Runtime prediction models for Web-based system resources

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

Abstract

Several activities of Web-based architectures are managed by algorithms that take runtime decisions on the basis of continuous information about the state of the internal system resources. The problem is that in this extremely dynamic context the observed data points are characterized by high variability, dispersion and noise at different time scales to the extent that existing models cannot guarantee accurate predictions at runtime. In this paper, we evaluate the predictability of the internal resource state and point out the necessity to filter the noise of raw data measures. We then verify that more accurate prediction models are required which take into account the non stationary effects of the data sets, the time series trends and the runtime constraints. To these purposes, we propose a new prediction model, called trend-aware regression. It is specifically designed to deal with on the fly and short-term forecast of time series which originate from filtered data points belonging to internal resources of Web system. The experiment evaluation for different workload scenarios shows that the proposed trend-aware regression model improves the prediction accuracy with respect to popular algorithms based on auto-regressive and linear models, while satisfying the computational constraints of runtime prediction.

1 Introduction

The increasing complexity of Web-based systems requires accurate models that can support management and coordination activities of the internal components, where several algorithms take decisions by getting continuous information about the state of the internal system resources. This paper represents a step further in the direction of investigating and predicting the performance of internal resources of Web-based architectures in a short-term horizon which is crucial for taking several runtime decisions on load and resource management (e.g., load balancing, load sharing, admission control, job dispatching). We observe that in the considered Web context any user click causes a large number of arrivals at an open system. These arrivals activate a possibly huge number of requests to the internal hardware and software resources with unpredictable mutual interactions and interdependency. As a consequence, it is becoming impossible to understand, for example, where the system load is and where the system is going from monitored raw data, such as snapshots of the CPU utilization, request arrivals, average number of queued processes.

We are interested to the models that can predict at runtime the future state of system resources in a short-term horizon that is sufficient to take more "conscious" management decisions. We first analyze the predictability of some typical performance indexes concerning the servers of Web-based systems, and we show that existing predictive models are not satisfactory because of limited accuracy or excessive computational time. For this reason, we propose a new prediction algorithm (*Trend-Aware Regression* algorithm or TAR) that shows best precision for different scenarios and stable results for a wide range of parameters.

There are several external and internal system factors which make this problem interesting, difficult and not deeply investigated. *External* factors are well known: the traffic reaching Internet-based services is characterized by heavy-tailed distributions [3, 8], bursty arrivals [13] and hot spots [4] that are difficult to predict and manage because the server side has no control on the client side. *Internal* factors have received less attention. All opinions agree to indicate that the complexity of the middleware and of the systems for the support of Web-based applications is increasing, but there is not yet an adequate characterization of the internal load that is the focus of this paper.

On one hand, commitment to high availability and to short-time services requires runtime models that are able to evaluate and, when possible, predict the load conditions of the system components. On the other hand, the observed data points coming from system monitors are characterized by high variability, dispersion and noise, to the extent that existing models cannot guarantee an accurate time series forecast at runtime. A first positive result of this paper is

that the raw data points of the considered resources are characterized by a high frequency component (*noise*) that perturbs the measures and contributes to their unpredictability. We reduce the noise component through a runtime smoothing process that allows us to obtain a low frequency component (*filtered data*) exhibiting some short-term dependency. This process opens the possibility of predicting the state of the internal resources in terms of filtered performance indexes. Nevertheless, the most popular predictive models (e.g., Auto Regressive, ARIMA, Exponential Weighted Moving Average, Linear Regression), when adopted at runtime in an extremely dynamic context, show a limited precision or excessive delay even in a short-term horizon. Unlike the existing models, the proposed TAR algorithm is able to forecast on-the-fly the future resource state with a better accuracy, because it considers for prediction both the filtered data set and its behavioral *trend*.

The rest of the paper is organized as follows. Section 2 discusses the motivations of this research and analyzes the statistical characteristics of the raw data points coming from system monitors. Section 3 selects the models that can be adopted as runtime predictors and introduces the proposed TAR algorithm. Section 4 evaluates the quality of the considered prediction models and their stability to different parameters. Section 5 presents related work and compares main contributions with the state of the art. Section 6 concludes the paper with some final remarks.

