Online Spike Detection in Cloud Workloads

Amardeep Mehta*, Jonas Dürango**, Johan Tordsson* and Erik Elmroth*

*Department of Computing Science, Umeå University, Sweden
*(amardeep, tordsson, elmroth)@cs.umu.se
**Department of Automatic Control, Lund University, Sweden
**jonas.durango@control.lth.se

***Abstract*—We investigate methods for detection of rapid workload increases (load spikes) for cloud workloads. Such rapid and unexpected workload spikes are a main cause for poor performance or even crashing applications as the allocated cloud resources become insufficient. To detect the spikes early is fundamental to perform corrective management actions, like allocating additional resources, before the spikes become large enough to cause problems. For this, we propose a number of methods for early spike detection, based on established techniques from adaptive signal processing. A comparative evaluation shows, for example, to what extent the different methods manage to detect the spikes, how early the detection is made, and how frequently they falsely report spikes.**

I. Introduction

In this paper we investigate the problem of online early detection of spikes in cloud workloads. Unanticipated changes in workload characteristics can potentially lead to service slowdown and might end in service-failure due to insufficient resource allocation. For this reason, having the ability to early detect build up of workload spikes is beneficial for making proactive resource management decisions.

Workload spike detection is complicated by the fact that no coherent definition of spikes exists: what appears to be a spike for one application might be considered normal workload for another application. This makes any attempt to characterize spikes harder. Spikes manifest themselves by sudden changes in a workloads statistical properties. This can include changing mean, variance and autocorrelation structure. The events responsible for generating spikes can either be internal to the application, or due to an exogenous influence. Their occurrence can be well-known in advance, such as the Olympic games, or completely unexpected events such as sensational news. In the current work, we focus on the early detection of spikes considered to be unanticipated in cloud workloads.

Detection and identification of spikes is tightly coupled to workload modeling. We adopt the view that spikes are to be seen as significant deviations in any aspect of a workload from a model-based prediction of said workload. Models might already be readily available for some workloads, or may be derived as part of designing spike detection algorithms. In this paper we employ the latter approach. The view that spikes are considered deviations from a model implies that the problem of detecting spikes is heavily application dependent: while a large sudden burst in traffic for one application could be considered a spike, if it occurs regularly due to some well-understood reason, it could probably be modeled and therefore not be considered a spike.

In this paper we consider online workload spike detection using adaptive signal processing techniques. We derive a number of workload models that are coupled with a stopping rule for detecting the onset of a spike and evaluate them using the well-known and publicly available FIFA 1998 World Cup workload [1], shown in Figure 1. Of special interest is the ability to detect the onset of a spike as early as possible, thus leaving enough time to mitigate the effects of it. We explore the tradeoff between the ability to correctly detect large traffic increases as spikes and how prone the detectors are to generate false alarms when no spike is present. Also explored is the tradeoff between how early detections can be made, and the number of false alarms generated. We conclude that these are tradeoffs that need to be carefully considered by an application owner, as for example some applications can withstand a fairly large fraction of undetected spikes, while it can only make use of a detection if it arrives well in time.

II. Related Work

Previous efforts have been made on the topic of modeling and characterizing workloads and spikes, see [2], [3], [1]. [3] classifies spikes into volume spikes (sudden increase in the total workload of a system) and data spikes (sudden increase in demand of a certain object). Spikes are characterized using

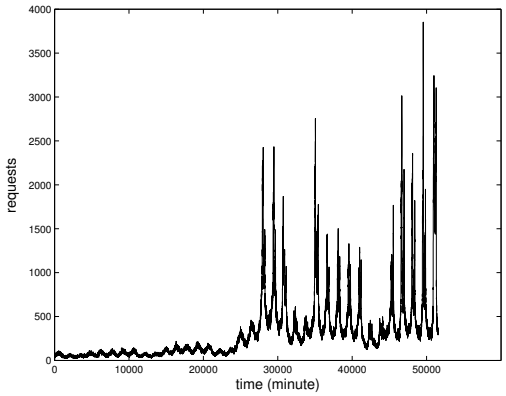


Fig. 1. Workload data from the FIFA 1998 World Cup.

seven statistical parameters, and modeled into four different phases: increase, plateau, decrease and normal. In our previous work, [4], we show how slow variations in the Wikipedia workload can be modeled using B-splines in order to extract the seasonality and trend in the workload. In [5], the authors employ an approach where exponential smoothing is used for workload prediction and hot spot detection.

