Runtime state change detector of computer system resources under non stationary conditions

Sara Casolari, Michele Colajanni
Department of Information Engineering
University of Modena and Reggio Emilia
{sara.casolari,michele.colajanni}@unimore.it

Francesco Lo Presti
Department of Computer Engineering
University of Rome Tor Vergata
lopresti@info.uniroma2.it

***Abstract*—All runtime management decisions in computer and information systems require immediate detection of relevant changes in the state of their resources. This is accomplished by continuously monitoring the performance/utilization of key system resources and by using appropriate statistical tests to detect the occurance of significant state changes. Unfortunately, the complexity of today systems and applications and the unpredictability of user request patterns result in highly variable and non stationary time series which are difficult to analyze. As a consequence, present solutions for detecting state changes at runtime are affected by excessive time delays or false positives. We propose a novel "agile" runtime detector that solves the delay vs. false positive tradeoff: it is able to detect the relevant state changes as fast as the best reactive models with the lowest percentages of false positives. All evaluations carried out for a large set of scenarios confirm the efficacy and robustness of the proposed model.**

I. Introduction

Most management decisions related to computer and information systems are activated after a notification that a relevant state variation has occurred in some system resource(s). Request redirection, resource monitoring traffic anomaly, diagnosis and fault detection of the system components, process migration, access control and any autonomic-related control are examples of processes that are activated after the detection of a significant and non-transient system state change. For this reason, the most important system resources are continuously monitored and data are passed to some statistical model that decides whether a relevant state change has occurred or not. The main difficulty is to detect at runtime relevant statistical changes in time series deriving from monitored system resources that are highly variable and non-stationary, and may be affected by perturbations and by non-deterministic behavior when passing from one state to another. The quality of the detection problem is exacerbated when the state change detector has to support *runtime* decisions instead offline analysis.

In this paper we propose and compare a novel runtime methodology to solve the change detection problem in the context of computer and information systems, where the time series represent continuous measures of the state of the internal resources. Several models that can be applied to detect relevant state changes, such as Particle Filtering [1], [2], Kalman filter [3], Sequential Monte Carlo Method [4], and Bayesian Bootstrap Filtering [5], require an anticipated analysis to determine some statistical characteristics of the time series [6], [7]. In this paper we consider time series that are non-stationary, non-deterministic and are characterized by a variable variance of the data distribution. These data sets are typical of modern information systems where multicore architectures host multiple interactive Internet/Web-based services. These characteristics prevent the possibility of defining one statistical model with a set of a statistically determined parameters that is able to represent the evolution of the entire time series. Other traditional state change detectors are inadequate because they are affected either by an extensive delay for detecting a state change or by a large number of false detections. This tradeoff is a well known issue that is very difficult to address, especially in the considered non-stationary conditions.

We propose a new runtime model that is based on an evolution of the Cumulative Sum (CUSUM) algorithm [8]. It allows us to minimize the number of false detections with respect to other runtime models such as GLR [8] and baseline CUSUM [9], still guaranteeing a delay for the detection that is comparable to the existing fastest models that, on their hand, are affected by enormous percentages of false detections. The proposed model is able to address the non-stationarity and the large variance of the considered time series and, most importantly for the considered application context, it shows high robustness to noise variation.

In this paper, we evaluate the quality of the proposed detector for synthetic and realistic time series coming from a typical Web-based system and compare it against existing runtime detection models. In all of our experiments, the proposed model minimizes the number of false detections and is able to guarantee the robustness of these results for different noise variations.

The rest of the paper is organized as follows. In Section II we present related work and compare our contribution with the state of the art. Section III discusses the motivations of this research. We propose a formal definition of runtime state change detection and of the metrics for evaluating the quality of the models. Section IV describes the traditional CUSUM detector and the modified version. In Section V we present another runtime state detection algorithm that we consider in our evaluation. Section VI compares the quality and the

efficacy of the three considered state change detection models. We conclude the paper in Section VII with some final remarks.

II. Related work

There are two main classes of state change detection algorithms: off-line and on-line. In off-line detection [10], the entire data set is available before the analysis that aims to evaluate medium-long term characteristics. The computational complexity of the off-line detection algorithms is not a big issue because these algorithms are not subject to severe temporal constrains.

In on-line detection, that is of interest for this paper, the detection algorithms must be efficacious and efficient with respect to the time constraint imposed by the application context. These algorithms play a wide and crucial role in most stream processing problems related to anomaly detection, quality control, request redirection, resource management, diagnosis and fault detection of the system components, process migration, access control and limitation and, in general, in all situations where some real-time decision or reaction must be taken. We can classify on-line detection models in three main categories: *deterministic*, *probabilistic* and *threshold-based*.

The *deterministic* models require some a priori knowledge of the data set especially about the probability of the state changes and their distributions. They are based on a representation of the system state that may be described through some Bayesian filtering models [7], such as the Particle Filtering [1], [2], the Kalman filter [11], the Sequential Monte Carlo Method [4], the Bayesian Bootstrap Filtering [5]. As in this paper we consider scenarios characterized by non-deterministic behaviors, all the deterministic models are inadequate.

The *probabilistic* methods are closer to the context considered in this paper, although with some significant differences. They operate on a representation of the time series combined with some statistical tests [3], [6], [8], [9] and require some knowledge about the system being monitored. A popular statistical technique that can support on-line state change detections is the Cumulative Sum (CUSUM) model [3], [8], which uses information from one representation of the time series to decide whether a stage change occurs or not in the system. The full Generalized Likelihood Ratio (GLR) test [11], [12] requires multiple time series representations, hence its complexity grows linearly in time. All the *probabilistic* models are affected by a severe trade-off between the delay for detection and the number of false detections. The literature has not yet addressed this well known issue. For example, the CUSUM and GLR models tend to introduce a number of false detections that increase significantly as the variance of the time series augments. Hence, their results are poor in the considered context that is characterized by extremely variable and noisy time series. The proposed ARL0 CUSUM model is the first to achieve a good trade-off between the percentage of false detections and the mean delay for detection. Its results are appreciable for any considered time series profile, for a wide range of noise perturbations and different correlation levels among the time series values.