2 Statistical analysis

Management and coordination of Web-based servers are usually carried out through several algorithms that take on-the-fly decisions on the basis of continuous information related to the state of the internal resource components. In order to take adequate management decisions at runtime, it is necessary to study new models and algorithms that are able to deal with the state of the internal system resources for evaluation and prediction purposes.

Any prediction is based on some historical information that can be considered as a time series, that is, an ordered collection of n data, S_i, beginning at time t_{i-n-1} and covering events up to a final time t_i. Specifically, we have a time series $S_i = (s_{i-n-1}, \ldots, s_i)$, where the *j-th* element, $i-n-1 \leq j \leq i$, is a pair $s_j = (t_j, v_j)$. The first element t_j indicates its time of occurrence, whereas the second element v_j of the pair denotes the value of one variable of interest. The elements of S_i are time ordered, that is, $t_j \leq t_x$ for any $j < x$.

An important premise is that no prediction model can work if the analyzed time series does not exhibit some predictability factors. The data set has to show some temporal dependency, otherwise any prediction is infeasible. The *auto-correlation analysis* on the time series allows us to show the presence or not of some time dependence and to distinguish between the possibility of achieving long-term or just short-term prediction. In [2], we have carried out a large set of experiments in a multi-tier Web system. All results gave the same clear message: independently of the considered system resource (e.g., CPU, disk) and type of server (e.g., http, application, back-end), the auto-correlation functions of the data sets related to the internal monitors decay steeply. In these conditions, it is impossible for any prediction model to achieve some accurate results even in a short-term horizon. Now, the main question is whether some noise affects or not the observed data points. To this purpose, we apply to the entire observed data set an offline filter that returns a noiseless representation, and we evaluate the noise perturbation as the difference between offline filtered and observed values.

We consider an offline Kalman filter that takes as its input the sequence of the observed data sets $(t_i, \bar{v_i})$ and gives a sequence of $s_j^* = (t_j, v_j^*)$ where v_j^* denotes the filtered value, that is:

$$\begin{cases} v_i^* = v_{i-1}^* + K_i(\bar{v_i} - v_{i-1}^*) \\ K_i = \frac{P_{i-1}}{P_{i-1}+R_e} \\ P_i = (1-K_i)P_{i-1} \end{cases} \quad (1)$$

where R_e is the variance of the noise e_t that perturbs the original signal values $\bar{v_i}$, K_i is the Kalman gain, and P_i is the covariance error [5]. We estimate the noise perturbation δ as the mean value of the N residuals between the observed values $\bar{v_i}$ and the values v_i^* that are generated by the *offline* filter:

$$\delta = \frac{1}{N} \sum \frac{v_i^* - \bar{v_i}}{v_i^*} \quad (2)$$

The first line in Table 1 reports the results of the noise analysis for the utilization of three system resources of the application and back-end server. A data set is considered to be affected by a significant noise when $\delta > 0.3$, hence the results in the first line of Table 1 confirm that a high noise component perturbs all the observed data sets ($\delta \gg 0.3$). We can conclude that, unlike external contexts characterized by predictable time series (e.g., [1, 4]), when we consider the internal resources of Web system, no runtime prediction model can forecast the future state by taking as its input the observed data points. However, the presence of noise is a positive result because it allows us to apply some *online* filters. For example, if we apply an online filter based on moving average, then we get a filtered data set that becomes predictable: in the second line of Table 1 we report the noises of the filtered data set that are well below $\delta < 0.3$. The possibility of achieving a predictable time series is confirmed by the auto-correlation function of the filtered data set that has a slow decay on a time scale of 150 seconds. As in the context of Web systems, a temporal dependence in the

range of minutes is considered a valid basis for short-term prediction [20], we expect that filtered data sets represent a feasible way for prediction. The statistical analysis concerned the CPU, disk and network of the application and database servers but for space reasons hereafter we focus on the CPU of the database server that is among the most representative performance index of the system behavior for the considered workloads.