In [6] the authors analyze the accuracy of different existing models for predicting workload spikes, continue by developing a new model, and finally present a novel metric for assessing the prediction accuracy of said model. [7] introduces the concept of characterizing burstiness in time series of workload arrivals using the index of dispersion. In [8], [9] the authors describe how the index of dispersion can be used to model and parametrize bursty workloads when investigating service times in multi-tier applications. [10] presents Fasttrack, a dynamic resource provisioning solution for multi-tiered applications that estimates the index of dispersion and utilizes it for determining when the workload is entering and exiting a bursty state.

III. Methods

For doing online spike detection in cloud workloads, we use in this paper a commonly employed technique in adaptive signal processing: combining a model of the system or signal of interest, here the workload, with a stopping rule that signals whenever it detects that a system change has occurred. In particular, we employ the one-model approach described in [11]. In this approach, the workload measurements y_t, $t = 0, 1, \ldots$, are compared to the one-step ahead predicted output $\hat{y}_{t|t-1}$ of a model that has been derived. If the model correctly captures the dynamics of the workload, taking the difference between the workload and prediction will result in the residuals

$$\varepsilon_t = y_t - \hat{y}_{t|t-1} \tag{1}$$

looking like a white noise sequence, i.e. as a sequence of independent, identically distributed (often Gaussian) random variables. If, at any point in time, an event takes place that changes the system in such a manner that the model is no longer able to correctly describe the system behavior, this will show up as a change in the characteristics of the residuals (changing mean, variance, autocorrelation, etc.). In accordance with [11], the residuals are used for deriving a so-called distance measure $s_t = f(\varepsilon_t)$, which is then used in conjunction with a stopping rule for detecting when a change has taken place. The particular choice of distance measure depends on the type of change that we are interested in detecting. Some common choices include using the residuals directly, i.e. $s_t = \varepsilon_t$, which is useful for detecting changing means, and the squared residuals, $s_t = \varepsilon_t^2$, which is useful for detecting changes in variance.

The rest of this section is devoted to describing the stopping rule and the adaptive filter models used in this paper. Also described is a time aggregation technique of the workload data that two of the proposed approaches make use of.

A. Stopping rule - CUSUM test

The idea behind using stopping rules for change detection is simple: by monitoring critical aspects of interest of the system under investigation, we sound an alarm when these aspects exceed a threshold. Here we have adopted the well-known CUSUM test [12], which can be considered a special case of the sequential probability ratio test [13]. In the CUSUM test, the distance measure sequence s_t is cumulatively summed up (hence the name CUSUM – cumulative sum) to calculate the test statistic g_t until the sum hits a preset threshold h, at which time point an alarm is raised and the sum is reset to zero. During the calculation, g_t is bounded from below, so if at any time point $g_t < 0$, it will also then be reset to zero. If the distance measure s_t used is the residuals ε_t themselves, and the workload model used is correct so that the residuals become a white noise sequence, g_t will undergo a random walk. This will lead to g_t crossing the threshold h spuriously even though no change has taken place, leading to false alarms. For this reason, a drift term ν is used in the calculation of g_t to slowly push it back to zero. Algorithm 1 outlines pseudo code for the CUSUM test used in the paper. Both the threshold h and drift term ν are concerned with determining the speed and robustness of the detector. For example, a careful choice of ν should ensure that g_t is kept small enough to not cross the threshold randomly, but still allow for g_t to cross the threshold when a change has taken place. On the other hand, a large robustness to spurious alarms can come at the cost of relative large delays before a detection is made. This tradeoff is problem-specific and in Section IV, we explore it for a particular workload.

To avoid too frequent alarming, we implement a hanging window approach. When an alarm is fired, any subsequent detections during the duration of the hanging window length are ignored. When the hanging window expires, the next detection can trigger a new alarm. Another possible solution that is often employed to avoid too frequent alarming is to raise an alarm only if two or more detections during a short time period are made.

Algorithm 1 CUSUM test

```
Input: drift ν, threshold h
Output : alarm times
g_0 ← 0
for t = 1 to T do
    g_t ← max(g_{t-1} + s_t − ν, 0)
    if g_t > h then
        g_t ← 0
        alarm ← 1
```

B. Time aggregation of workload data

For modeling purposes, choosing a high enough sampling rate for a signal is crucial for fully capturing the dynamics of the workload. On the other hand, a high sampling rate comes with the drawback of increased noise levels and the need for increased model complexity, e.g. in order to describe

workload pattern variations on an hourly basis we require a higher model complexity if new measurements arrive every second, as opposed to every ten minutes.

