU, Memory, etc.) metric M taking values from a continuous temporal process. In essence, monitoring M produces a *metric stream*, an ordered sequence $M_t = m_1, m_2, \ldots, m_t, \ldots$ of real values obtained at equally spaced points in time (e.g., secondly, hourly, or daily) and whose distribution may change overtime. M_t is therefore a time-series that is (a) characterized by successive points arriving continuously, individually or in batches of a given length and (b) whose entire history is not available offline. Note that we consider the specific case where M_t is one-dimensional, that there exist an immediate and continuous temporal relationship between data points at adjacent times (e.g., dependence of m_t on m_{t-1}), and that we may only store a few observations.

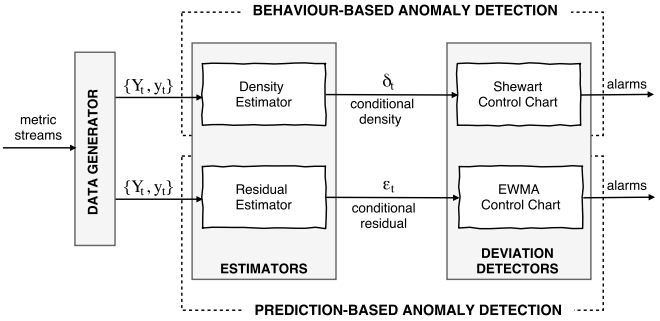


Fig. 1. Overview of adaptive anomaly detection approach.

Typically, M_t can be of varying form or pattern. Examples include long-term stationarity, level or mean shifts, sustained trends (e.g., growing request queue), or periodic fluctuations due to seasonal factors (e.g., time-based work pattern, routine system maintenance or backups). Also, a metric stream may exhibit one or more of these patterns. For example, the CPU utilization of a service that hovers around a constant mean over at idle periods may experience a sudden level-shift at busy periods. Detecting anomalies in M_t is thus concerned with identifying time-points where there is a significant change in the stream. Such points are considered 'unusual' for reasons such as exhibiting lack of continuity with immediate observations or deviation from the stream's short- or long-term behaviour [25]. In general, having a precognition of the form of M_t can provide clues on assumptions and analysis approach [8]. However, since it is non-trivial to estimate this accurately in online settings, methods that are somewhat agnostic to temporal patterns of the stream are desirable.

Therefore, we propose two similar but independent techniques for detecting contextual deviations in streams M_t of metric M namely; Behaviour-based Anomaly Detection (BAD) and Prediction-based Anomaly Detection (PAD) as shown in Fig. 1. The learning methodology is by *incrementally learning* from most recent behaviour of a given stream. The motivation for incremental learning is the need for computationally inexpensive techniques that evolve with the data with limited parameter settings or offline retraining. Both approaches are similar in that they share the same underlying detection methodology based on adaptive estimation of a *statistic*–a property that tracks either temporal or statistical behaviour–of a given metric stream. We hypothesize that by tracking such statistic we might be able to automatically detect changes in the performance of service S in terms of metric M online. The design goal is to achieve high detection

rate without sacrificing low false-alarms in order to reduce the risk of missing anomalies or wrongfully deploying resources to address a false-alarm.

Fig. 1 is an overview of our proposed approaches and below we define the 3 main categories of components:

- *Data Generator:* The function of this component is to ingest data into the system. Most especially to partition metric streams over a sequence of temporal windows via two windowing strategies describing the data horizon and obsolescence as shown in Fig 2. The source of the data streams may be from a monitoring agent or from a time-series database such as InfluxDB[1] or OpenTSDB.[2]
- *Estimators:* We propose two types of estimators, each estimating a different type of statistic for every pair of input $\{Y_t, y_t\}$, where Y_t is a time-series of observations in the most recent window and y_t is an observation whose statistic is to be estimated.
- *Deviation Detectors:* We apply two types of adaptive control charts to discover deviations in sequential estimates of statistics of interest.

Behaviour-based Anomaly Detection (BAD) involves learning and detecting changes in underlying metric behaviour as described below:

(a) *Tumbling Windows Learning:* This partitions metric streams into a set of k-length windows $W' = \{W'_j : j \geq 1\}$ that are similar to sliding windows but with a sliding step equals to k the window size, and observations arriving sequentially. Let Y_t be the set of observations in a given window W'_j, then Y_t contains observations over the interval $(k \times (j-1)+1,\ j \times k)$. The window size indicates the time horizon such that at any time t within a given window W'_j, there exist observations from a preceding window W'_{j-1} in storage.

(b) *Density Estimation:* The estimator tracks the underlying statistical behaviour–unknown probability density function (PDF)–of a given Y_t using adaptive kernel density estimation [25]. Note that Y_t is a sequence of observations in the last tumbling window. The PDF obtained is then used to estimate statistic δ_t, the *conditional density* of y_t based only on observations in Y_t.

(c) *Deviation Detection:* Here we derive an adaptive threshold based on the lower side of a Shewharts control chart [26]. The mean and standard deviation parameters of the control chart are obtained from conditional densities in the most recent tumbling window. Out of control observations are considered anomalies.

Prediction-based Anomaly Detection (PAD) operates by tracking continuity in temporal behaviour of a stream and uncover anomalies as abrupt discontinuity as follows:

(a) *Sliding Windows Learning:* This involves a sequence of k-length windows $W = \{W_t : \forall\ t > k\}$ that advance in one step at a time. Let Y_t represents observations over the time horizon $(t-k,\ t-1)$ at time t. Hence, Y_t is a time-series of the most recent k observations up to time t with observations that expire one at a time.

[1]https://www.influxdata.com/time-series-platform/influxdb/

[2]http://opentsdb.net/

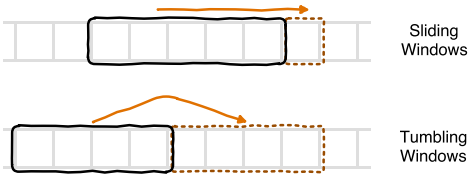


Fig. 2. Window strategies.

(b) *Residual Estimation:* The estimator first makes a forecast $\hat{y}_t$ using adaptive cubic spline model built from observations in Y_t. The forecast, $\hat{y}_t$, is the expected value of y_t at time t based only on Y_t, that is, observations in the most recent sliding window. Based on this forecast we compute the *conditional residual*, $\varepsilon_t = (y_t - \hat{y}_t)$, a statistic which tells how much the observed value deviates from expectation.

(c) *Deviation Detection:* Using the same sliding windows as the residual estimator, an Exponentially Weighted Moving Average (EWMA) control chart [26] is constructed to track the conditional residuals. The chart is characterized by two adaptive bounds such that observations whose residuals fall outside these bounds are classified as anomalous.

Consequently, the output of the techniques are actionable alarm signals that can be used to alert system engineers, activate an automatic performance diagnosis system, or automatically adjust system resources in response in an autonomous environment. For the rest of the paper, we will talk mainly in terms of the approaches rather than the sub-components. Details of the techniques are presented in Sections III and IV respectively, including the formulation of the estimators and detectors.

III. Behaviour-Based Anomaly Detection

The behaviour-based anomaly detection (BAD) technique exploits a behaviour, such as the unknown probability distribution of a given metric stream in order to identify anomalies. It relies on periodic learning by using the behaviour of a preceding tumbling window to classify observations in the following. In this section, we describe this process in Section III-A and how to identify anomalies in Section III-B.

A. Adaptive Density Estimator

Here we address the issue of how to estimate the conditional density of a given observation y_t based only on a window of observations Y_t. We apply kernel density estimation (KDE) [25], a major statistical learning technique, to realize this similar to the approach in [27]. KDE relies on the assumption that there are more normal data points in a given data than anomalous and that normal data points lie in dense localities (high density) and anomalies lie in sparse localities (low density). However, the technique is known to produce spurious density estimates due to the use of equal-sized kernels without accounting for variations from one data point to another [28]. In addition, KDE is an instance of generalized n-body problems known to have quadratic complexity.

Therefore, we develop an adaptive density estimator based on KDE. This technique offers two advantages of over the basic KDE: (a) it improves accuracy of estimation by accounting for local variations, and (b) improves time complexity via a tree-based implementation. Let $Y_t = \{y_j\} : j = 1, 2, \ldots$ be the set of observations in a given window, the conditional density of each observation y_j is derived as follows [28]:

$$\hat{f}_h(y_j|Y_t) = \frac{1}{|Y_t|} \sum_{y_i \in Y_t, i \neq j} \frac{1}{h\lambda_i} K_h\left(\frac{y_j - y_i}{h\lambda_i}\right) \quad (1)$$

Eq. (1) estimates a continuous density of each observation conditioned on the rest of the data by aggregating a set of scaled Gaussian kernel functions, $K_h(\cdot) = \frac{1}{h\sqrt{2\pi}} \exp(-\frac{(\cdot)^2}{2h^2})$, centered at each data point. The width of each kernel or rate of smoothing, is controlled by the *global* and *local* bandwidth parameters h and $h\lambda_i$ respectively. While h is fixed and controls the variance across the entire data, $h\lambda_i$, adapts the amount of smoothing based on the local density at each time. The latter is estimated in two steps. It first approximates a pilot density profile using only global smoothing. The pilot density profile is then used to compute parameter λ_i using the square root rule and the procedure described in [28].