Table 1. Noise analysis on resource utilization

	Application server			Database server		
	CPU	**Disk**	**Net**	**CPU**	**Disk**	**Net**
raw data	0.78	0.36	0.65	0.98	1.37	0.87
filtered data	0.16	0.08	0.07	0.18	0.27	0.10

3 Prediction models

3.1 Requirements and algorithms

The choice of the most appropriate model depends on the characteristics of the context. If we consider our focus on runtime and short-term prediction, we are interested to the following properties: the *prediction accuracy* denotes the precision of the model in representing the future data set points, including its capacity of self-adapting to load variations when conditions are unstable; the *prediction delay* measures the time required by a predictor to follow significant load variations; the *computational cost* of the algorithm implementing a prediction model is a driving factor for accepting it in a runtime prediction context.

The rationale behind these attributes is that we want an accurate prediction, but we are also interested to the execution time required to obtain it. This feature is even more critical in a short-term prediction context where the horizon is represented by few minutes. In this paper, we consider four popular classes of linear prediction models that can be applied to runtime contexts.

- Exponential Weighted Moving Average (EWMA), [14].
- Auto-Regressive Model (AR), [11].
- Auto-Regressive Integrated Moving Average Model (ARIMA), [11].
- Linear Regression (LR), [4].

Other prediction models (e.g., genetic algorithms, neural networks, Support Vector Machines and Fuzzy systems) are intrinsically designed for off-line applications and medium-long term predictions, and are rather inadequate to support runtime decision systems [10] in highly variable contexts.

Indeed even the AR and ARIMA prediction models require a careful choice of the model parameters, that is typically based on the evaluation of the auto-correlation and partial auto-correlation functions [6]. In this paper, we evaluate the parameters on the basis of the initial 10% values of the data set of the entire experiment.

3.2 Trend-Aware Regression model

The data points deriving from the monitored resources of the Web systems are characterized by non stationary effects. In this context, existing prediction models satisfy the computational constraint, but their prediction quality may show some limits that we quantify in Section 4. Hence, it is important to define a new prediction algorithm that is able to limit the drawbacks of the existing models in the specific context of internal resources of Web systems. We think that the complexity of this novel scenario requires a *meta-model* that can improve the prediction quality by gathering the best properties from the other prediction algorithms and still satisfying the computational constraints. An ideal prediction model should combine the simplicity of the LR model, the AR and ARIMA qualities of reproducing the stochastic pattern of the data set and the EWMA ability of smoothing the noises components. The idea is that a valid prediction algorithm should not consider just the data points, but it should also be able to estimate from the data set the *trend* of the load behavior. The load trend gives an additional information that can be used to evaluate whether a resource load is increasing, decreasing, oscillating or stabilizing. We can define several models that can combine data set values and load trend information for prediction purposes. In this paper, we propose the Trend-Aware Regression (TAR) model, that bases its predictions on the following steps.

All runtime prediction models at time t_i work on the basis of an ordered set of past information S_i. A prediction is the output of a generic function conditioned on S_i, that is, $\hat{v}_{i+k} = g(S_i) + \epsilon_i$ in which $g()$ captures the predictable component, ϵ_i models the effects of the noise and k denotes the number of steps in the future. The first choice for the estimation of a future value of the time series concerns the past values that the prediction model wants to consider. At time t_i, the TAR model defines the *history vector*, namely H_i, as a subset of the data set S_i that is associated with the last value $s_i = (t_i, v_i)$. The history vector contains m values that are selected with a constant frequency q, from v_i backward to $v_{i-(m-1)q}$. Here, we consider the more general instance of $q > 1$. The typical goal is to use these past values to predict the value of the point $\hat{v}_{i+k}$ which is positioned k steps ahead at time t_{i+k}. The idea behind

the TAR model is not to use directly the past values v_{i-jq} ($0 \leq j \leq m-1$), but the degree of variation between each couple of consecutive points $s_{i-jq} = (t_{i-jq}, v_{i-jq})$ and $s_{i-(j+1)q} = (t_{i-(j+1)q}, v_{i-(j+1)q})$ for $0 \leq j \leq m-2$. In particular, the *degree of variation* a_{i-jq} is computed as following:

$$a_{i-jq} = \gamma_j \frac{v_{i-jq} - v_{i-(j+1)q}}{t_{i-jq} - t_{i-(j+1)q}}; \quad 0 \leq j \leq m-2 \qquad (3)$$

where

$$\gamma_j = \frac{t_{i+k} - t_i}{t_{i-jq} - t_{i-(j+1)q}} \qquad (4)$$

The γ_j parameter represents the *scaling factor* which adapts the degree of variation computed in the Cartesian space at time t_i with the degree of variation that will be utilized in the space k steps ahead. Figure 1 offers a geometric representation of these phases of the TAR prediction model when the history vector contains $m = 4$ values. These points denote three segments (s_{i-3q}, s_{i-2q}), (s_{i-2q}, s_{i-q}) and (s_{i-q}, s_i). We associate a degree of variation to each segment, thus obtaining a_{i-2q}, a_{i-q} and a_i. The absolute value of the j-th degree of variation $\mid a_{i-jq} \mid$ represents the intensity of the variation between two consecutive points s_{i-jq} and $s_{i-(j+1)q}$. The sign of a_{i-jq} denotes the direction of the variation: a plus represents an increment of the value between the s_{i-jq} and $s_{i-(j+1)q}$ data points; a minus denotes a decrease of the trend. The problem is now related

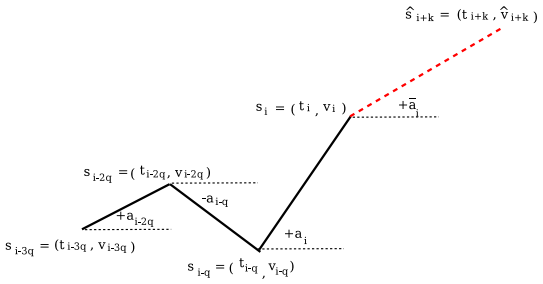


Figure 1. Variation trend and degrees of variation for $m = 4$ past values.

to how to combine the degrees of variation for future predictions. In other words, we want to evaluate the so called *variation trend* $\bar{a}_i$ for estimating the value $\hat{v}_{i+k}$ of the point $\hat{s}_{i+k} = (t_{i+k}, \hat{v}_{i+k})$, as shown in the rightmost part of Figure 1. There are several possible alternatives, hence in this paper we investigate one solution that gives satisfactory results. The variation trend is evaluated as the weighted linear regression of the $m-1$ degrees of variation:

$$\bar{a}_i = \sum_{j=0}^{m-2} p_j a_{i-jq}; \qquad \sum_{j=0}^{m-2} p_j = 1 \qquad (5)$$

where the best choice of the p_j coefficients opens another space of alternatives that are outside the scope of this paper. For reasons of space, we consider just an exponential decrease of the weights as a function of m, that gives more importance to the more recent trend coefficients. Hence, the trend-aware regression model for the estimation of the future point $\hat{v}_{i+k}$ is obtained through the following linear combination of the computed and available values:

$$\hat{v}_{i+k} = \bar{a}_i (t_{i+k} - t_i) + v_i \qquad (6)$$

In the experiments, when not otherwise specified, we have set $q = 5$ and $m = 3$. The sensitivity analysis shows the stability of the TAR model to these parameters.

4 Prediction results

4.1 Experimental testbed

The experimental evaluation refers to a popular Web system referring to a multi-tier architecture. It is based on the implementation presented in [7], where the first node executes the HTTP server, the application server is deployed through the Tomcat servlet container, and the back-end node runs a MySQL database server. The workload follows the TPC-W model, an industrial benchmarking for the performance evaluation of dynamic Web systems [7]. Client requests are generated through a set of *emulated browsers*, where each browser is implemented as a Java thread reproducing an entire user session with the Web site. The experiments last for 8000 seconds for the following three workloads.

Stable scenario: it describes a TPC-W workload in the ideal case of a number of clients that does not change during the experiment. (The shown results refer to 120 emulated browsers.)