In order to keep noise levels and model complexity low while still making use of a higher sampling rate, we propose an approach where we, apart from using the actual workload measurements y_t also construct the more coarsely-granulated measurement sequence $\{y_k^H\}$ by aggregating the measurements in bins of size t_s:

$$y_k^H = \sum_{i=s}^{k \cdot t_s} y_i \tag{2}$$

where $s = (k-1) \cdot t_s + 1$. From this new sequence we derive fairly simple models that can capture the general workload dynamics. In Figure 2 we show the effect of aggregating measurements on a more coarsely-granulated timescale.

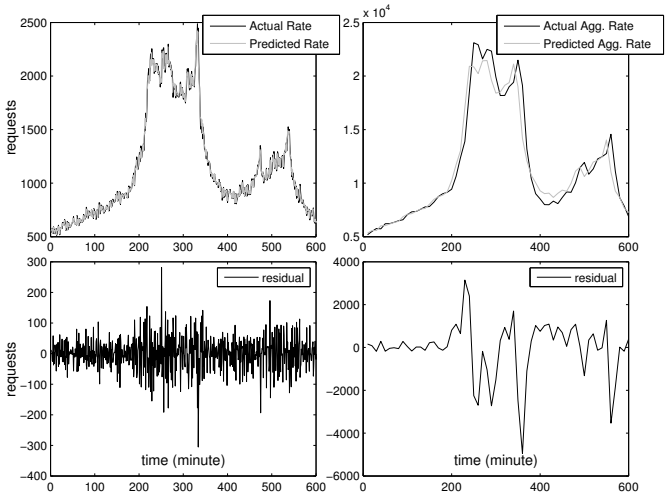


Fig. 2. The effect of aggregating workload measurements on a more coarsely-granulated timescale. *Left:* the original measurement sequence and residuals when calculating one-step ahead prediction using an AR model. *Right:* same as left, but for data aggregated on a more coarsely-grained timescale. It is apparent that the effect on the residuals from the spike is more prominent in the latter case.

Algorithm 2 Modified CUSUM test

Input: drift ν, threshold h
Output : alarm times
$g_{prev} \leftarrow 0$
for $k = 1$ to K **do**
  $g_0 \leftarrow 0$
  for $t = 1$ to t_s **do**
    $g_t \leftarrow max(g_{t-1} + s_t + g_{prev} - \nu, 0)$
    $g_{prev} \leftarrow 0$
    if $g_t \geq h$ **then**
      $g_t \leftarrow 0$
      $alarm \leftarrow 1$
    else if $t = t_s$ **then**
      $g_{prev} \leftarrow g_t$

C. Adaptive filter models

What different workload modeling techniques should have in common is the ability to provide a model that encompasses the dominating features of the workload while being agnostic to features that are not considered important for the task at hand. Here we present a number of workload models based on adaptive filtering methods.

1) Auto-regressive modeling using time aggregation (*AR-TA)*: in the first approach, we make use of the time aggregation technique described in section III-B and estimate an auto regressive (AR) model of order p using the coarsely-granulated data. This gives a model on the form:

$$y_k^H = \phi_0 + \sum_{i=1}^{p} \phi_i y_{k-i}^H + e_k \tag{3}$$

where e_t is white noise. The parameters $\phi = [\phi_0, \ldots, \phi_p]$ are estimated using ordinary least squares over a sliding time window of size wl and re-estimated every time step in the coarsely-granulated timescale. The one-step ahead prediction of the workload using this model is:

$$\hat{y}_{k|k-1}^H = \phi_0 + \sum_{i=1}^{p} \phi_i y_{k-i}^H \tag{4}$$

The one-step ahead prediction on the coarsely-granulated timescale is then up sampled to the finer timescale at which workload measurements arrive using linear interpolation. The combination of two time scales allows us to predict, using the coarsely-granulated timescale, the general direction in which the workload is heading and compare that to the behavior of the workload on the finer timescale. If a spike is building up, the incoming workload measurements will start deviating from the prediction. The modified CUSUM test outlined in Algorithm 2 is used for detection purposes. In Section IV we investigate the sensitivity of this approach to particular choices of the window length wl and time aggregation level t_s.