Another class of approaches uses the *threshold* methods to support state change detection without the need for statistical test [13]–[15]. These solutions are not comparable to the *deterministic* and *probabilistic* models because their performance depends on the choice of the threshold value(s). There is no theoretical support for the right choice of the threshold that depends completely on the application context and workload. For these reasons, all the comparisons will be carried out against *probabilistic* models.

III. Problem definition

The runtime detection of a relevant state change refers to the models and methodologies that are able to decide whether such a change occurred by evaluating the statistical characteristics of a monitored system resource(s). From the practical point of view, these methodologies are at the basis of the most important runtime decisions in computer and information systems. All applications are mainly interested to *state changes* that occur rapidly with respect to the period of measurement and that last for a certain time interval. Let $\{y_i\} = (y_1, \ldots, y_N)$ be a time series with conditional density $p_\vartheta(y_i \mid y_{i-1}, \ldots, y_1)$, where $1 \leq i \leq N$. A time series *state* is characterized by a constant conditional density parameter ϑ. Before the time t_0, that is assumed as the instant of the state change, the conditional density parameter ϑ is equal to ϑ_0. After t_0, the same parameter is equal to ϑ_1. The problem we consider is to detect the occurrence of the state change as soon as possible, with a minimum or null rate of false detections before and after t_0.

The state change may occur in an increasing or decreasing direction of the time series values. The statistical *change detection rule* for an increasing detection is typically obtained by comparing the *test statistics* g_i at time i with a characteristic threshold H. There are several test statistics, but their common point is that they can be expressed as a function of the likelihood techniques [9]:

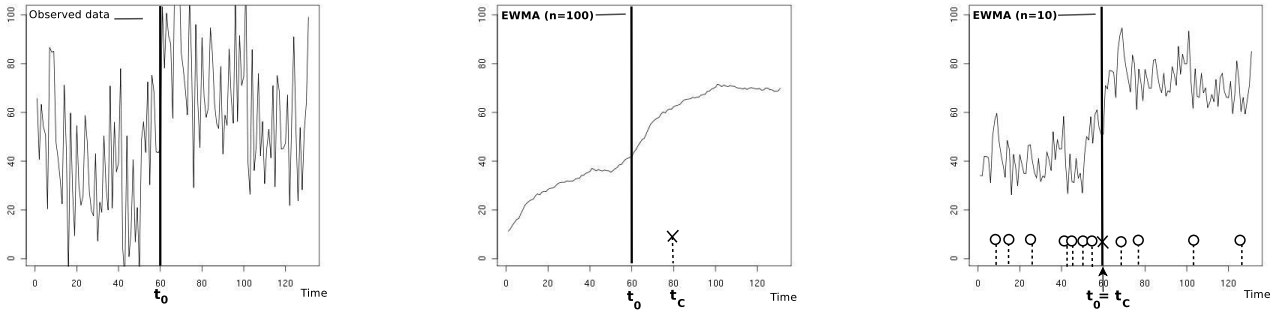
$$g_i \simeq \sum_{j=i-(n-1)}^{i} \ln \frac{p_{\vartheta_1}(y_j)}{p_{\vartheta_0}(y_j)} \tag{1}$$

where $p_{\vartheta_1}(y_j)$ and $p_{\vartheta_0}(y_j)$ are the two distributions characterizing the time series before and after t_0, and n is the number of the considered past elements. The choice of the characteristic threshold H depends on the test statistics g_i for state change detection and on the type of performance required by the state change detection algorithm. The problem of the right choice for H has been extensively addressed in the quality control literature [16]. The *change detection rule* in the increasing direction occurs when

$$g_i \geq H \tag{2}$$

Analogously, the change detection rule in the decreasing direction is denoted by:

$$g_i \leq -H \tag{3}$$



(a) Time series profile **(b) Delay for detection** **(c) False detections**

Fig. 1. State change detection algorithms evaluation.

We define the *change detection time*, t_c, as the instant at which the test statistics g_i verifies the change detection rule of the Equations 2 and 3 to detect the relevant state variation. This instant is given by

$$t_c = \mathbf{inf}\{i : g_i \geq H\} \tag{4}$$

$$t_c = \mathbf{inf}\{i : g_i \leq -H\} \tag{5}$$

for the detections in the increasing and decreasing direction, respectively.

The choice of a valid state change detection model depends on several factors, among which the statistical properties of the time series and the time constraints of the application are two of the most important. Existing models are affected either by too many false detections or by excessive time delays in signaling a state change. We formalize these metrics by considering a time series subject to one state change at time t_0 from the state with conditional density ϑ_0 to the state characterized by the parameter ϑ_1.

The **delay for detection** is related to the ability of an algorithm to detect a state change when it actually occurs. It quantifies the time delay for the detection of a new state through the distances $\tau = t_c - t_0$ between the change detection time and the actual change time. The best runtime algorithm for change detection minimizes τ.

The **percentage of false detections** is measured as the ratio between the total number of false detections and the total number of detections signaled by the state change detector, that is,

$$FP = \frac{\left|\sum_{i=1}^{N}(g_i \geq H) + \sum_{i=1}^{N}(g_i \leq -H) - d_0\right|}{\sum_{i=1}^{N}(g_i \geq H) + \sum_{i=1}^{N}(g_i \leq -H)} * 100 \tag{6}$$

where $\sum_{i=1}^{N}(g_i \geq H)$ and $\sum_{i=1}^{N}(g_i \leq -H)$ are the total number of detections in a time series of N values, d_0 is the number of real state changes. The best runtime algorithm for state change detection is characterized by no false detection.

In Figure 1, we give some examples about the problems that may affect a state change detection by considering the Exponential Weighted Moving Average model (EWMA [16]), that will be described in Section V.

An example of time series that we want to analyze is represented in Figure 1 (a): it shows non-stationary statistical characteristics, high and variable variance. This behavior can be found in the time series related to the performance indicator of the resources (e.g., CPU utilization, disk and network throughput) of modern information systems. In the considered case, at time $t_0 = 60$, the time series is subject to an external perturbation and the time series values grow significantly. Consequently, the time $t_0 = 60$ represents the instant of the real state change generated by the external perturbation. A state change detection model has to detect as soon as possible this change by evaluating the test statistics g_i of the time series. Figures 1 (b) and (c) evidence the typical trade-off of current state change detectors by considering the application of the EWMA detector working on 100 and 10 past values of the time series, respectively. The bold vertical line in Figure 1 (a) represents the real state change detections, while in Figure 1 (b) and (c) the small line with a circle head denotes a false detection, and the vertical line with a cross denotes a true detection signaled by the two EWMA models. These figures show that the model with $n = 100$ is not affected by any false detection, but it signals the state change occurring at time $t_0 = 60$ with a significant delay at time $t_c = 80$. This delay may be a serious drawback for certain classes of runtime decisions. The model with $n = 10$ is much more responsive and it is able to detect immediately the state variation at time $t_0 = t_c = 60$. On the other hand, this model is affected by several false detections that are evidenced by the eleven lines with a circle head.