Real scenario - *light* and *heavy*: they describe a realistic Web workload where the pattern of active clients changes according the profile presented in [4]. The *light* and *heavy* workloads are characterized by service demands having low and high impact on system resources, respectively. Moreover, this different impact is also characterized by a higher variability of the resource measures in the context of *heavy* workloads.

All scenarios represent the typical Web workload that is characterized by heavy-tailed distributions [3,8]. Moreover, the two real scenarios include bursty arrivals [13] that contribute to augment the request skew.

4.2 Quality of the prediction models

We analyze the quality of five runtime prediction models (AR, ARIMA, EWMA, LR, TAR) that satisfy the computa-

tional requirement. Hence, in this section we are interested to evaluate the precision in forecasting the future data values and the ability to react soon to the variations of the time series. The best model should minimize both the *precision error* and the *delay*. The precision error represents the distance between the ideal filtered data set and the predicted values:

$$\sum_{i=1}^{N} \frac{|\hat{v}_i - v_i^*|}{N} * 100\% \quad (7)$$

where N is the total number of predictions, $\hat{v}_i$ is the predicted value and v_i^* is the offline filtered data. The delay denotes the lack of reactivity of a prediction model to detect a change of a resource condition. We measure the *average delay* as the average of the horizontal distances in intervals of 30 lags between the observed data set and the predicted data set values. The quality of a prediction algorithm depends on the model characteristics but also on the nature of the data sets. When not otherwise specified, we consider as representative instances of a larger set of results a prediction window corresponding to $k = 30$ lags, a noise index of filtered data equal to 0.1 and three workload scenarios (one stable, two realistic).

Table 2 reports the precision errors for the considered prediction models by distinguishing raw and filtered data. If we limit the acceptability of a model when its precision error is below 20%, the first remark from the results of this table is that no model can work on raw data. As expected, this occurs for any real scenario, and more surprisingly for the stable workload too. For all scenarios, the ARIMA model shows the best results when applied to raw data. Although its precision error is never acceptable, this result confirms the ability of the ARIMA model to follow even complex patterns of the time series.

If we refer to the filtered data, the results of Table 2 confirm that all models are able to reduce the precision error. This error is similar for all models (with the exception of the LR algorithm) in the ideal case of a Stable scenario. When the data set is characterized by a highly variable behavior, such as in both real scenarios, the precision errors differ.

Table 2. Precision error ($k = 30$ steps ahead)

	AR	ARIMA	EWMA	LR	TAR
	Stable scenario				
Raw data	20.6%	20.1%	23.0%	45.5%	25.2%
Filtered	3.3%	2.9%	2.8%	4.4%	2.8%
	Real scenario - *light*				
Raw data	27.0%	23.4%	26.0%	43.8%	32.5%
Filtered	8.0%	9.0%	6.2%	12.1%	3.7%
	Real scenario - *heavy*				
Raw data	28.6%	25.0%	27.0%	45.8%	37.5%
Filtered	14.0%	12.0%	7.0%	12.1%	4.0%

The AR and ARIMA algorithms are the most affected models because they tend to reproduce the data set behavior. However, when the variance of the time series values varies over time, they would require a continuous evaluation and update of their parameters. As this method is unfeasible at runtime for short-term prediction, the consequence is a precision error two-three times higher than that characterizing the TAR model. It is important to observe that in our experiments this last algorithm has always demonstrated its ability to limit the precision error to 4%. Even the EWMA model guarantees adequate precision, but we will see that it is affected by a high delay.

Another important measure for evaluating the prediction models is represented by the *average delay* in detecting load state changes. Table 3 reports this delay for the considered