A high value of h leads to very smooth estimates but may ignore local variance while low values produce less smooth but potentially more accurate estimates [29]. Though h can be set to standard approximations for Gaussian kernels, we derive h via grid-search cross-validation. Following the propositions in [30], the density estimation algorithm is implemented using kd-trees, which reduces the run-time complexity to $O(n \log n)$, where n is the number of data points.

The estimator operates over the entire metric stream via periodic batch-incremental learning [31] using consecutive tumbling windows (Fig. 2) to estimate conditional densities of the data. Let W_{j-1} and W_j be two consecutive windows. Also, let Y_t be the sequence of observations in W_{j-1} and Y'_t be the sequence of observations in W_j. We construct a baseline density profile $\theta_{j-1} = \hat{f}_h(y_t \in Y_t|Y_t)$ according to Eq. (1). The conditional density of a given observation $y_0 \in Y'_t$ is given as $\delta_0 = \hat{f}_h(y_0|Y_t)$. The conditional density δ_0 can be also be interpreted as the likelihood of Y_t taking on value y_0 given its distribution, θ_{j-1}. In order to reduce the influence of anomalies when obtaining the baseline density profile, only normal observations from the training window are used.

The density estimation and window-based approach described above does not explicitly exploit temporality of the stream and so may result in poor precision on streams with strong seasonal and trend patterns. Specifically, it may generate many false positives on such streams as the trend slope changes from window to window or if the window does not properly align with seasonal peaks and troughs in the stream. Hence, the method will be bias towards stationary streams whose statistical property such as the mean do not depend on time. One way to reduce this effect is to perform an optional step that filters each window's stream Y_t via first order differencing, also known as *detrending* [32]. By computing the change (i.e., $y'_t = y_t - y_{t-1}$) between consecutive observations, the mean can be stabilized and the trend and seasonality of the stream are thereby minimized. It thus become easier to obtain unbiased density estimates using the differenced series. In Section V we show how this step, which will be referred to as BAD*, improves the precision of the density-based approach.

Finally, a problem arises of how to determine the optimal number of observations to accumulate in a window. An effect of using a fixed value for all streams be that the model may overfit recent behaviour should this value be too small that it cannot generalize. Conversely, the model may become less sensitive to recent changes should the window size be too large. Many studies, such as in [31] and [33] have shown that different streams will generally require different window sizes, and so advised that the optimal window size be selected in a data-driven manner for different group of streams. In Section V, we describe one such strategy using the case study dataset.

B. Deviation Detector: Shewhart Control Chart

Here, we will describe how to determine if an observation y_0 is an anomaly or not based on its conditional density estimate δ_0. A simple way to detect deviations is to define a suitably low threshold, ϵ, such that y_0 is anomalous if $\delta_0 < \epsilon$ (e.g., $\epsilon = 0.001$). However, since the static threshold defines deviations in terms of observations at a specific time, it is not suitable for tracking aggregate change in the underlying stream. Also, different threshold values affect detection rates differently. Instead, we consider y_0 anomalous if $\delta_0 < (\mu_{j-1} - \mathcal{L}\sigma_{j-1})$, based on the Lower Control Limit (LCL) of an Individual Shewhart control chart [26]. The mean μ_{j-1} and σ_{j-1} are the mean and standard deviation of the conditional densities obtained using observations from window W_{j-1}. The parameter $\mathcal{L}$ is typically set according to the tradeoff between sensitivity to anomalies and false alarm rate. Generally $\mathcal{L} \geq 3$ is recommended [10], [26], since the $(\mu \pm 3\sigma)$ region of a normal distribution contains about 99.7% of data. However, since the distribution of the conditional density profile may not necessarily be Gaussian, we select $\mathcal{L}$ adaptively as the absolute 95^{th} percentile z-score of items in the baseline density profile θ_{j-1}. Let δ_t be an instance of θ_{j-1} at time t, the corresponding absolute z-score is computed as $z_t = \frac{|\delta_t - \mu_{j-1}|}{\sigma_{j-1}}$, a measure of how far δ_t is from the mean in terms of the standard deviation. Defining parameter $\mathcal{L}$ absolve us of the need to define a fixed value and so the method can to some extent adapt with the data.

IV. Prediction-Based Anomaly Detection

The approach in Section III is based on training a model periodically to recognize new independent data that do not belong to the distribution of typical or normal data. In some settings, it maybe difficult to accurately define the tumbling windows. Also, it does not take into account the temporal dependence between observations and may not capture important trends or repetitive behaviour. Prediction-based anomaly detection (PAD) exploits these properties. The assumption is that if the temporal relationship is continuous, we can exploit properties of the stochastic component, the residual, to detect deviations (e.g., spikes or level shift) in the metric stream.

For example, the residuals of a normal metric stream should exhibit consistent variation from one time step to another; a sudden or consistent shift of a certain magnitude may suggest an anomaly [32].

Given a series Y_t containing observations from the most recent sliding window, the conditional residual is obtained as the difference between the expected value of Y_t and actual observation at specific time (typically one-step ahead). In the first part of this section, we introduce an adaptive splines-based estimator to achieve this. Afterwards, we describe the process of detecting deviations in streams of conditional residuals using adaptive EWMA control chart.

A. Adaptive Residual Estimator

Metric streams often contain pronounced seasonal pattern due to seasonal factors (e.g., hour of the day or day of the week) or non-linear shapes depending on the particular metric. Such patterns may be hard to track by basic forecasting techniques, such as exponential smoothing, especially when they occur abruptly within a stream. Also, exponential smoothing techniques such as the ETS [32] are known to require significant historical data with low noise level [34]. Another weakness of the ETS is the need to manually define up to three smoothing parameters. While these parameters can be estimated on the fly via exhaustive grid search, the high computational requirement of such 3D search is a limitation of this approach in online settings.

There exist alternative prediction methods that can be employed. On one hand is the autoregressive integrated moving average (ARIMA) and its seasonal variant (SARIMA) commonly used for time-series forecasting [32]. However, they generally require access to the entire dataset in advance to assess autocorrelation and partial autocorrelation properties to aid parameterization. Also, they do not cope well with missing data and rely heavily on stationarity assumption [35]. On the other hand are techniques based on artificial neural networks (ANN) [36]. However, ANN techniques are most suited for offline cases due to the long trial-and-error training to determine optimal structure of the network and generally slower than splines. These limitations make ARIMA and ANNs unsuitable for our case since we have to deal with diverse metric behaviour autonomously.

Hence, we present an adaptive residual estimator that fits a sequence of adaptive cubic spline models over the stream as it evolves. There are many benefits to the choice of splines. Splines are known to be flexible, computationally fast [34], easy to automate and to handle missing values, and can easily pick up complex non-linear shapes [35]. They can also be trained incrementally on local data, thereby honouring temporal continuity in the stream. Coefficients of a spline model can also be easily estimated via ordinary least squares (OLS).

Let Y_t be the series of observations in the most recent sliding window. We can describe Y_t as the sum of the deterministic component X_t, representing trend and seasonal components of the stream, and a stochastic component Z_t, as follows:

$$Y_t = X_t + Z_t, \tag{2}$$

The stochastic component Z_t is stationary and can be described by an autoregressive moving average (ARMA) model. The deterministic component X_t cannot be so described but can be captured using a non-linear spline. To achieve this, let X_t be a continuous function, $x(t)$, that can be described by a linear combination of r known basis functions $\alpha_1(t), \alpha_2(t), \ldots, \alpha_r(t)$ over t as follows:

$$x(t) = \sum_{i=1}^{r} c_i \alpha_i(t) \tag{3}$$

The coefficients, $\boldsymbol{c} = c_1, c_2, \ldots, c_r$, are selected by minimizing the sum of squared errors (SSE); $SSE(\boldsymbol{c}) = \sum_{t=1}^{|Y_t|} (Y_t - x(t))^2$, using OLS.

Furthermore, let the repetitive seasonal component of Y_t be P_t and its length or periodicity be s. Depending on the dataset, P_t may exhibit changing characteristics (amplitude, level and shapes) with time. Thus, we need a sequence of at least d periodicity to properly estimate the pattern to cope with changing time dynamics. We use $d = 2$ to capture short-term changes since $d < 2$ will be too sensitive to temporary fluctuations in the seasonal pattern while $d > 2$ will be too slow to adapt to any dynamic shifts in the seasonal pattern [35]. Hence, the length of the sliding-window W_t at time t is then computed as $k = d \times s$, covering observations from time $t - k + 1$ up to t.

To predict the one-step ahead seasonal pattern, p_{t+1}, at time t, a cubic spline $x_t(z), z \in [0, s]$, is fitted by OLS to Y_t consisting of the $d = 2$ overlaying sequences of periodicity s according to Eq. (3). This yields a spline estimate $\hat{x}_t(z) \in [0, s]$ which contains expected values of the stream for the next k time steps. Then, the seasonal component at $t+1$ is predicted as $\hat{p}_{t+1} = \hat{x}_t(1)$. The forecast error, or residual, $\varepsilon_{t+1} = y_{t+1} - \hat{p}_{t+1}$ is how much the observed value, y_{t+1}, deviates from the predicted pattern. The residual ε_{t+1} is expected to have a positive auto-correlated structure that also brings information about future values and can thus be described by a simple lag-2 auto-regressive model (AR(2)) of the form $\varepsilon_{t+1} = \phi_0 + \phi_1 \varepsilon_t + \phi_2 \varepsilon_{t-1} + e_t$, where e_t is white noise.