2) Double exponential smoothing using time aggregation (*DS-TA)*: in the second approach, we combine the time aggregation technique with a double exponential smoothing model for computing the one-step ahead prediction. The model assigns exponentially decreasing weights over time to the measurements and takes into account the trend of the workload. The smoothed data S_k and trend b_k for the coarsely-granulated measurements y_k^H at time instant $k > 1$ is estimated as

$$S_k = \alpha y_k^H + (1-\alpha)(S_{k-1} + b_{k-1}) \tag{5}$$

$$b_k = \beta(S_k - S_{k-1}) + (1-\beta) b_{k-1} \tag{6}$$

where α is a data smoothing factor, $0 < \alpha < 1$, and β is the trend smoothing factor, $0 < \beta < 1$. We initialize $S_1 = y_1^H$ and $b_1 = y_1^H - y_0^H$.

The one-step ahead prediction is given by

$$\hat{y}_{k|k-1}^H = S_{k-1} + b_{k-1} \tag{7}$$

As for AR-TA, the one-step ahead prediction is up sampled to the finer-granulated timescale using linear interpolation. The

procedure for detecting spikes is also identical to that for ARTA.

3) Low-pass filtered differentiation (*Diff*): the next approach implicitly assumes that the workload will change fairly slowly. By differentiating the workload measurements, we can estimate the rate of change in the workload. During normal traffic, the rate of change will be distributed around zero. In contrast, during a spike we expect the mean rate of change to transiently increase to a positive number, a fact that is exploited for spike detection. Since differentiating measurements often gives a noisy result, we improve the signal-to-noise ratio by low-pass filtering the differentiated signal using a second order Butterworth filter before feeding it to the CUSUM detector. This allows us to more robustly detect when the rate of change is deviating from zero.

4) Constant mean (*CM*): the final approach takes a fairly agnostic view of the workload by using a model that simply assumes a constant workload mean:

$$\hat{y}_t = \theta + e_t \tag{8}$$

where θ is the workload mean and e_t white noise. The one-step ahead prediction then simply becomes

$$\hat{y}_{t|t-1} = \theta \tag{9}$$

The mean θ is estimated online using recursive least squares as new measurements become available. In order to still allow for some slow variation in the mean when computing the estimate, a forgetting factor λ that puts exponentially decreasing weight on old data is used. During a spike, the measurements will significantly deviate from the estimated long term mean, leading to the residuals $\varepsilon_t = y_t - \hat{y}_{t|t-1}$ no longer being distributed around zero. The residuals are then used in the CUSUM test to detect spikes.

In addition to the approaches described here, we consider also the approach in [10] where the *index of dispersion*, commonly used in networking for describing burstiness, is used for detecting the onset and end of a workload burst. Adopting their notation, the index of dispersion for a stationary process $\{X_n\}$ is defined as

$$I = SCV\left(1 + 2\sum_{k=0}^{\infty}\rho_k\right) \tag{10}$$

where SCV is the squared coefficient of variation, and ρ_k the lag-k autocorrelation coefficients. Estimation of the index of dispersion can be carried out in different ways. In contrast to the approach in [10], we perform the estimation using the fact that for a stationary time series $\{X_n\}$ it holds that the index of dispersion can be put as

$$I = \frac{Var(X)}{E(X)} \tag{11}$$

i.e. the ratio of the variance of the stochastic process to its mean. The estimation is then carried out by using measurements of the workload during a sliding window of size j so that the estimate becomes:

$$I_t = \frac{\hat{\sigma}_j^2}{\hat{\mu}_j} \tag{12}$$

where $\hat{\sigma}_j^2$ and $\hat{\mu}_j$ denote the estimated variance and mean of the workload using the measurements $\{X_t, \ldots, X_{t-j}\}$ in the sliding window. In neuroscience Equation (12) is often referred to as the *Fano factor (FF)*, and is used for describing burstiness in neural spike trains. For detecting spikes the estimate of the index of dispersion is then paired with the algorithm in Figure 2 in [10].