The state change detection model proposed in Section IV aims to solve this trade-off between detection delay and false detections. We compare the characteristics of existing and proposed state change detection models through two metrics that are commonly used in this context [8]: delay for detection and percentage of false detections.

IV. CUSUM Models

CUSUM is a widely used model for detecting changing points in, otherwise stationary, time series. Proposed by Page [10], this technique has been extensively studied since. The CUSUM algorithm has been shown to be optimal when the

variance of the time series does not change in that it guarantees minimum mean delay to detection t_c in the asymptotic regime when the mean time between false alarms goes to infinity [17].

In the following we first describe the baseline CUSUM algorithm and discuss its performance. We will then illustrate our modified version of the CUSUM algorithm which is specifically designed to handle the variable and non-stationary behavior of the internal system resources of modern information systems.

A. The baseline CUSUM algorithm

Given a time series $\{y_i\}$, $i = 1, \ldots, N$, the one-sided CUSUM for detecting an increase in the mean computes the following test statistics

$$g_0^+ = 0 \quad (7)$$

$$g_i^+ = \max\{0, g_{i-1}^+ + y_i - (\mu_0 + K^+)\} \quad (8)$$

which measures positive deviations from a nominal value μ_0. The test statistics g_i^+ accumulates deviations of y_i from the target value μ_0 that are greater than a pre-defined constant K^+, and resets to 0 on becoming negative. The term K^+, which is known as the allowance or slack value, determines the minimum deviation that the statistics g_i^+ accounts for. A positive change is signaled when g_i^+ exceeds a chosen threshold H^+.

The one-sided CUSUM algorithm for detecting negative deviations is defined similarly, as

$$g_0^- = 0 \quad (9)$$

$$g_i^- = \max\{0, g_{i-1}^- + (\mu_0 - K^-) - y_i\} \quad (10)$$

A negative change is signaled when g_i^- exceeds a chosen threshold H^-.

A two-sided algorithm to detect both increases and decreases is obtained by applying the two tests simultaneously. As in this paper we are interested in detecting both increases and decreases we will consider the two-sided CUSUM model. For the sake of simplicity we will consider the symmetric case whereby $K^+ = K^- = K$ and $H^+ = H^- = H$.

When a shift is detected, the CUSUM model also estimates the new system state μ_1 as follows:

$$\mu_1 = \begin{cases} \mu_0 + K + \frac{g_i^+}{D^+} & \text{if } g_i^+ > H \\ \mu_0 - K - \frac{g_i^-}{D^-} & \text{if } g_i^- > H \end{cases} \quad (11)$$

where D^+ (D^-) denotes the number of steps elapsed since the last time g_i^+ (g_i^-) was set to zero, that is $D^+ = i - \inf\{j \mid g_j^+ = 0\}$ and similarly for D^-. The performance of the CUSUM model is expressed in terms of the so called *Average Run Lengths* (ARL): ARL_0 denotes the average number of samples between false alarms when no change has occurred; ARL_1 denotes the average number of samples to detect a change when it does occur. Ideally, ARL_0 should be large while ARL_1 should be small. Both ARL measures are affected by the design parameters H and K. To achieve good performance, the suggested values are $K = \frac{\Delta}{2}$, where Δ is the minimum shift to be detected, and $H = 5\sigma_y$, where σ_y is the time series standard deviation [16]. The choice of $K = \frac{\Delta}{2}$ has been shown to provide near minimal ARL_1 for a wide range of threshold values H, while $H = 5\sigma_y$ guarantees a high ARL_0 for a shift of $\Delta = \sigma_y$ which is typically considered as a reference value.

There are several techniques to compute ARL_0 and ARL_1. In this paper, we consider the Siegmund approximation [18] that combines simplicity and efficacy. For a one-sided CUSUM with parameters K and H, the Siegmund approximation gives:

$$ARL = \frac{e^{-2\eta(\frac{H}{\sigma_y}+1.166)} + 2\eta(\frac{H}{\sigma_y} + 1.166) - 1}{2\eta^2} \quad (12)$$

where $\eta = \frac{\Delta^* - K}{\sigma_y}$ for the upper one-sided CUSUM and $\eta = \frac{-\Delta^* - K}{\sigma_y}$ for the lower one-sided CUSUM. Δ^* is a constant which must be set to 0 for computing ARL_0^+ and ARL_0^-, and to Δ for computing ARL_1^+ and ARL_1^-. If $\eta = 0$, the Siegmund approximation is simply $ARL = (\frac{H}{\sigma_y} + 1.166)^2$. The ARL of the two-sided CUSUM is related to the ARLs of the two-sided statistics as follows:

$$L = \frac{L^+ L^-}{L^+ + L^-} \quad (13)$$

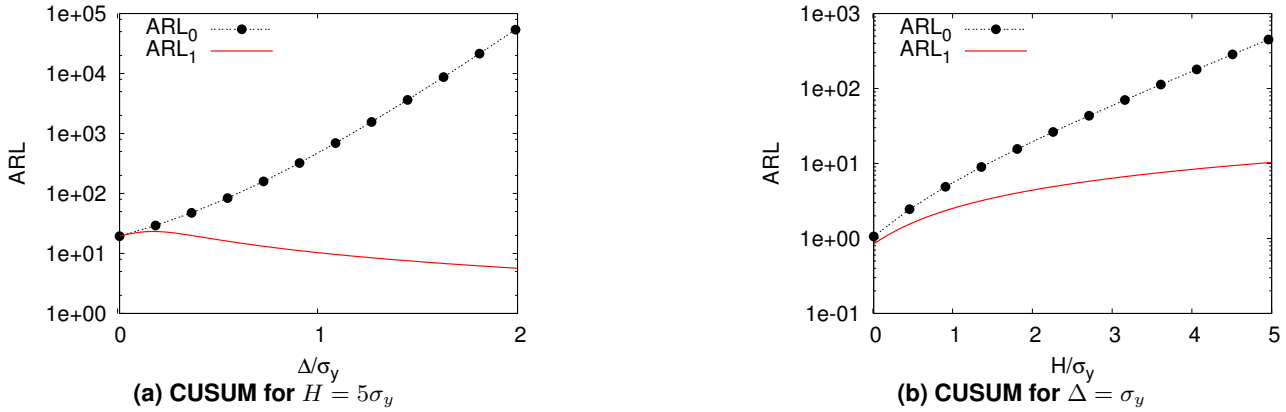
where $L \in \{ARL_0, ARL_1\}$.