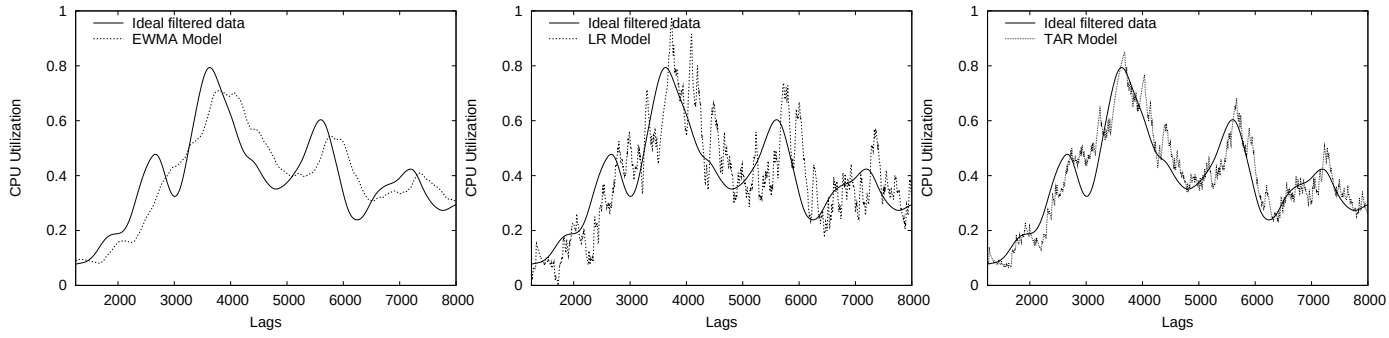
Table 3. Average delay (*lag*)

Scenario	AR	ARIMA	EWMA	LR	TAR
Stable	2	2	30	26	2
Real - *light*	5	3	34	25	2
Real - *heavy*	6	3	36	25	3

prediction models. These results indicate that AR, ARIMA and TAR are the most responsive prediction models because they are characterized by a delay of few lags. On the other hand, the EWMA and LR models are affected by a high delay. This is a consequence of the smoothing effects of lower order prediction models that do not allow them to react quickly enough to the changes of the data set pattern. The delay effect is confirmed by the Figures 2 showing the predicted result pattern with respect to the filtered data set. These figures confirm from a qualitative point of view the capacity of the TAR model to respond quickly to the variations of the data set. Figure 2(b) evaluates that the LR model is affected by high delay and scarce precision. On the other hand, the EWMA model is rather precise, but it follows the pattern changes with a consistent delay that is evidenced by the horizontal distances of the two curves. A delay error of more than 30 lags when the prediction window is equal to 30 lags shows that the EWMA model is inadequate for short-term predictions. From all these results, we can conclude that in ideal conditions the *precision error* of the prediction models is similar, while in realistic Web scenarios the models show significant differences with a clear advantage of the TAR model. In the next subsection we aim to validate these results for different conditions.

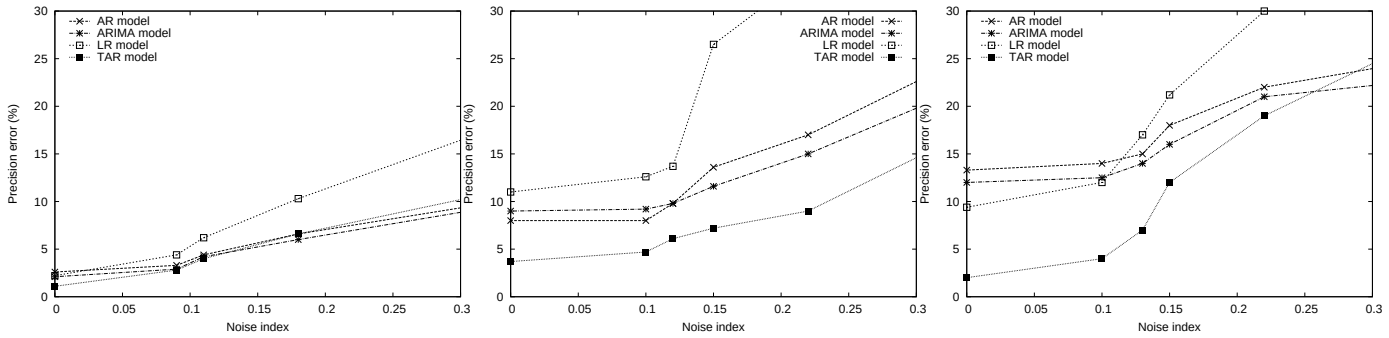
4.3 Sensitivity analysis

All the previous results refer to a short-term prediction with $k = 30$ steps ahead and a noise factor equal to $\delta = 0.1$. In these conditions, the TAR algorithm shows the best precision and minimum delay. However, it is important to