IV. Evaluation

To investigate the properties of the detectors described in Section III we evaluate them using the well-known FIFA World Cup 1998 workload, which consists of counts of requests made to the web servers serving the official website during the lead-up to and the final stages of the tournament. The sampling rate of the workload is one minute. During data preprocessing, we manually identify $N_s = 19$ spikes throughout the workload duration. For each spike, we manually record a starting time, ending time and duration of the spike. The starting points in time are not picked according to a strict definition, but rather to cover the complete rising phase of all spikes. The ending times are consistently chosen to coincide with the peak of each spike.

During evaluation we run the detectors online as if measurements arrive sequentially. Tuning is done offline, representing us having some a priori knowledge of the workload characteristics. A set of different metrics appropriate for describing the performance of the detectors are calculated a posteriori. If an alarm is raised during one of the manually identified spike intervals, we count that as a hit (true positive). An alarm raised outside of the identified spike intervals is counted as a miss (false positive). Note that this implies that alarms raised during the decreasing phase of a spike are counted as misses, which is natural since an alarm that late into a spike will probably be useless from a proactive resource management point of view. As is commonly done in pattern recognition, we calculate the so-called *precision* and *recall* metrics, defined as

$$precision = 100 \times \frac{\#\text{hits}}{\#\text{hits} + \#\text{misses}} \tag{13}$$

$$recall = 100 \times \frac{\#\text{hits}}{N_s} \tag{14}$$

Using these metrics we can also calculate the *F-score* defined as the harmonic mean of recall and precision:

$$F = 2 \times \frac{precision \times recall}{precision + recall}$$

Additionally, in order to capture the performance of the detectors in terms of how early or late they detect a spike when a hit is generated we define for each detector the *average time before peak* (ATBP) as the average time interval between the time point at which a hit is generated and the time point at which the peak of the corresponding spike occurs. We

also calculate the difference between the peak workload level during each correctly detected spike and the workload level at the time of detection, and define the *average relative change* (ARC) as the average ratio between said difference and the peak workload level.

As also mentioned in Section III, we make use of a hanging window to avoid too frequent alarming. We set this parameter to 205 minutes, which coincides with the average length of the manually identified spikes.

The rest of this section is organized as follows: firstly we explore the sensitivity of the proposed approaches with respect to different choices of the parameters available for tuning. Secondly we investigate the performance of the detectors when tuning them for a specific case.

A. Sensitivity and model parametrization

Here we present the results of an investigation into the sensitivity in performance of the AR-TA detector for varying parameters. For the sake of brevity we restrict ourselves to this single case. Further investigations reveal that the results here carry over also to the other proposed approaches.

First we investigate the detector behavior when varying the time aggregation parameter t_s and the size wl of the sliding window used for parameter estimation. For this we fix the order, threshold and drift parameters to $p = 2$, $h = 7300$ and $\nu = 100$. Note that among the other detectors considered, this part of the analysis only applies to DS-TA as it is the only one apart from AR-TA that makes use of the time aggregation approach. In Figure 3 we show plots for the resulting recall level, number of misses, F-score and ATBP. From this we conclude that the results are insensitive to variations in window size wl. For variations in time aggregation t_s we note that the recall level and F-score stay fairly constant, while there is a general tendency for the number of misses and ATBP to increase with increasing t_s. This co-variation between the number of misses and ATBP is repeatedly found throughout our experimental investigations of different detectors.

Next we fix the order, time aggregation and window size parameters to $p = 2$, $wl = 10$ and $t_s = 20$ and vary the detector parameters h and ν. The result of this experiment can be seen in Figure 4. Here the tradeoffs between recall, number of misses and how early spikes can be detected become clear. Choosing low values of h and ν gives a high recall, while the precision will suffer due to the large number of misses and vice versa, choosing very large parameter values leads to a low recall level and few misses. Again we observe the co-variation between the number of misses and ATBP. The F-score exhibits a tendency to increase with increasing parameter values.

The investigation presented here can form the basis of a procedure for choosing appropriate parameters when designing a detector for a workload. A first candidate set of parameters can be identified by considering only parameters that yield an F-score exceeding some specific level. From there we consider the specific numbers for recall, number of misses and ATBP. In the present case, Figure 4 reveal a band of parameters where ATBP and the number of misses vary greatly while the recall is constantly high. This leaves us with having to take into consideration the tradeoff between the number of misses and ATBP for the final particular choice of parameters.