In Figures 2 (a) and 2 (b) we report ARL_0 and ARL_1 for $H = 5\sigma_y$ and $\Delta = \sigma_y$, respectively. In both plots, $K = \frac{\Delta}{2}$. Note that the ARL performance depends heavily on the ratio Δ/σ_y and H/σ_y . ARL_0 improves its performance for increasing values of Δ/σ_y and H/σ_y; ARL_1 performance improves for larger values of Δ/σ_y, since it is easier to detect larger shifts, but it degrades for larger values of H/σ_y as it takes more time to detect a shift. CUSUM performance is critical for systems that exhibit large variance with respect to the shift to be detected. Indeed, for small Δ/σ_y values, ARL_0 decreases almost exponentially for a fixed H.

B. The ARL0 CUSUM algorithm

The baseline CUSUM algorithm is the best choice for statistical quality control of many processes [16], but it cannot be directly applied as a runtime state change detector in the modern computer systems. We should consider that the CUSUM algorithm requires a reference value μ_0 for a time series against which measuring the time series deviations. Our experience shows that there is no notion of reference value in resource dynamics of modern systems: indeed, CPU, disk and network usage dynamics are clearly non-stationary with all relevant statistics, (e.g., mean value and standard deviation vary over time). Moreover, the non-stationarity makes very difficult to set the design parameters H and K that can guarantee good performance over time. This is critical since for any fixed value of H the CUSUM performance rapidly degrades as the time series standard deviation σ_y increases.

In this section, we propose a version of the CUSUM algorithm, namely ARL0 CUSUM, capable of detecting state changes in face of variable and non-stationary characteristics of the time series. The proposed algorithm aims to solve the limitations of the baseline CUSUM algorithm as following:



(a) CUSUM for $H = 5\sigma_y$ **(b) CUSUM for $\Delta = \sigma_y$**

Fig. 2. Average Run Length for the baseline CUSUM algorithm

1) we replace the reference value μ_0 by a continuous estimate $\widehat{\mu}_i$ of the time series mean, *i.e.*, of the system state;
2) we replace the standard deviation σ_y in (12) by a continuous evaluation $\widehat{\sigma}_i$ of the time series variance;
3) we dynamically adjust the static threshold H with H^* as to provide a target ARL_0 performance in spite of a high and possibly time-varying variance.

The proposed CUSUM-based detector consists of two components: an EWMA filter that tracks the slow varying mean, and a two-sided CUSUM algorithm with varying thresholds for detecting abrupt state changes. We consider the following EWMA filter:

$$\widehat{\mu}_i = \alpha y_i + (1-\alpha)\widehat{\mu}_{i-1} \tag{14}$$

where $0 < \alpha \leq 1$ is typically set to $1/(1+2\pi * f)$ and f is the cutoff frequency of the EWMA filter [19].

The time series variance σ_y is in general unknown and varying over time. We resort to a widely adopted approximation of the variance that, for the sake of simplicity necessary in an online context, basically replaces the standard deviation σ_y with the mean deviation $E[\mid y - E[y] \mid]$ computed over time [20]:

$$\widehat{\sigma}_i = \alpha \mid y_i - \widehat{\mu}_i \mid + (1-\alpha)\widehat{\sigma}_{i-1}. \tag{15}$$

where $0 < \alpha \leq 1$ is the same as in (14).

By combining (12) and (15) and by setting a suitable high value for ARL_0^* so to guarantee good performance in terms of low false detections, we are able to dynamically adjust the value of the threshold H^* to reflect the variation of the system variance as to keep a desired performance of the state change detector. The computation of H^* for the desired ARL_0^* is carried out by a numerical inversion of the equation (12) where the parameters are set as follows: $K = \frac{\Delta}{2}$, where Δ is the smallest shift we want to detect (state changes smaller of Δ are accounted for by $\widehat{\mu}_i$ which tracks the state); η is replaced with $K/\widehat{\sigma}_i = \Delta/(2\widehat{\sigma}_i)$. Hence equation (12) to evaluate H^* becomes:

$$ARL_0^{+/-} = \frac{e^{\frac{\widehat{\Delta}}{\widehat{\sigma}_i}(\frac{H^*}{\widehat{\sigma}_i}+1.166)} - \frac{\widehat{\Delta}}{\widehat{\sigma}_i}(\frac{H^*}{\widehat{\sigma}_i}+1.166) - 1}{\frac{\widehat{\Delta}^2}{2\widehat{\sigma}_i^2}} \tag{16}$$

The detection rule is provided by modifying (8) and (10) as follows:

$$g_0^+ = 0 \tag{17}$$
$$g_i^+ = \max\{0, g_{i-1}^+ + y_i - (\widehat{\mu}_i + K^+)\} \tag{18}$$

$$g_0^- = 0 \tag{19}$$
$$g_i^- = \max\{0, g_{i-1}^- + (\widehat{\mu}_i - K^-) - y_i\} \tag{20}$$

where μ_0 is replaced by the tracked state $\widehat{\mu}_i$. A change is detected whenever g_i^+ or g_i^- exceeds the threshold H^*. When a change occurs, the tracked state $\widehat{\mu}_i$ is updated using (14).

V. Other Stage Change Detection Models

The choice of the most appropriate model for state change detection depends on several factors, among which the application context and the statistical properties of the time series are the most important. As our focus is on state change detection algorithms working at runtime in the context of computer and information systems, many existing models cannot be considered because intrinsically unsuitable. Another important premise is that the considered time series are characterized by non-stationary and non-deterministic behavior in highly variable contexts. These features further limit the subset of appropriate models. For example, popular state change detection models, such as the Kalman filter [11], the Bayesian Filters [5], and variation thereof as the Particle Filtering [1], [2], the Sequential Monte Carlo Method [4], require an accurate pre-analysis of the behavior of the entire time series dynamics. Hence, as a term of comparison we consider the Exponential Weighted Moving Average (EWMA) that is largely used both as a filter [19] and as a state variation detector [16].