To decide if observation y_{t+1} is anomalous or not we evaluate ε_{t+1} using an EWMA control chart that is described in the next section. Note that the estimation of the repetitive pattern is based on the accuracy of past estimations, which means that care should be taken to reduce the influence of anomalies or missing values. A simple backtracking mechanism is employed to address this. Since we can already determine if an observation is anomalous or not, we update anomalies with the expected value at that point. We achieve this by taking advantage of the expected auto-correlated behaviour of the residuals. So, if the residual ε_{t+1} is anomalous, to keep the seasonal pattern, the observation y_{t+1}, will be replaced by the predicted value, $\hat{y}_{t+1}$ obtained as $\hat{y}_{t+1} = \hat{p}_{t+1} + \hat{\varepsilon}_{t+1}$, where $\hat{\varepsilon}_{t+1}$ is estimated by fitting an AR(2) model to the most recent sequence of residuals:

$$\hat{\varepsilon}_{t+1} = \hat{\phi}_0 + \hat{\phi}_1 \varepsilon_t + \hat{\phi}_2 \varepsilon_{t-1} \tag{4}$$

where the parameters $\hat{\phi}_0$, $\hat{\phi}_1$, $\hat{\phi}_2$ are estimated by OLS. Note that $\hat{p}_{t+1}$ and $\hat{\varepsilon}_{t+1}$ corresponds to X_t and Z_t in Eq. (2) respectively. This strategy allows the residuals to keep their time

dependency from the normal pattern when estimating future residual values without being affected by anomalies.

In general, the estimator uses a sliding window based batch-incremental learning [31]. To estimate the conditional residuals of subsequent observations, the estimator shifts the window one time step and the method above is repeated in an *interleaved-test-then-train* cycle. This enables the model to capture both short-term behaviour and emerging trend in the stream. Since only recent data are used to train the model, it is able to adapt quickly to the changing time dynamics of the stream.

B. Deviation Detector: EWMA Control Chart

Having introduced the prediction-based estimator, here in this section, we present a robust technique for discovering deviations in streams of conditional residuals. EWMA control charts among others such as Shewhart's $\bar{x}$ and R are process control mechanisms for monitoring variability and achieving stability in production processes [26]. A control chart shows sequential observations of a given process characteristic, Y_t over time; with, a center line (CL) indicating the mean of Y_t under normal conditions, bounded above and below by lines indicating the upper (UCL) and lower (LCL) control limits respectively. While most control charts assume that observations follow a Gaussian distribution, EWMA charts are robust against this assumption and particularly suited for time-series data [26].

We design an adaptive EWMA chart that is tightly coupled to the prediction-based estimator in order to close the detection loop. The chart is adaptive because its bounds are derived based on streams of conditional residuals $\{\varepsilon_t \in \mathbb{R} : \forall\ t > 0\}$ arriving sequentially from the estimator. The EWMA control chart is constructed as

$$\hat{\varepsilon}_t = \lambda\varepsilon_t + (1-\lambda)\hat{\varepsilon}_{t-1} \tag{5}$$

by exponentially smoothing the forecast residuals over time at a rate controlled by parameter λ, where $0 < \lambda \leq 1$. The center line and control limits for the EWMA control chart are then defined as follows [26]:

$$(UCL_t, LCL_t) = \mu_t \pm \mathcal{L}\sigma_t\sqrt{\frac{\lambda}{(2-\lambda)}\big[1-(1-\lambda)^{2t}\big]} \tag{6}$$

where $\mathcal{L}$ is a multiple of the standard deviation controlling how sensitive the chart is to shifts around the center line. Again, since $\mathcal{L}$ serves the same purpose as in Section III-B, we set it to the 95^{th} percentile of the z-score distribution of the most recent k observations. Depending on the desired level of sensitivity, the percentile threshold can be set higher (e.g., 99^{th}). The arithmetic mean, μ_t, and standard deviation, σ_t, are estimated incrementally according to the same sliding window as the predictor. Parameter, σ_t, is derived based on the adjusted variance, $\sigma_{t'}^2 = \sigma_t^2\frac{\lambda}{2-\lambda}[1-(1-\lambda)^{2t}]$, where σ_t^2 is the true variance of the most recent k observations. Note that as t grows, the term $[1-(1-\lambda)^{2t}]$ approaches unity and the control limits converge to steady-state limits $(UCL_t, LCL_t) = \mu_t \pm \mathcal{L}\sigma_t\sqrt{\frac{\lambda}{(2-\lambda)}}$ [26].

The smoothing parameter λ is chosen based on the size of the sliding windows using the technique in [15]. It is based on a basic property of Eq. (5). Each $\hat{\varepsilon}_t$ is a weighted average of all previous samples with geometrically decreasing weights, $\lambda(1-\lambda)^t, t \geq 0$ that all sum to unity. However, the sum of the most recent, k observations is obtained using $1-(1-\lambda)^k$. Using these facts, the value of λ is computed based on the speed at which older observations should be dampened or forgotten. For short-term (fast) smoothing, $\lambda = e^{(\log(1-\mathcal{P})/k)}$, and $\lambda = 1-e^{(\log(1-\mathcal{P})/k)}$ for long-term (slow) smoothing, where $\mathcal{P}$ is the percentage of all weights that the most recent k observations should account for. Since we are interested in capturing rapid changes in the stream, we use the fast smoothing so that the most recent k observations take 95% of the weights. Hence, λ increases or decreases according to the size of the sliding window. This eliminates the need for arbitrary smoothing values that are not only hard to define but also alter smoothing accuracy differently.

The EWMA chart is used to detect deviations according to the following rule. An observation y_t is classified as anomalous if its smoothed residual $\hat{\varepsilon}_t > UCL_t$ or $\hat{\varepsilon}_t < LCL_t$ at time t. That is, anomalies are those out-of-control observations whose smoothed residuals violate the control limits of the adaptive control chart at any given time.

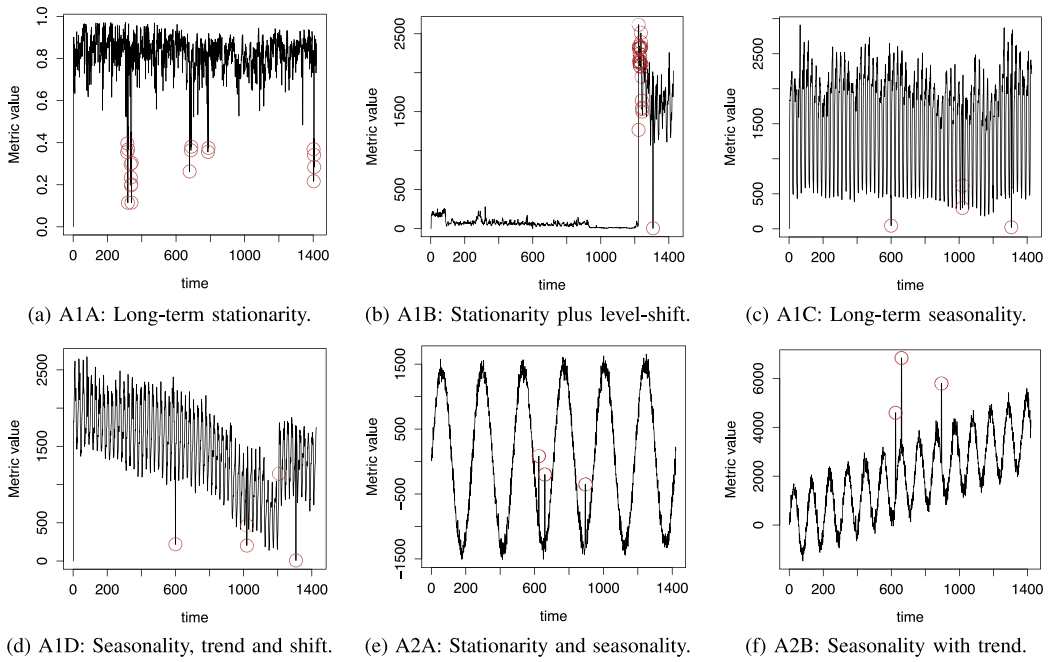
V. Evaluation and Results

In this section we evaluate the performance of the proposed behaviour-based and prediction-based anomaly detection techniques in a trace-driven manner by analyzing multiple real-world datasets. The experiments are performed by replaying diverse time-series streams from the Yahoo! Webscope S5 anomaly detection datasets, as if they were collected in real-time. Furthermore, we compare the performance of our approach with ADVec, a module in Twitter's open-source R package for time-series anomaly detection [37]. ADVec is an offline technique that employs Generalized Extreme Studentize Deviation to robustly detect anomalies in the presence of seasonality and an underlying trend. For fair comparison, we deployed ADVec so that it detects anomalies in batches by feeding it sequential windows of a given stream. We also report its performance when deployed offline (ADVec*), that is, an entire input time-series is fed into the module at once. Using the Yahoo traces as a baseline, we also test the generality of our techniques using two other traces of ISP traffic and Web service QoS measurements.

A. Evaluation Datasets

A major challenge in performance anomaly detection is access to open, reliable, annotated dataset of the performance of computing systems and services [4]. Though empirical data can be obtained from experimental testbeds, it is non-trivial finding the right mix of applications and workloads to induce interesting behaviour close to production.