B. Performance evaluation

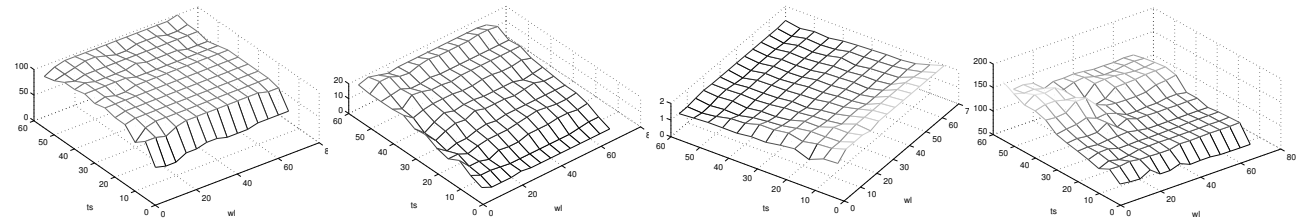
Next we investigate the performance of the detectors described in Section III. As was made clear in Section IV-A, there is generally a tradeoff between a detectors recall level and how prone it is to generate misses, at the same time there is a tradeoff between the number of misses and the ATBP.

To evaluate the performance we use a scenario where we find for each detector a set of parameters that correctly detects all manually identified spikes while also generating a fairly low number of misses (in this case 5). The parameters are found by using a similar approach as for the sensitivity analysis: we first fix the CUSUM parameters and manually identify appropriate detector-specific parameters. These parameters are α, β (DS-TA), cutoff frequency f for the low-pass filter (Diff), and the forgetting factor λ (CM). Next h and ν are found using the procedure outlined previously. Table IV-B show the parameters found for the different detectors, along with the performance attained for that choice of parameters. The adaptive filter based detectors behave similarly for this case, with AT-TA providing the highest ATBP and CM the highest ARC. We conclude that for this particular scenario, we can design detectors that fulfill the scenario goal while also providing a decent ATBP. Note that the FF approach lacks any tunable parameter apart from the window size j. This is the reason we cannot not match its performance with the other approaches. It is therefore not unexpected that it provides the worst performance in terms of recall and precision in this case. Interestingly enough it has the highest number for both ATBP and ARC. We attribute this to its trigger happy nature.

Finally, in Figure 5 we show for a part of the workload covering a spike the result of running the detectors using the parameters described here and illustrate hits and misses for the different detectors. The plot clearly shows how the FF detects the spike early but is also prone to generate misses, whereas the proposed methods detect the spike roughly at the same time, while in some cases also generate some misses.

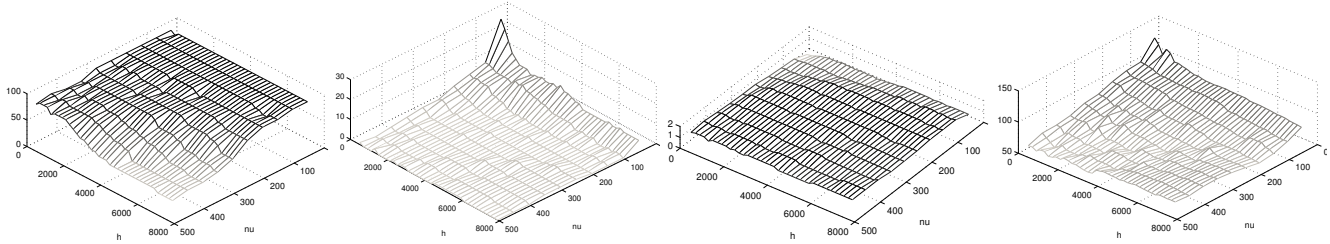
Method	Recall	Precision	ATBP	ARC	Input Parameters	
AR-TA	100	79.2	109.2	33.68	$t_s = 25$ $h = 7300$	$wl = 10$ $\nu = 100$
DS-TA	100	76.0	91.6	34.27	$\alpha = 0.2$ $h = 3000$ $t_s = 15$	$\beta = 0.1$ $\nu = 100$
Diff	100	79.2	99	40.47	$f = 0.02$ $h = 40$	 $\nu = 5$
CM	100	79.2	101.9	44	$\lambda = 0.95$ $h = 240$	 $\nu = 150$
FF	89.47	31.12	133.4	51.5	$j = 10$	

TABLE I
RESULTS FOR PERFORMANCE ANALYSIS



(a) Recall (b) Misses (c) F-score (d) ATBP

Fig. 3. Result for AR-TA when varying wl and t_s while h and ν are fixed.