The state function of EWMA is defined as: $g_i = \lambda y_i + (1-\lambda)g_{i-1}$, where typical values are $\lambda = \frac{2}{n+1}$ and $g_0 =$

$\sum_{i=1}^{n}(y_i)/n$ [16]. The value g_i characterizes the central tendency of the current state.

The *detection rule* of the EWMA state change detection model is based on:

$$g_i \geq g_0 + M\sigma_y\sqrt{\frac{\lambda}{2-\lambda}} \text{ and } g_i \leq g_0 - M\sigma_y\sqrt{\frac{\lambda}{2-\lambda}} \quad (21)$$

where $g_0 + M\sigma_y\sqrt{\frac{\lambda}{2-\lambda}}$ and $g_0 - M\sigma_y\sqrt{\frac{\lambda}{2-\lambda}}$ are the upper and lower control limit, M is the length of the control limits typically chosen as 3 [16] and σ_y is the standard deviation of the time series computed over the first evaluation interval. The EWMA detector signals a state change when the state function g_i reaches the upper (for the changes in the increasing direction) or the lower (for the changes in the decreasing direction) control limit. By increasing the number of elements n of the data set considered in the evaluation window, the EWMA detector tends to reduce the number of false detections, while a low n reduces the mean delay for detection but augments false detections. Consequently, the choice of the number of elements n determines the trade-off between false detections and time delay for signaling a relevant state change.

VI. Results

In this section, we evaluate the performance of the three change detectors by considering synthetic scenarios (Section VI-A) and realistic traces coming from the resource measures of a multi-tier Web-based system (Section VI-B).

For the sake of presentation, for each algorithm we consider some typical design parameters. Where applicable, the values for the baseline CUSUM and EWMA have been chosen to provide best performance in terms of false detections and mean delay time. The details for each algorithm are indicated below.

- **ARL0 CUSUM.** The ARL0 CUSUM is characterized by the target ARL_0^* which is then used to dynamically compute the threshold H^*. We set ARL_0^* with a high value in order to ensure a very small false detection rate. For the ARL0 CUSUM as all the other CUSUM based detectors we set $K = \frac{\Delta}{2}$, where Δ is the smallest shift to detect.
- **Baseline CUSUM.** We also consider the baseline implementation of the CUSUM algorithm with the recommended value of $H = 5\sigma$ as a threshold and $K = \frac{\Delta}{2}$.
- **EWMA based detectors.** The performance of the EWMA detectors depends on the choice of considered past values n. We consider two extreme values $n = 30$ and $n = 200$ and denote the respective detectors as EWMA_{30} and EWMA_{200}. EWMA_{30} is chosen to minimize the average detection delay at the cost of a potentially high number of false detection; EWMA_{200} is chosen to minimize the number of the false detections, although it likely that it comes at the cost of a possible high detection delay.

A. Synthetic time series

The time series of interest for this paper are characterized by a significant noise component that changes as a function of the system load. The noise intensity is denoted by the standard deviation of the time series in an interval of stability. High values of the noise tend to limit the efficacy of the state change detection algorithms, hence it is important to verify the robustness of the proposed detection models on time series characterized by different noise levels.

For this analysis, we initially consider a synthetic scenario that is characterized by a sudden increment of the time series values from a relatively low to a higher state [21]. The lower state is kept constant for 500 samples, then it is suddenly increased for 100 samples. The increase is followed by a similar decrease. From the basic scenario, we generate other synthetic scenarios by perturbing the initial profile through different levels of noise characterized by standard deviation values equal to 0.2, 0.5, 0.7 and 1 (which are normalized values with respect to the state shift) with no correlation among the noise components. The results of these perturbations are shown in the Figures 3. A qualitative comparison among these graphs evidences that the complexity of the time series increases as a function of the standard deviation. For example, in the synthetic profile with standard deviation 1 (Figure 3(d)) it is really difficult to distinguish the load change occurring at sample 500.

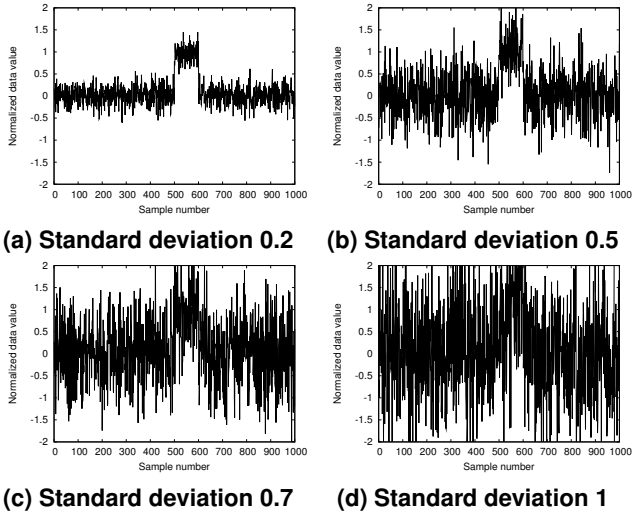


Fig. 3. Time serie profile.

Figure 4 and 5 illustrate the behaviour of the different algorithms. For the CUSUM we set Δ equal to size of the state jump (which has been normalized to 1 in Figure 3). In Figure 4, we plot the mean delay for detection for increasing values of the noise standard deviation. An expected trait of all the state change detection algorithms is the increase of the mean delay for detection, as a consequence of higher noise. All the algorithms but one are characterized by similar and low mean delay values. The EWMA_{200} model, instead, is characterized by significant higher delays. In particular, when the noise level is equal or above 0.8, the mean delay for detection is so high to the extent that its detections are useless

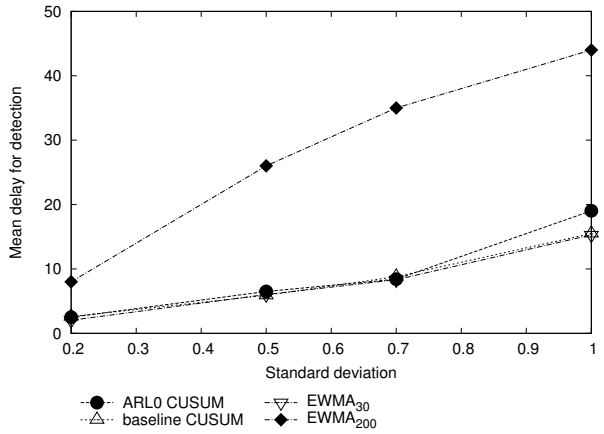


Fig. 4. Mean delay for detection.