1) Yahoo! Webscope Dataset: The Yahoo! S5 computing systems data [38] was released as part of the Webscope Program to address these challenges. It is a collection of both real and synthetic time-series datasets. The real datasets (A1Benchmark) are collected from some of Yahoo Web services which means that the accompanying anomalies have



(a) A1A: Long-term stationarity. (b) A1B: Stationarity plus level-shift. (c) A1C: Long-term seasonality.

(d) A1D: Seasonality, trend and shift. (e) A2A: Stationarity and seasonality. (f) A2B: Seasonality with trend.

Fig. 3. Characteristic behaviour of time-series in the Yahoo! datasets. Anomalous data points are indicated in red.

been observed in production and marked by human operators while the synthetic datasets (A2Benchmark) have been generated by an automated tool based on realistic distributions. The datasets are also labeled using a boolean field to indicate normal or anomaly observations.

From initial exploratory analysis, we observed some striking commonalities and differences across the time-series which motivated us to categorize them into six groups. Fig. 3 describes the characteristic shape of streams in each group. Those with A1 prefix are subset of A1Benchmark while those with A2 prefix belong to A2Benchmark. We randomly chose a subset of each groups that fulfill some criteria. First, our window based techniques require at least the first window of the time-series to be anomaly free, that they contain both positive (spikes) and negative (dips) anomalies, contain at least two anomalies, and bear representative shapes of their respective groups. This results in 10 time-series streams from A1A, 7 from A1B, A1C and A2A, and 8 from A1D and A2B respectively. In the end, we have a total of 47 time-series streams on which we evaluate both BAD, PAD as well as ADVec. The distribution of anomalies contained by each group is shown in the violin-plots of Fig. 4. Time-series streams in A1A vary the most in the number of anomalies they contain followed by A1D. This shows that A1Benchmark is much more diverse than A2Benchmark in terms of shapes and the distribution of anomalies.

Moreover, we observed a significant imbalance between the number of anomalies and normal observations within and across the two categories. For example, on average, anomalies make up only 2% of observations in the real datasets and a meagre 0.35% of the synthetic datasets. To adjust for this disparity, we developed a tool to inject additional anomalies into streams in A2Benchmark based on statistical characteristics of existing anomalies, so that anomalies account for

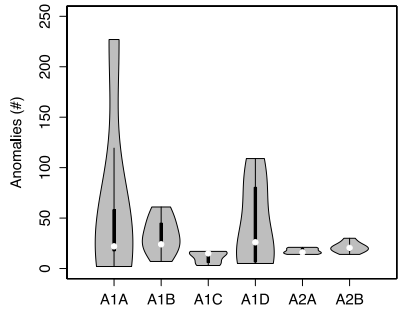


Fig. 4. Distribution of anomalies across datasets.

2% of the observations in both categories. A periodogram was used to automatically estimate the length of dominant periods (i.e., periodicity) in the seasonal streams. We found the mean periodicities to be 24 and 240 for A1Benchmark and A2Benchmark respectively, and the average stream length being about 1400 observations.

2) Web Service QoS Dataset: We also use employ response time data of two real Web services provided by Cavallo *et al.* [39] (a) *Amazon*[3]: a Web service which searches for books in the online book store, and (b) *BLiquidity*[4]: A financial Web service providing information on liquidity in a banking system. The data were collected by invoking the services every hour for about four months and recording the observed response time.

3) ISP Internet Traffic Dataset[5]*:* Similarly to studies in network traffic anomaly detection, we also analyze two Internet ISP traffic provided by [36]

[3]http://www.xmlme.com/WSAmazonBox.asmx
[4]http://webservices.lb.lt/Bliquidity/Bliquidity.asmx
[5]http://data.is/TSDLdemo

(a) *U.K. traffic*: Aggregated Internet traffic in the United Kingdom academic network backbone, collected hourly by an ISP between November 2004 and January 2005, (b) *EU traffic*: Internet traffic data from a private ISP with centers in 11 European cities collected hourly from a transatlantic link over 2 months (June - July, 2005).

Using similar technique as for the synthetic traces of the Yahoo datasets, we randomly injected synthetic anomalies in both the QoS and ISP traffic datasets based on inherent statistical properties of the traces as shown in Fig 5.

B. Evaluation Metrics

The performance of proposed techniques are described in terms of detection rate (DR), precision (PR), false-alarm rate (FR), and the F-measure in percentages. The metrics are derived based on standard evaluation outcomes namely, True Positives (TP), False Negatives (FN), True Negatives (TN), and False Positives (FP) where anomalous observations belong to the `POSITIVE` class while normal observations belong to the `NEGATIVE` class. Hence $DR = \frac{TP}{TP+FN}$, the proportion of anomalies that are correctly identified; $PR = \frac{TP}{TP+FP}$, the proportion of all detections that are truly anomalies; $FR = \frac{FP}{FP+TN}$, proportion of normal observations wrongly classified as anomalies; and $F_\beta = \frac{(\beta^2+1)(PR\times DR)}{(\beta^2\times PR+DR)}$, the trade-off between precision and detection rate. The F-measure is good for comparing the quality of two techniques, where $F_\beta \simeq 100\%$ indicates good balance between precision and recall while $F_\beta \simeq 0$ indicates poor balance. We set $\beta = 2$ to give priority to recall to have the F_2-score. The Mathew's Correlation Coefficient, $MCC = \frac{TP\times TN - FP\times FN}{\sqrt{(TP+FN)(TN+FP)(TP+FP)(TN+FN)}}$, is also computed. The MCC is the degree of agreement between true classes of observations (*ground truth*) and the assigned classes by the methods, where $MCC \simeq +1$ indicates perfect classification, $MCC \simeq 0$ means no better than random classification, and $MCC \simeq -1$ indicates total disagreement.

Also, since the performance of PAD is dependent on its forecast accuracy, we also evaluate this in terms of widely used forecast accuracy measures [32], namely, (a) Mean Absolute Percentage Error (MAPE), (b) Weighted Absolute Percentage Error (WAPE) and (c) Mean Absolute Error (MAE).

In the following sections, first we describe how we determine optimal window sizes, describe the performance of the proposed techniques by answering a number of relevant questions.

C. Determining Optimal Window Sizes

The window hyper-parameter k plays a major role in the operation and performance of both BAD and PAD because it signifies the time horizon over which variations go from being ordinary noise to an indication of true temporal changes in the stream. Instead of using arbitrary window sizes, we select the value of k in a supervised manner based on the Receiver Operating Characteristic (ROC) curve which plots the recall (*DR*) against the false-alarm rate (*FR*) [31], [33]. A window size k is considered to perform well if it is closest to the upper left corner of the ROC plot, which means it has very high *DR* and very low *FR*. We can therefore view the selection of k as an optimization problem of minimizing the Euclidean distance between the upper left corner (coordinate (100%, 0%)) and the (*DR*, *FR*) coordinate of a series of window sizes $k_1, k_2, \ldots, k_n$. The optimal k for each dataset group is expressed as follows:

$$k = \underset{k}{\operatorname{argmin}} \sqrt{(FR-0)^2 + (DR-100)^2}$$

To obtain this value, we randomly select a stream from the those that have not been selected for testing in each group, vary the value of k from 30 to 240 observations in step of 30, and run both the BAD and PAD techniques independently to obtain the FR and DR quantities. Table I highlights the optimal k along with the respective distance ($\mathcal{D}$) for the proposed techniques. Notice that PAD has undefined entries for the seasonal datasets because according to Section IV-A, k is determined based on the length of a seasonal period. The difference in value of optimal k reaffirms existing understanding that window sizes will differ for different streams [31]. Hence, Table I provides justification for our choices of k in subsequent analysis of each stream type.

TABLE I
OPTIMAL WINDOW SIZES

	A1A		A1B		A1C		A1D		A2A		A2B	
	k	$\mathcal{D}$	k	$\mathcal{D}$	k	$\mathcal{D}$	k	$\mathcal{D}$	k	$\mathcal{D}$	k	$\mathcal{D}$
BAD	**120**	13.68	**180**	1.88	**150**	0.31	**240**	2.46	**240**	0.26	**240**	0.94
PAD	**120**	1.62	**150**	5.8	-	-	-	-	-	-	-	-

TABLE II
FORECAST PERFORMANCE OF PAD: 75%ILE ERROR RATES

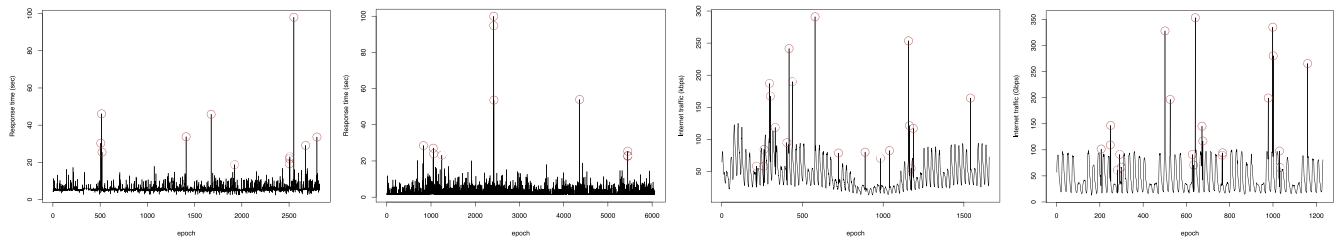
	A1A	A1B	A1C	A1D	A2A	A2B
MAPE (%)	∞	∞	0.11	0.46	0.47	0.06
WAPE (%)	0.46	1.04	0.11	0.46	1.41	0.06
MAE	24.41	454.28	28.43	7873.72	182.93	705.97

D. Performance Evaluation

Testing proceeds by replaying observations in a given time-series sequentially as they would arrive from an online monitoring tool or time-series database and served into each technique as illustrated in Fig. 1. We compute performance metrics described in Section V-B and aggregated using the *median* since it is more robust to outliers than the mean. We also report the median absolute deviation (MAD) which is the median's equivalence for the standard deviation.