(a) Recall (b) Misses (c) F-score (d) ATBP

Fig. 4. Result for AR-TA when varying h and ν while wl and t_s are fixed.

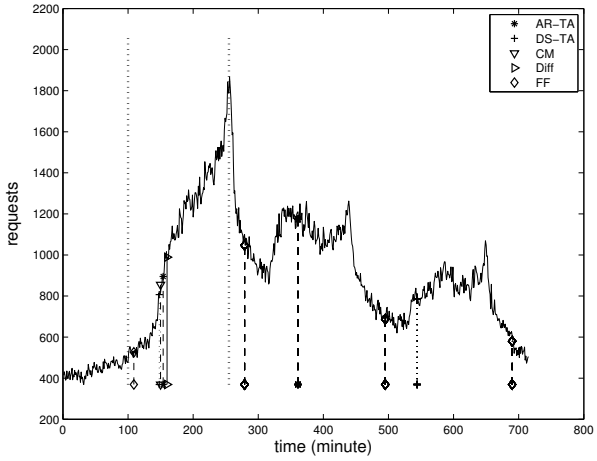


Fig. 5. Hits and misses for the different detectors evaluated during and after one of the spikes in the workload. Wide dotted vertical lines indicate the manually identified spike interval. Alarms generated in the interval are considered hits, while if outside as misses.

V. Conclusion

We derive and present detectors for cloud workload spikes for online use. Our solutions rely on a combination of workload modeling using adaptive filtering techniques and the CUSUM test. After evaluating our detectors on the FIFA 1998 World Cup workload we conclude that the proposed solutions are viable for spike detection. Our investigation reveals tradeoffs between being able to reliably detect all spikes and generating false alarms, and between increased detection time before each spikes peak and false alarms. These tradeoffs must be considered when designing spike detectors for cloud workloads. Future work will reveal the applicability of the proposed approaches to other scenarios and workloads.

Acknowledgment

The project is supported by Swedish VR grant for Cloud Control project, and through the LCCC Linnaeus and ELLIIT Excellence Centers.

References

[1] M. Arlitt and T. Jin, "A workload characterization study of the 1998 world cup web site," *Netwrk. Mag. of Global Internetwkg.*, vol. 14, no. 3, pp. 30–37, May 2000.

[2] A. B. Downey and D. G. Feitelson, "The elusive goal of workload characterization," *SIGMETRICS Perform. Eval. Rev.*, vol. 26, no. 4, pp. 14–29, 1999.

[3] P. Bodik, A. Fox, M. J. Franklin, M. I. Jordan, and D. A. Patterson, "Characterizing, modeling, and generating workload spikes for stateful services," in *SoCC*, 2010.

[4] A. A. Eldin, A. Rezaie, A. Mehta, S. Razroev, S. S. d. Luna, O. Seleznjev, J. Tordsson, and E. Elmroth, "How will your workload look like in 6 years? analyzing wikimedia's workload," in *IC2E*, 2014.

[5] P. Saripalli, G. V. R. Kiran, R. R. Shankar, H. Narware, and N. Bindal, "Load prediction and hot spot detection models for autonomic cloud computing," in *UCC*, 2011.

[6] M. Lassnig, T. Fahringer, V. Garonne, A. Molfetas, and M. Branco, "Identification, modelling and prediction of non-periodic bursts in workloads," in *CCGRID*, 2010.

[7] R. Gusella, "Characterizing the variability of arrival processes with indexes of dispersion," *Selected Areas in Communications, IEEE Journal on*, vol. 9, no. 2, pp. 203–211, 1991.

[8] G. Casale, N. Mi, L. Cherkasova, and E. Smirni, "How to parameterize models with bursty workloads," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, 2008.

[9] N. Mi, G. Casale, L. Cherkasova, and E. Smirni, "Burstiness in multi-tier applications: Symptoms, causes, and new models," in *Middleware*, 2008.

[10] A. Caniff, L. Lu, N. Mi, L. Cherkasova, and E. Smirni, "Fastrack for taming burstiness and saving power in multi-tiered systems," in *ITC*, 2010.

[11] F. Gustafsson, *Adaptive Filtering and Change Detection*. Wiley, 2000.

[12] E. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 110–115, 1954.

[13] A. Wald, "Sequential tests of statistical hypotheses," *Ann. Math. Statist.*, vol. 16, no. 2, pp. 117–186, 1945.