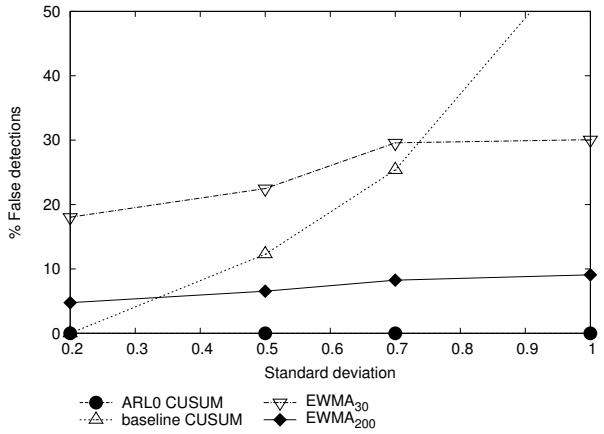


Fig. 5. Percentage of false detections.

for taking any decision at runtime.

In Figure 5, we report the fraction of false detection as a function of the noise standard deviation. This figure shows interesting differences among the state change detection algorithms. The results of the baseline CUSUM degrade rapidly as the noise variance increases. Consequently, baseline CUSUM is unacceptable in a realistic context characterized by non-stationary and variable noises. As usually, the performance of the EWMA depends on n. EWMA_{30} is much more reactive but it shows unacceptable percentages of false detections. ARL0 CUSUM outperforms the other algorithms because it consistently exhibits a negligible fraction of false detections. This is achieved by automatically adjusting the threshold H as to meet a target value of the run length ARL_0.

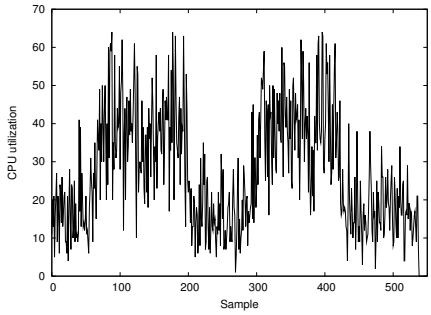
From all these results, we can conclude that for different levels of noises the mean delays for detection of the considered algorithms are similar, while the percentages of false detections show significant differences with a clear advantage of ARL0 CUSUM. The proposed algorithm is able to reduce the false detections without increasing the mean delay for detection. In the next section we aim to confirm these positive results for realistic time series coming from a multi-tier Web-based system.

B. Realistic time series

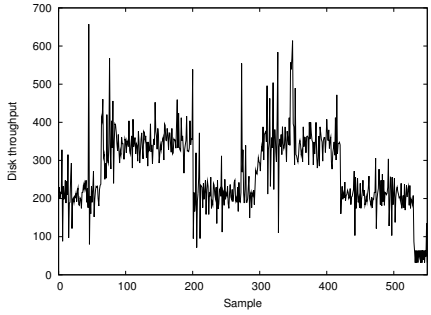
The architecture that we use as a testbed for the experiments is a typical multi-tier Web system that is based on the implementation presented in [22]. The application servers are deployed through the Tomcat servlet container, and are connected to MySQL database servers. In our experiments, we exercise the system through realistic traces; each experiment lasts for one hour.

We emulate more variable scenarios by means of the TPC-W workload generator [23]. Throughout the paper, we show the results reached through a workload scenario described by an alternating increase and decrease of the load between 400 and 700 emulated browsers. The first load change occurs after 70 samples; the other changes occur every 120 samples.

In order to determine which internal system resources are directly influenced by the external load variations, we evaluate the correlation between the internal and external system views. To this purpose, we measure the Pearson product-moment correlation coefficient [24], which is obtained by dividing the covariance of the two random data sets by the product of their standard deviations. We apply this correlation analysis to the main resources of the Web system nodes. The results indicate that the strongest correlation (characterized by a correlation coefficient > 0.8) exists between the external load and the CPU utilization of the database server, and the disk throughput of the application server. Hence, we use the measures correspondent to these two system resources as the representative internal views in the state change detection models.