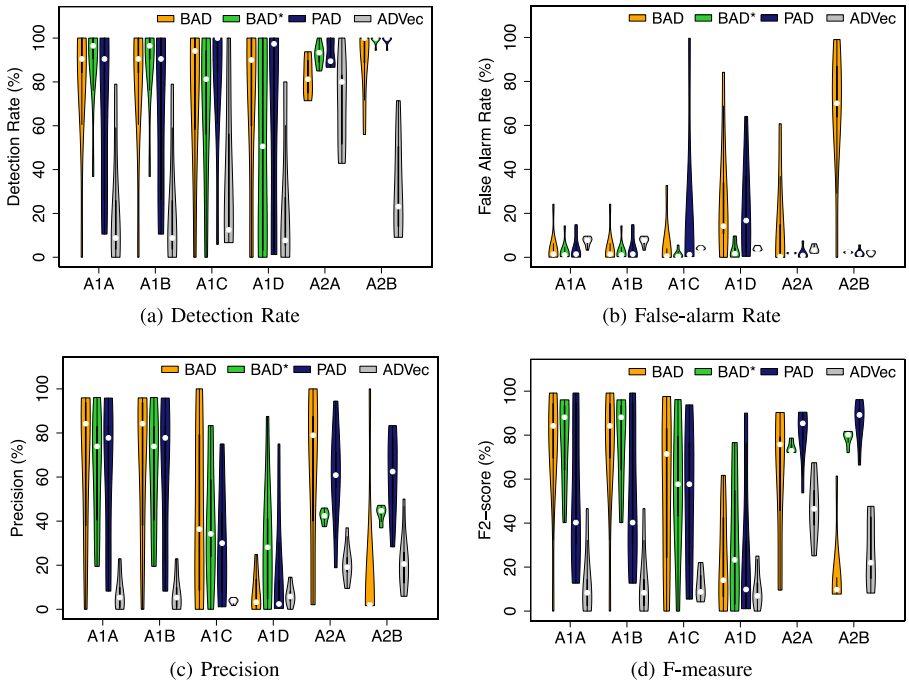
To clear doubts about the accuracy of the prediction-based method PAD, we present its forecast error rates obtained on a random stream from each group in Table II. It is obvious that the accuracy is quite high suggesting that results of anomaly detection is not dominated by poor prediction. MAPE is undefined in some cases due to dividing point errors by zero-valued observations. The WAPE corrects for this by dividing by the mean of observations in such cases.

In the next section, results shown in Fig. 6 and Table III are used to address specific research questions. Note that `BAD*` represents the case of applying the BAD technique on detrended streams as described in Section III-A.



(a) Amazon web service (b) Financial web service (c) UK Internet traffic (d) EU Internet traffic

Fig. 5. Time series plots of four different datasets with anomalies annotated in red. Plots (a) and (b) are hourly QoS measurements two real Web services over a four month period while (c) and (d) shows the hourly volume of Internet traffic within the U.K. and the EU collected over two months by an ISP.



(a) Detection Rate (b) False-alarm Rate

(c) Precision (d) F-measure

Fig. 6. Performance distribution of proposed methods. The white dots on the bars indicates the median.

1) How Effective Are BAD and PAD in Detecting Anomalies Across the Datasets? How Do They Compare With ADVec?: Fig. 6 shows the performance of BAD, BAD*, and PAD compared to ADVec using grouped Violin plots. Each bar is a sideways plot of the distribution of each metric across time-series per group. The median is indicated in white dot while the thick dark spine represents the interquartile range. Also, Table III aggregates performance across seasonal (A1C, A1D, A2A, and A2B), non-seasonal (A1A and A1B) and combined streams.

Notice that none of the techniques can be said to be consistently superior across all dataset groups. While BAD achieves good accuracy of at least 80% median detection rate across all groups, it generates too many false-alarms on the seasonal datasets especially on A2B with equally poor F_2 scores. Though it has a near-zero median false-alarm rate on A2A, we can observe that it produces non-zero false-alarms on at least 50% of the streams in that group. This explains the variation in its F_2 score on the same group.

PAD achieves comparable performance with BAD on A1A and A1B and superior performance on both A2A and A2B with a median detection rate of at least 90% and negligible false-alarms with competitive F_2 scores. However, PAD's detection rate, false-alarm rate and F_2 scores are just as poor as BAD's on A1C and A1D. Across the entire data, PAD shows good balance between precision and recall with a FPR that is 77% than BAD's. Surprisingly, BAD* offers the best performance across board especially in terms of false-alarm rates. The detection performance is clearly comparable to BAD on nonseasonal datasets and to PAD on the seasonal ones. However, similar to BAD and PAD, it has poor precision on A1C and A1D.

ADVec shows poor performance in general compared to our techniques. Its low false-alarm rates is made unattractive due to its poor precision (Fig. 6c) and accuracy (Fig. 6a) across the all streams especially on those with level-shifts. In contrast, our techniques offer high detection accuracy in general with comparably low false-alarm rates. Though the offline method ADVec* is generally better than its batch-oriented variant ADVec, its detection rate is barely at par with the proposed techniques. Using the combined performance in Table III, it is easy to see that our online techniques offer a factor of 5-6

TABLE III
AGGREGATE PERFORMANCE EXPRESSED IN MEDIAN (MAD)

	Non-Seasonal			Seasonal			Combined		
	DR(%)	**FR(%)**	**F_2(%)**	**DR(%)**	**FR(%)**	**F_2(%)**	**DR(%)**	**FR(%)**	**F_2(%)**
BAD	93.16 (7)	5.11 (5)	64.48 (28)	93.31 (7)	13.96 (14)	16.89 (17)	93.31 (7)	8.94 (9)	40.12 (31)
BAD*	98.24 (2)	1.77 (1)	69.23 (19)	94.09 (6)	1.95 (1)	72.41 (8)	94.72 (5)	1.91 (1)	72.41 (15)
PAD	97.91 (2)	1.54 (1)	58.67 (28)	100 (0)	1.27 (1)	78.53 (13)	100 (0)	1.40 (1)	72.64 (20)
ADVec	4.35 (4)	7.63 (2)	2.48 (2)	33.03 (24)	3.42 (0)	17.55 (10)	15 (15)	3.94 (1)	14.08 (11)
ADVec*	93.75 (6)	1.73 (2)	68.25 (15)	75.73 (24)	0.45 (1)	69 (19)	82.98 (17)	0.57 (1)	68.25 (19)

improvement in detection accuracy compared to ADVec, with both BAD* and PAD yielding about 50% fewer false-alarms. Surprisingly, though the false-alarm rates of the offline variant ADVec* is negligible at 1%, its detection rate is 12-20% lower than our online techniques.

2) Are the Observed Performance Differences Between the Techniques Statistically Significant Across the Dataset?: Given the results in Fig. 6 and Table III it is clear that there exists observable differences in the performance of the proposed techniques. However, what is not clear is whether these differences is due to pure chance or sampling error. To address this we construct the following hypotheses;

$$H_0 : \tilde{M}_{\text{BAD}} = \tilde{M}_{\text{PAD}}$$
$$H_1 : \tilde{M}_{\text{BAD}} < \tilde{M}_{\text{PAD}}$$
$$H_2 : \tilde{M}_{\text{BAD}} > \tilde{M}_{\text{PAD}}$$

where $\tilde{M}$ is the median value of a given performance metric M. For the FR metric, the null hypothesis H_0 states that both methods generates roughly equal amount of false-alarms, while the alternative hypotheses H_1 states that BAD generates fewer false-alarms and H_2 state otherwise. The same interpretation can be given for the detection rate. A non-parametric one-sided Wilcoxon Signed Rank test is used to test H_0 against H_1 and H_2 at a *significance level* of 5% ($\boldsymbol{\alpha = 0.05}$). The tests are performed across three groups of datasets, non-seasonal, seasonal, and combined datasets. The resulting p-values are shown in Table IV for BAD and PAD and Table V for BAD* and PAD.

Based on the high p-values in Table IV we fail to reject the respective null hypothesis for the non-seasonal streams suggesting there exists no significant difference in performance. Similar conclusion is achieved on the seasonal streams except we reject H_0 in favour of H_1 for the F_2 score since $0.012 < \alpha$. Though there is no significant difference in detection rate across all datasets, the tests show that BAD has lower precision than PAD ($0.025 < \alpha$) across all streams and consequently produces more false-alarms ($0.041 < \alpha$). Conversely, looking at Table V we do not find any evidence that there is a significant difference in performance between PAD and BAD*.

3) Do the Methods Perform Any Better Than a Random Classifier?: Baseline anomaly detection can be achieved using a naive binary classifier which randomly assigns observations to one of two classes (i.e., `POSITIVE` or `NEGATIVE`) by tossing a coin. The classification quality, that is, the Matthews Correlation Coefficient (MCC) of such a classifier would be at most 0. A superior technique should have an $MCC \simeq +1$. Though considering the high detection rates it is easy to see

TABLE IV
p-VALUES FOR THE TEST OF SIGNIFICANCE BETWEEN BAD AND PAD

	Non-Seasonal			Seasonal			Combined		
	DR	**FR**	**F_2**	**DR**	**FR**	**F_2**	**DR**	**FR**	**F_2**
H_0 *vs* H_1	0.813	0.826	0.577	0.052	0.931	**0.012**	0.190	0.960	**0.025**
H_0 *vs* H_2	0.234	0.188	0.445	0.952	0.072	0.989	0.816	**0.041**	0.976

TABLE V
p-VALUES FOR THE TEST OF SIGNIFICANCE BETWEEN BAD* AND PAD

	Non-Seasonal			Seasonal			Combined		
	DR	**FR**	**F_2**	**DR**	**FR**	**F_2**	**DR**	**FR**	**F_2**
H_0 *vs* H_1	0.926	0.410	0.906	0.143	0.292	0.145	0.500	0.272	0.553
H_0 *vs* H_2	0.098	0.609	0.104	0.871	0.715	0.860	0.513	0.731	0.451

that the methods surpass such baseline model. To verify this, we examine the claim that the median MCC values are greater than zeroes using the following hypothesis:

$$H_0 : \tilde{M} = 0$$
$$H_1 : \tilde{M} > 0$$

where $\tilde{M}$ is the median MCC. We performed a one-sided Wilcoxon test for the proposed techniques at a 5% ($\boldsymbol{\alpha = 0.05}$) significance level across the datasets. We reject the H_0 in all cases since p-value $\ll 0.05$, which is enough evidence for our claim. Specifically, BAD has a lower correlation of 30% with the ground truth compared with 60% of its detrended variant BAD*. Both BAD* and PAD have an average positive correlation of about 60% across all datasets but the former shows a higher correlation of 70% on non-seasonal streams. The median MCC of ADVec is about 20% on seasonal streams but close to zero on non-seasonal ones. However, we observed about 60% positive correlation for its offline variant ADVec*.

4) To What Extent do the Methods Generalize on Independent Datasets?: Table VI shows the results of proposed methods on both the Web service and ISP traffic datasets (Fig. 5) with windows containing one week worth of data (i.e., $k = 168$) in both cases. The aggregate column on the right is mean of each metric across the datasets. Looking at the individual columns, the performance of the methods is, to some extent, comparable with that of Table III. Notice that the Amazon and BLiquidity data are non-seasonal and closer to A1A group (Fig. 3) in structure while the ISP traffic are seasonal with the U.K. and EU traffic closer to A1D and A1C

TABLE VI
PERFORMANCE OF METHODS ACROSS DIFFERENT DATASETS

	Amazon			BLiquidity			UK Internet Traffic			EU Internet Traffic			Aggregate		
	DR(%)	FR(%)	F_2(%)	DR(%)	FR(%)	F_2(%)	DR(%)	FR(%)	F_2(%)	DR(%)	FR(%)	F_2(%)	DR(%)	FR(%)	F_2(%)
BAD	100	0.4	87	73	0.5	49	43	0	48	11	0	13	57	0.2	49
BAD*	75	0.4	68	73	0.5	43	43	1.5	41	95	3.3	63	71	1.5	53
PAD	92	0.2	86	100	0.3	79	100	0.4	96	100	5.4	55	98	1.5	79

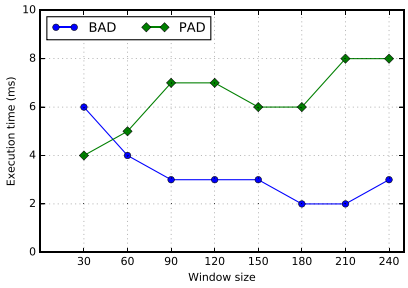


Fig. 7. Runtime requirement per train-test cycle.

respectively. Both BAD and PAD show greater performance on the Web services than on the ISP traffic. Similar to Fig. 6, BAD performs less on the U.K. traffic due to its irregular seasonality with trend and level-shift while BAD* works best on the EU traffic. Thanks to its ability to correctly distinguish contextual anomalies, PAD offers the most consistent performance having an aggregate accuracy and precision of 98% and 79% respectively with less than 2% false alarms rate.

E. Scalability Analysis

Fig. 7 demonstrates the scalability of proposed methods in terms of runtime requirement per train-test cycle obtained by varying the window size over a sample data. Clearly, the behaviour-based approach requires less time on average per cycle as window size increases compared to the prediction-based method. The reason for this disparity is due to two inherent characteristics of BAD. First is the tumbling window scheme used by BAD, which means that training is not performed as frequently as it is done in PAD. Secondly, the tree-based density estimator computes the density estimate of each data point using only its k-nearest neighbours without considering the geometrically distant points. While these two attributes make BAD seem to be more scalable, they however contribute to its poor precision on streams with level shifts and strong seasonal variation, cases in which PAD performs better since it is able to track changes almost immediately. The measurements were obtained on a machine with 2.6GHz Intel Core i5 processor and 8GB DDR3 memory.

VI. DISCUSSION

In this section, we discuss four main insights that can be inferred from the results in Section V.

1) Efficacy of an Online Anomaly Detection Technique Depends on Stream Behaviour: From the results, it is clear that no single technique performs uniformly well across all stream types and metrics (Fig. 6). For instance, BAD is better suited to non-seasonal datasets according to the statistical tests (Table IV). However, it generates many false-positives at the point of sharp level-shifts and takes a little more observations at the new level before it adjusts to the sudden change. BAD performs poorly on seasonal streams due to poor precision and prohibitive false-alarm rates especially those with long-term trend (Fig. 6). This is due to the fact that the density estimation does not track changing level or trends in seasonal streams. Also, the windowing method of the technique is not necessarily temporal, as the window size may be narrower or wider than the width of a season or period in the stream. Interestingly, detrending the stream, as in the case of BAD*, eliminates these weaknesses which greatly improves the performance across datasets irrespective of pattern as demonstrated by our results (see tables III, V, and VI).

There is also significant evidence that PAD works better on seasonal streams especially those without level shifts, and has comparable results with BAD on non-seasonal streams as well as with BAD*. This is due in part to its use of temporal non-linear splines which allows it to capture both stationary, trend and periodic patterns, and its explicit use of periodicity to determine the size of its sliding windows.

2) There Exist a Tradeoff Between Narrow and Wide Windows That Affect Performance: The effect of window size is much pronounced in BAD than in PAD as it fails to detect anomalies correctly and produce more false-alarms if there are too few observations in a window. A considerable amount of past is needed to properly generalize normal behaviour. Hence, the wider the windows the more accurate the estimated density profile becomes. Large window sizes, however, require more time to learn the model while smaller windows sizes are much faster. We balance this tradeoff by selecting window sizes in a data-driven way and observed the runtime requirement (*time-to-train-and-test*) of the behaviour-based approach to be on the average lower than the prediction-based method (see Fig. 7) owing to its tumbling window and nearest-neighbour based density estimation. Also, the effect of $\mathcal{L}$ on control charts is the deviation detectors are sensitive at a rate that is proportional to the control bounds, which are multiples of $\mathcal{L}$. The higher the value of $\mathcal{L}$ the wider the bounds of the control chart and consequently fewer false-alarms but with a greater chance of missing true anomalies. The converse is the case when the bounds are narrow. We observe that our adaptive z-score based approach to setting $\mathcal{L}$ yields a 60% and 7% improvement in precision for BAD and PAD techniques respectively compared to using a fixed value of 6.

3) Applying Off-the-Shelf Offline Techniques Is Not Suitable for Online Settings: Results for ADVec in batch mode suggests

that attempting to deploy offline algorithms in online settings results in poor accuracy and precision. This is because offline algorithms target global anomalies that are relevant only within the context of the larger input stream. If the input stream is a window of observations like in our experiment, ADVec disregards interesting behaviour in previous windows and simply identify local anomalies resulting in poor detection on non-seasonal metric streams. If the window is expanded to contain the entire stream, just as in offline setting, then it takes advantage of having a global view to detect global anomalies while disregarding local variations in the streams. This is evident by high performance of ADVec* in Table III.

4) Incremental Learning Makes Online Anomaly Detection Practical: A major insight from our results is that online anomaly detection can achieve acceptable accuracy via incremental learning. Since our techniques do not detect deviations based on static thresholds but from incremental estimates of an underlying property of the stream using only recent normal observations, they are able to better track the evolving stream. The adaptive nature of the control charts enable them to easily track dynamic variations in the stream in order to distinguish local anomalies from global ones. PAD has a major strength in that it can be easily adapted to predict anomalies up to at least half the length of its window into the future thereby making early-warning possible. Similarly, BAD* has the advantage that it can function well on diverse streams even when characteristics such as periodicity may be unknown. There is strong evidence that streams with sharp level-changes are particularly hard to deal with based on the poor precision of our techniques on such streams (see Fig. 6c and EU traffic results in Table VI).

While the sliding window based batch-incremental technique (i.e., PAD) seems most applicable under different settings since it explicitly exploits time-dependency, there is strong evidence that the tumbling window based batch-incremental approach, such as BAD*, that exploits statistical behaviour of metric streams can offer competitive performance if temporal characteristics such as trends are minimized.