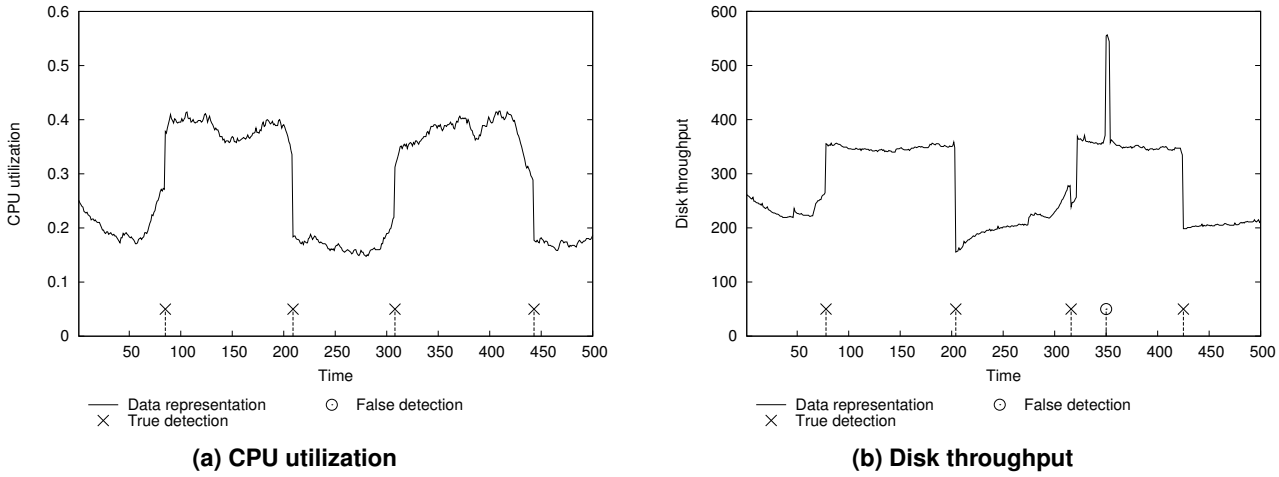


(a) CPU utilization on database server



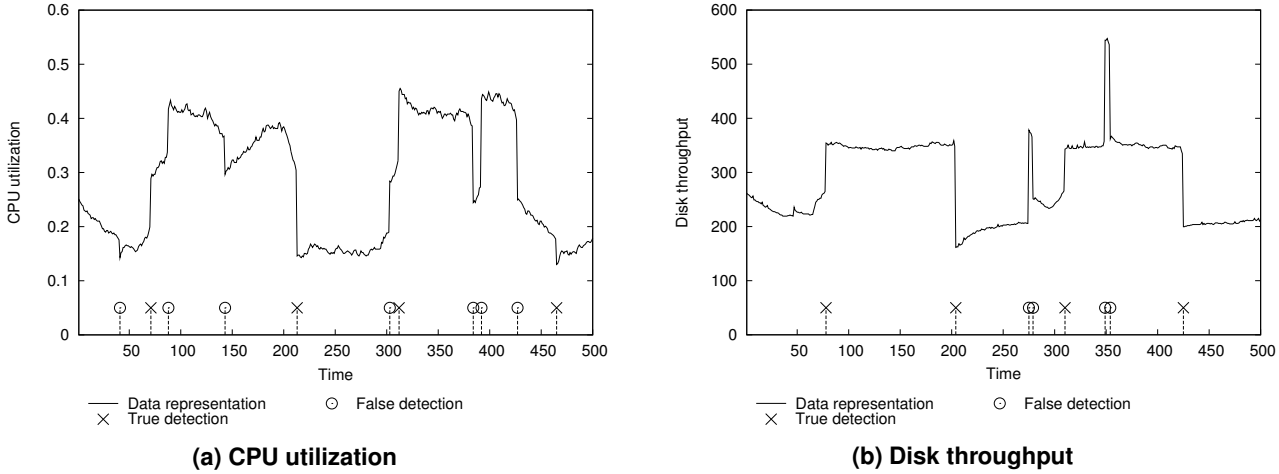
(b) Disk throughput on application server

Fig. 6. Resource metrics profiles



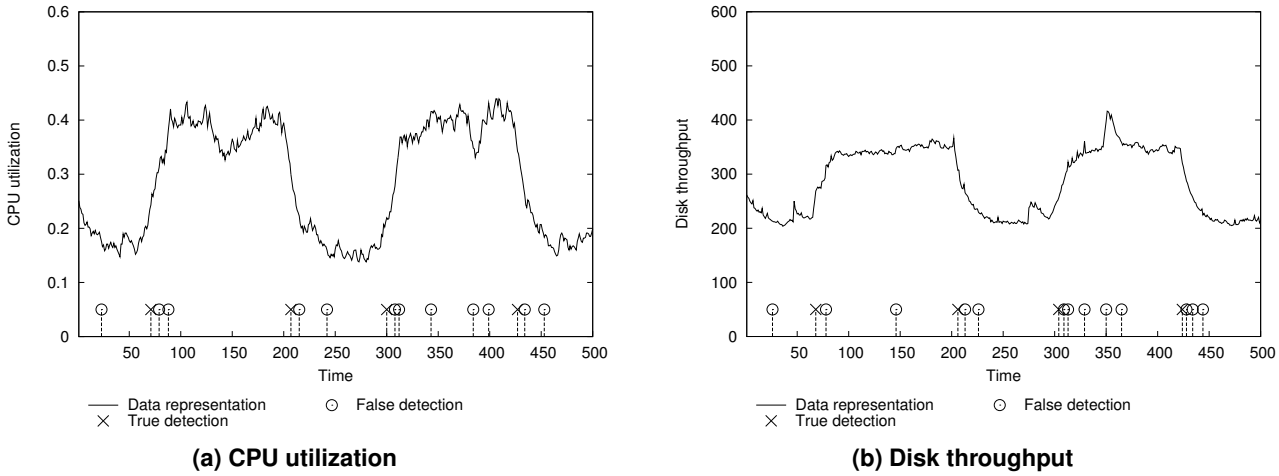
(a) CPU utilization **(b) Disk throughput**

Fig. 7. ARL0 CUSUM algorithm



(a) CPU utilization **(b) Disk throughput**

Fig. 8. Baseline CUSUM algorithm



(a) CPU utilization **(b) Disk throughput**

Fig. 9. EWMA_{30} algorithm

In Figure 6 we show the behavior of the time series generated by monitoring the CPU utilization of the database server and the disk throughput of the application server. The considered time series exhibit two different statistical behaviors: the time series in Figure 6 (a) is characterized by a high noise level uniformly distributed during the entire experiment; the time series in Figure 6 (b) shows that the disk throughput is subject to a lower noise component, but it is perturbed by some unpredictable spikes.

We should consider that each state change detection algorithm has a twofold role: to determine its own representation

of the original time series, and to detect the state changes on the basis of this representation. Figures 7, 8 and 9 show the results achieved by the different algorithms by taking into account both state representation and right/false state change detections. The small vertical lines on the bottom of the figures indicate detected load changes: a circle denotes a false detection, while a cross denotes a right load change detection. The six continuous lines denote the different state representations. The state representation is an important qualitative parameter to evaluate the change detection algorithms. An ideal representation should be characterized by a composition of vertical and horizontal segments, where an horizontal segment denotes a new stable state, and a vertical segment denotes a transition from a state to another.

In Figures 7 and 8, the state representations of both CUSUM-based algorithms evidence that they quickly adapt the representation to the new state conditions after a load change (vertical tendency). Besides that, the ARL0 CUSUM in Figures 7 is able to represent a stable state during stable conditions (horizontal tendency), while the baseline CUSUM in Figure 8 is unable to maintain a stable load representation. Figure 9 shows that the EWMA algorithm determines a state representation that is less reactive in following load changes and also less stable.