VII. Related Work

There exist many prior work in performance anomaly detection in computing systems and in the general problem of anomaly detection that have been covered in previous surveys [4] and [14]. We highlight relevant works under four topics.

A. Network Monitoring

Anomaly detection remains a critical task in the field of network monitoring and security for diagnosing events that have severe impact on performance, security and QoS. This has led to a number of studies dedicated to using statistical learning to diagnose volume-based anomalies in network traffic [15]–[18]. However, alternative approaches such as the use of biologically inspired approaches have been explored in [40]. We only review relevant works addressing volume anomalies since it is similar to the problem of performance anomalies investigated in this paper.

As one of the early works in network anomaly detection, Lakhina *et al.* [16] proposed a PCA-based subspace method to diagnose volume anomalies in network traffic from multiple links. A prediction-based framework for traffic anomaly detection in large networks is proposed in [41]. The approach detects flow volume anomalies in multi-link network-wide traffic using real world and synthetic network traffic datasets for testing. The authors investigated and compare the performance of their method using multiple statistical detection schemes on prediction residual data with offline evaluation.

Also, Ahmed *et al.* [42] investigates the applicability of machine learning techniques for real-time anomaly detection in network traffic by comparing a block-based technique to a recursive method using traffic traces collected from an IP network and a network of webcams. They extended this work in [17] where KOAD, an incremental learning approach for real-time diagnosis of volume anomalies. KOAD exploits temporal variations in the structure of multivariate network traffic. Though it yields high detection accuracy and faster time-to-detection compared to a baseline approach, the method show sensitivity to varying detection threshold since selection is not automated. Also, it is not clear how it will perform with high dimensional traffic or datasets with irregular seasonality in different dimensions. Conversely, Kind *et al*. [19] introduced a feature-based approach which builds histogram models describing detailed characteristics of traffic features allowing to handle a wider range of anomalies unlike volume-based techniques. The effects and trade-offs of different design options and features are assessed using real world datasets.

A number of studies have also gone beyond detection anomalies to reasoning and identifying actual root-causes of observed network anomalies. The use of entropy- and distribution-based approaches for diagnosing traffic anomalies in Internet services (such as the DNS) is investigated in [43] using both real ISP traces and semi-synthetic mobile network traffic. Also, [44] builds on developments in statistical analysis and evidential reasoning to classify network anomalies based on their root-causes.

B. Software Performance Testing

Recent research in software performance testing focus on a related problem of the detecting regressions in the performance of software applications between releases with interesting proposals targeting pre- or post-deployment scenarios. Shang *et al*. [2] identify regressions (or deviations) between releases of an application service using machine learning. However, anomalies alerts are raised based on rather simple thresholds which may produce spurious false-alarms in streams with high variability and outliers. Other authors have employed statistical process control charts for the same problem. For example, the technique in [23] detects QoS violations using a simple Shewart chart, while [45] identifies performance regressions using CUSUM control charts. These charts are unsuitable for time-series streams, assume normal distributions and are not as sensitive to varying shifts as the EWMA chart. Our EWMA control chart is adaptive and robust against non-normality, small shifts and temporal streams.

C. Performance Diagnosis

Performance diagnosis frameworks often rely on simplistic static threshold-based mechanisms or intrusive instrumentation to identify deviations in service-level performance metrics. For example, point maxima thresholds and 3-sigma rules have been used in [20] to detect latency SLA violations, in EGADS [8] to detect change-points given expected behaviour of a metric stream, in CloudPD [12] to detect latency degradation, in E2EProf [46] to detect spikes in request latencies, and in CPI2 [11] to detect anomalies in per job cycle-per-instruction (CPI) of Google's web-search cluster. In addition, [9] assumes Gaussian distribution to define adaptive thresholds for uncovering anomalous behaviour in performance of an advertising service. The limitation of such techniques is sensitivity to variations in metric behaviour and high false-alarms when assumed distribution becomes obsolete.

The dynamic nature of cloud and Internet services today is driving the need for techniques that can autonomously parameterize and adapt to changes leading to adoption of machine learning methods. The wavelet-inspired method in [1] detects deviations based on a low-dimensional spectogram of normal service behaviour but requires extensive offline training. vPerfGuard [13] employs a sliding-window and prediction-based mechanism for diagnosing cloud service performance degradation. Performance deviations are detected via a regression model that relates service performance with other systems metrics. However, the approach requires extensive measurements of workloads and resource metrics and relies on the t-test which may produce high false-alarms even for slight changes in underlying distribution. Also, there is no evidence of how the approach will perform on service metric streams with seasonal patterns.

ATAD [10] define adaptive thresholds using time-series segmentation and localized statistics to detect anomalies. The segmentation technique goes over the data in two passes, first to segment the stream and second to find anomalies. Similarly, robust principal component analysis have been applied to adaptively identify service anomalies in [47]. However, only one stationary time-series stream from the Yahoo Webscope data was used.

Our approach address issues raised above in specific ways:

- Instead of assuming that statistical distribution of the stream is static, our proposed techniques can adapt to changing distribution and temporal dynamics in the underlying streams via incremental learning.
- Except for the windows size that is determined in a data-driven manner in our evaluation, the techniques require no need for offline parameterization and training, an attribute which makes them suitable for production and real-time environments.
- We address the issue of spurious false-alarms that is common with threshold-based approaches by exploiting the z-score distribution of recent normal observations in sequential temporal windows.
- The efficacy of our methods have been evaluated using diverse metric streams from three different real-world traces demonstrating their applicability for handling scale-out behaviour.

VIII. Conclusion

Detecting anomalies in sequential metric measurements in a timely fashion is crucial for preventing SLO violation and guaranteeing good user experience in application services and systems. While some existing solutions are rather too problem-specific due to rigid statistical assumptions, many others target offline scenarios where entire data streams can be used a priori for model training and selection. Such approaches may result in high false-alarm rates when applied in online scenarios as the models become obsolete which raise the need for efficient techniques that adapt with the evolving metric stream.

In this paper, we propose unsupervised schemes to address the problem of detecting contextual anomalies in performance metric streams. Through extensive evaluation by replaying over 40 diverse metric streams from the Yahoo! Webscope S5 datasets in addition to 4 Web service QoS and ISP traffic datasets, we have shown the efficacy of our approaches for online anomaly detection with high detection rates and few false-alarms. Our methods also achieve better accuracy compared to an open-source package for time-series anomaly detection. In general the highlights of this work are:

- We propose two adaptive techniques to detect contextual anomalies in continuous service measurements via incremental learning; (a) a behaviour-based technique exploiting changes in the underlying statistical behaviour of the stream, and (b) a prediction-based technique exploiting abrupt discontinuity in temporal behaviour. The techniques rely on statistically robust adaptive control charts to identify both small and large shifts.
- Our techniques are suitable for online environment with limited baseline data, parameterization, and offline training. Majority of anomalies were detected in the evaluation datasets mostly obtained from real systems, service and network traces.
- Results suggest that the behaviour-based approach is well-suited to stationary streams without trend while the prediction-based techniques is more robust to stream patterns. We also observed that trend removal via first-order differencing improves the generality of the behaviour-based techniques. Also, while tumbling window based learning is much faster, sliding window learning is better at tracking rapid changes in streams.

We are currently working on extending our approach for multivariate metric streams with emphasis on how to automatically explain and mitigate anomalies. By complementing our techniques with automated root-cause analysis and automated mitigation, we can reduce diagnosis effort and cost and prevent unnecessary service degradations or outages.

References

[1] D. O'Shea *et al.*, "A wavelet-inspired anomaly detection framework for cloud platforms," in *Proc. 6th Int. Conf. Cloud Comput. Services Sci. (CLOSER)*, Rome, Italy, 2016, pp. 106–117.

[2] W. Shang, A. E. Hassan, M. Nasser, and P. Flora, "Automated detection of performance regressions using regression models on clustered performance counters," in *Proc. 6th ACM/SPEC Int. Conf. Perform. Eng.*, Austin, TX, USA, 2015, pp. 15–26.

[3] *The Performance of Web Applications: Customers Are Won or Lost in One Second (Research Report)*, Aberdeen Group, Boston, MA, USA, May 2015, accessed: Mar. 31, 2016. [Online]. Available: http://www.aberdeen.com/research/5136/ra-performance-web-application/content.aspx

[4] O. Ibidunmoye, F. Hernández-Rodriguez, and E. Elmroth, "Performance anomaly detection and bottleneck identification," *ACM Comput. Surveys*, vol. 48, no. 1, pp. 1–35, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2791120

[5] M. Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: Problem determination in large, dynamic Internet services," in *Proc. Int. Conf. Depend. Syst. Netw. (DSN)*, Bethesda, MD, USA, 2002, pp. 595–604.

[6] R. R. Sambasivan *et al.*, "Diagnosing performance changes by comparing request flows," in *Proc. 8th USENIX Conf. Netw. Syst. Design Implement.*, Boston, MA, USA, 2011, pp. 43–56.

[7] D. J. Dean, H. Nguyen, P. Wang, and X. Gu, "PerfCompass: Toward runtime performance anomaly fault localization for infrastructure-as-a-service clouds," in *Proc. 6th USENIX Conf. Hot Topics Cloud Comput.*, Philadelphia, PA, USA, 2014, p. 16.

[8] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Sydney, NSW, Australia, 2015, pp. 1939–1947.

[9] B. Zhou and S. Shariat, "Finding needle in a million metrics: Anomaly detection in a large-scale computational advertising platform," *ArXiv e-prints, arXiv:1602.07057v1 [cs.AI]*, Feb. 2016.

[10] M.-C. Dani, F.-X. Jollois, M. Nadif, and C. Freixo, "Adaptive threshold for anomaly detection using time series segmentation," in *Proc. Int. Conf. Neural Inf. Process.*, Istanbul, Turkey, 2015, pp. 82–89.

[11] X. Zhang *et al.*, "CPI2: CPU performance isolation for shared compute clusters," in *Proc. 8th ACM Eur. Conf. Comput. Syst.*, Prague, Czechia, 2013, pp. 379–391.

[12] B. Sharma, P. Jayachandran, A. Verma, and C. R. Das, "CloudPD: Problem determination and diagnosis in shared dynamic clouds," in *Proc. 43rd Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, Budapest, Hungary, 2013, pp. 1–12.

[13] P. Xiong, C. Pu, X. Zhu, and R. Griffith, "vPerfGuard: An automated model-driven framework for application performance diagnosis in consolidated cloud environments," in *Proc. 4th ACM/SPEC Int. Conf. Perform. Eng.*, Prague, Czechia, 2013, pp. 271–282.

[14] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surveys*, vol. 41, no. 3, 2009, Art. no. 15.

[15] J. D. Brutlag, "Aberrant behavior detection in time series for network monitoring," in *Proc. LISA*, vol. 14. 2000, pp. 139–146.

[16] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 219–230, 2004.

[17] T. Ahmed, M. Coates, and A. Lakhina, "Multivariate online anomaly detection using kernel recursive least squares," in *Proc. 26th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Barcelona, Spain, 2007, pp. 625–633.

[18] E. P. Freire, A. Ziviani, and R. M. Salles, "Detecting VoIP calls hidden in Web traffic," *IEEE Trans. Netw. Service Manag.*, vol. 5, no. 4, pp. 204–214, Dec. 2008.

[19] A. Kind, M. P. Stoecklin, and X. Dimitropoulos, "Histogram-based traffic anomaly detection," *IEEE Trans. Netw. Service Manag.*, vol. 6, no. 2, pp. 110–121, Jun. 2009.

[20] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, "SLA-driven automatic bottleneck detection and resolution for read intensive multi-tier applications hosted on a cloud," in *Proc. Adv. Grid Pervasive Comput.*, Hualien, Taiwan, 2010, pp. 37–46.

[21] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Netw. Syst. Manag.*, vol. 23, no. 3, pp. 567–619, 2015.

[22] M. Yoshizawa, T. Sato, and K. Naono, "Integrated monitoring software for application service managers," *IEEE Trans. Netw. Service Manag.*, vol. 11, no. 3, pp. 321–332, Sep. 2014.

[23] T. H. D. Nguyen *et al.*, "Automated detection of performance regressions using statistical process control techniques," in *Proc. 3rd ACM/SPEC Int. Conf. Perform. Eng.*, Boston, MA, USA, 2012, pp. 299–310.

[24] Y.-K. Chen, "Challenges and opportunities of Internet of Things," in *Proc. 17th Asia South Pac. Design Autom. Conf.*, Sydney, NSW, Australia, 2012, pp. 383–388.

[25] C. C. Aggarwal, *Outlier Analysis*. New York, NY, USA: Springer, 2013.

[26] D. C. Montgomery, *Introduction to Statistical Quality Control*. New York, NY, USA: Wiley, 2007.

[27] S. Malkowski, M. Hedwig, and C. Pu, "Experimental evaluation of N-tier systems: Observation and analysis of multi-bottlenecks," in *Proc. Int. Symp. Workload Characterization (IISWC)*, Austin, TX, USA, 2009, pp. 118–127.

[28] F. J. G. Gisbert, "Weighted samples, kernel density estimators and convergence," *Empir. Econ.*, vol. 28, no. 2, pp. 335–351, 2003.

[29] D.-Y. Yeung and C. Chow, "Parzen-window network intrusion detectors," in *Proc. 16th Int. Conf. Pattern Recognit.*, vol. 4. Quebec City, QC, Canada, 2002, pp. 385–388.

[30] R. N. Riegel, "Generalized n-body problems: A framework for scalable computation," Ph.D. dissertation, School Comput. Sci. Eng., Georgia Inst. Technol., Atlanta, GA, USA, 2013.

[31] J. Read, A. Bifet, B. Pfahringer, and G. Holmes, "Batch-incremental versus instance-incremental learning in dynamic and evolving data," in *Proc. Int. Symp. Intell. Data Anal.*, Helsinki, Finland, 2012, pp. 313–323.

[32] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. Heathmont, VIC, Australia: OTexts, 2014, accessed: Feb. 10, 2016.

[33] D. Zheng, F. Li, and T. Zhao, "Self-adaptive statistical process control for anomaly detection in time series," *Expert Syst. Appl.*, vol. 57, pp. 324–336, Sep. 2016.

[34] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn, "Self-adaptive workload classification and forecasting for proactive resource provisioning," *Concurrency Comput. Pract. Exp.*, vol. 26, no. 12, pp. 2053–2078, 2014.

[35] A.-R. Rezaie, "An automated forecasting method for workloads on Web-based systems," M.S. thesis, Dept. Math. Math. Stat., Umeå Univ., Umeå, Sweden, 2014.

[36] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Internet traffic forecasting using neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Vancouver, BC, Canada, 2006, pp. 2635–2642.

[37] K. Arun. (2015). *Twitter: Introducing Practical and Robust Anomaly Detection in a Time Series*. Accessed: Aug. 30, 2016. [Online]. Available: https://webscope.sandbox.yahoo.com/catalog.php?datatype=sAdditional

[38] Yahoo! Webscope. *S5—A Labeled Anomaly Detection Dataset, Version 1.0*. Accessed: Jan. 19, 2016. [Online]. Available: https://webscope.sandbox.yahoo.com/catalog.php?datatype=s

[39] B. Cavallo, M. Di Penta, and G. Canfora, "An empirical comparison of methods to support QoS-aware service selection," in *Proc. 2nd Int. Workshop Principles Eng. Service Orient. Syst.*, Cape Town, South Africa, 2010, pp. 64–70.

[40] F. Hashim, K. S. Munasinghe, and A. Jamalipour, "Biologically inspired anomaly detection and security control frameworks for complex heterogeneous networks," *IEEE Trans. Netw. Service Manag.*, vol. 7, no. 4, pp. 268–281, Dec. 2010.

[41] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *Proc. 5th ACM SIGCOMM Conf. Internet Meas.*, Berkeley, CA, USA, 2005, p. 31.

[42] T. Ahmed, B. Oreshkin, and M. Coates, "Machine learning approaches to network anomaly detection," in *Proc. 2nd USENIX Workshop Tackling Comput. Syst. Problems Mach. Learn. Techn.*, 2007, pp. 1–6.

[43] P. Fiadino, A. DAlconzo, M. Schiavone, and P. Casas, "RCATool—A framework for detecting and diagnosing anomalies in cellular networks," in *Proc. 27th Int. Teletraffic Congr. (ITC)*, Ghent, Belgium, 2015, pp. 194–202.

[44] N. Samaan and A. Karmouch, "Network anomaly diagnosis via statistical analysis and evidential reasoning," *IEEE Trans. Netw. Service Manag.*, vol. 5, no. 2, pp. 65–77, Jun. 2008.

[45] A. Amin, A. Colman, and L. Grunske, "Statistical detection of QoS violations based on CUSUM control charts," in *Proc. 3rd ACM/SPEC Int. Conf. Perform. Eng.*, Boston, MA, USA, 2012, pp. 97–108.

[46] S. Agarwala, F. Alegre, K. Schwan, and J. Mehalingham, "E2EProf: Automated end-to-end performance management for enterprise systems," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, Edinburgh, U.K., 2007, pp. 749–758.

[47] B. Agrawal, T. Wiktorski, and C. Rong, "Adaptive anomaly detection in cloud using robust and scalable principal component analysis," in *Proc. 15th Int. Symp. Parallel Distrib. Comput. (ISPDC)*, Fuzhou, China, 2016, pp. 100–106.

Olumuyiwa Ibidunmoye received the M.Tech. degree in computer science from the Federal University of Technology Akure, Nigeria. He is currently pursuing the Ph.D. degree with the Distributed Systems Group, Umeå University, Sweden. He joined the VR funded Cloud Control project in 2013, where he has been investigating how adaptive techniques based on statistical and machine learning as well as control techniques can be used to enhance autonomous management in cloud infrastructures.

Ali-Reza Rezaie received the M.Sc. degree in mathematical statistics from Umeå University, Sweden, where he, in collaboration with the Distributed Systems Group, worked on automated forecasting of Web-based workloads for his master's thesis and later as a Research Assistant on the Cloud Control Project.

Erik Elmroth is a Professor with Umeå University, Sweden, where he had been the Head and the Deputy Head with the Department of Computing Science for ten years. He is currently leading the Distributed Systems Group, the VR funded Cloud Control project, and was the Chair of the Swedish National Infrastructure for Computing. His main research interests include cloud, distributed, and autonomic computing, as well as high-performance computing.