If we pass from a qualitative to a quantitative analysis that measures right/false state change detections, we can see that the ARL0 CUSUM algorithm in Figure 7(a) and (b) always provides the best results: it detects timely the state changes and it is affected by just one false detection in the disk throughput case (Figure 7(b)). Nevertheless, after the false spike detection in the representation, the ARL0 CUSUM algorithm is able to adapt quickly to the stable state. This behavior demonstrates the ability of this algorithm to recover immediately from a wrong detection.

The baseline CUSUM, shown in Figure 8, is affected by many false detections especially when the time series is characterized by the stable conditions. This is a consequence of its inability to maintain a stable load representation as the ARL0 CUSUM does.

The EWMA model detects several consecutive signals in correspondence of every load change. In Figures 9 we report one of the best results for the EWMA algorithm achieved in the considered testbed for $n = 30$. Nevertheless, these figures evidence that EWMA_{30} is affected by a high number of false detections after each load change. By increasing significantly the number n of considered past samples, the number of false detections decreases even if far from the ARL0 CUSUM results. Unfortunately, this reduction comes at the price of not acceptable (in real-time contexts) increases in the load change detection delays.

VII. Conclusions

Most runtime management decisions from fault detection to access control to any autonomic-related control rely on immediate detection of relevant and non-transient state changes in the system. We consider a scenario where system resources are constantly monitored and the obtained data are fed into statistical models that decide whether significant state changes occurred. This detection is a tough task when the behavior of system resources is highly variable and non-stationary, and the problems complexity is further augmented when the state change detection must be carried out at runtime.

We present a novel agile runtime detector that aims to address the trade-off between detection delay and the number of false detections. Our experiments demonstrate that the proposed CUSUM-aware detector is robust and effective because it signals immediately the relevant state changes with very low percentages of false detections. We have shown the quality of our algorithm for different time series profiles coming from a simulated context and from a realistic prototype of a Web-based system. All analyses demonstrate that the proposed state change detection algorithm is able to preserve the efficacy of detections for different levels of noise and magnitude of the state changes.

The presented study concerns single resources, but a system consists of multiple resources characterized by some dependencies. Hence, the most important extension of this paper concerns scenarios with multiple resources where the state change of a system must be detected on the basis of several data sets possibly characterized by different statistical properties.

Acknowledgment

We express our gratitude to the anonymous reviewers for their positive feedbacks and helpful comments.

References

[1] F. LeGland, C. Musso, and N. Oudjane, "An analysis of regularized interacting particle methods in nonlinear filtering," in *Proc. of the IEEE European Workshop on Computer-Intensive Methods in Control and Signal Processing*, 1998.

[2] P. Del Moral, "Non linear filtering: Interacting particle solution," *Markov Processes and Related Fields*, vol. 2, no. 4, 1996.

[3] E. Hartikainen and S. Ekelin, "Enhanced network-state estimation using change detection," in *Proc. of the 31st IEEE Conference on Local Computer Networks*, Nov. 2006.

[4] G. Kitigawa, "Monte carlo filter and smoother for non-gaussian nonlinear state models," *Journal of Computational and Graphical Statistics*, vol. 5, 1996.

[5] D. L. Alspach and H. W. Sorenson, "Nonlinear bayesian estimation using gaussian sum approximation," *IEEE Trans. Automat. Contr.*, vol. 20, 1972.

[6] F. Gustafsson, *Adaptive Filtering and Change Detection*. John Wiley and Sons, 2000.

[7] D. F. Allinger and S. K. Mitter, "New results in innovations problem for nonlinear filtering," *Stochastics*, vol. 4, 1991.

[8] M. Basseville and I. Nikiforov, *Detection of Abrupt Changes:Theory and Application*. Prentice-Hall, 1993.

[9] A. S. Willsky and H. L. Jones, "A generalized likelihood ratio approach to the detection and the estimation of jumps in linear systems," *IEEE Trans. on Data and Knowledge Engineering*, vol. 14, no. 2, May/Apr. 2002.

[10] E. S. Page, "Estimating the point of change in a continuous process," *Biometrika*, vol. 44, 1957.

[11] E. Hartikainen, S. Ekelin, and J. M. Karlsson, "Change detection and estimation for network-measurement applications," in *Proc. of the 2nd ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, Nov. 2007.

[12] F. Gustafsson, "The marginalized likelihood ratio test for detecting abrupt changes," *IEEE Trans. on Automatic Control*, vol. 1, no. 2, 1996.

[13] P. Ramanathan, "Overload management in real-time control applications using (m,k)-firm guarantee," *Performance Evaluation Review*, vol. 10, no. 6, Jun. 1999.

[14] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. M. Nahum, "Locality-aware request distribution in cluster-based network servers," in *Proc. of the 8th Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS 1998)*, San Jose, CA, Oct. 1998.

[15] R. Pandey and R. Barnes, J. F. Olsson, "Supporting quality of service in HTTP servers," in *Proc. of the ACM Symposium on Principles of Distributed Computing*, Puerto Vallarta, Mexico, Jun. 1998.

[16] D. C. Montgomery, *Introduction to Statistical Quality Control*. John Wiley and Sons, 2008.

[17] G. Moustakides, "Optimal stopping times for detecting changes in distribution," *The Annals of Statistics*, vol. 14, no. 4, 1986.

[18] D. Siegmund, *Sequential Analysis. Tests and Confidence Intervals*. New York: Springer, 1985.

[19] M. Kendall and J. Ord, *Time Series*. Oxford University Press, 1990.

[20] V. Jaconson, "Congestion avoidance and control," in *Proc. of SIGCOMM'88*, vol. 21, Stanford, CA, Aug. 1988.

[21] M. Satyanarayanan, D. Narayanan, J. Tilton, J. Flinn, and K. Walker, "Agile application-aware adaptation for mobility," in *Proc. of the 16th ACM Intl. Symposium on Operating Systems Principles (SOSP 1997)*, Saint-Malo, France, Oct. 1997.

[22] H. W. Cain, R. Rajwar, M. Marden, and M. H. Lipasti, "An architectural evaluation of Java TPC-W," in *Proc. of the 7th Symposium on High Performance Computer Architecture (HPCA2001)*, Monterrey, ME, Jan. 2001.

[23] "TPC-W transactional Web e-commerce benchmark," 2004, – http://www.tpc.org/tpcw/.

[24] J. F. Kenney and E. S. Keeping, *Mathematics of Statistics*. Van Nostrand, 1962.