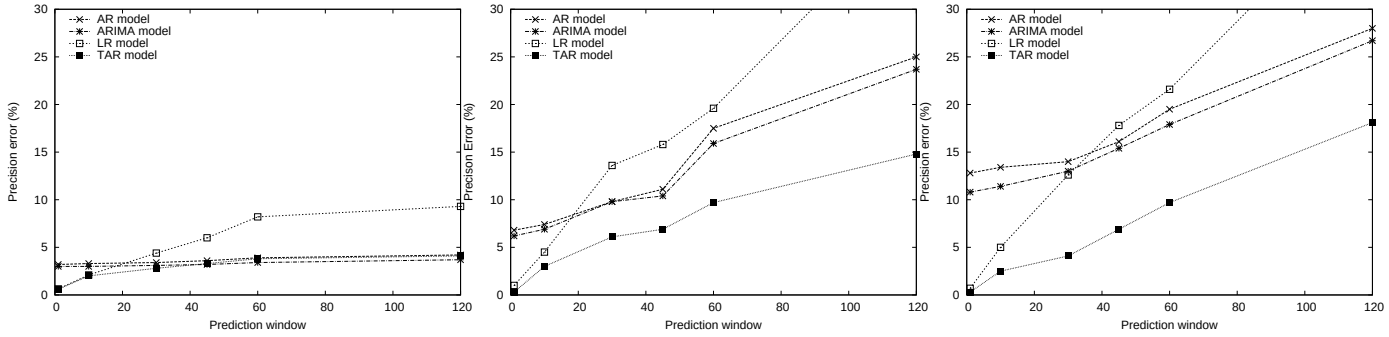
a) EWMA model **b) LR model** **c) TAR model**

Figure 2. Qualitative analysis of the delay



a) Stable scenario **c) Real scenario - *light*** **b) Real scenario - *heavy***

Figure 3. Precision error as a function of the noise index (δ)



a) Stable scenario **b) Real scenario - *light*** **c) Real scenario - *heavy***

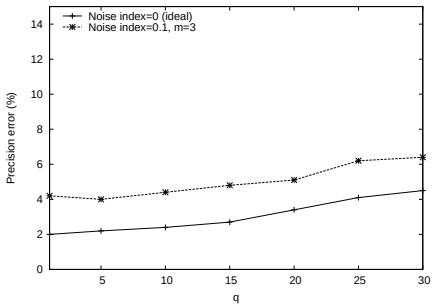
Figure 4. Precision error as a function of the prediction window (k lags)

evaluate the stability of the TAR results for different values of the noise index and for different prediction window sizes, and its robustness with respect to its main parameters.

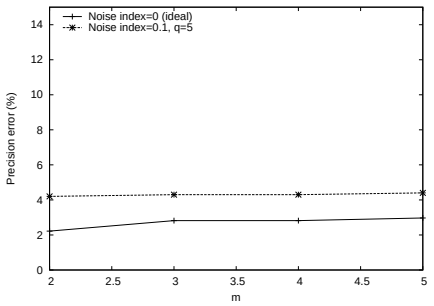
Figure 3 shows the precision errors as a function of different noise values of the filtered data sets. An expected trait of all scenarios is the increase of the precision errors as a consequence of a higher noise factor. In the ideal stable scenario, all prediction models (but the LR) achieve an acceptable precision below 10% (Figure 3(a)). In the real scenarios, when $\delta \geq 0.3$, all precision errors exceed 20%, hence it is reasonable to consider the predicted results useless. The range of feasibility of the models of interest for runtime management is for $\delta < 0.3$. Figures 3(b) and 3(c) show interesting differences among the prediction models: the LR model is clearly unacceptable in a realistic context; the AR and ARIMA models are characterized by close results, hence we could choose the former algorithm because of its minor computational complexity; the TAR model outperforms all the other predictors in the range of interest because it guarantees a precision error two-three times lower.

It is interesting that similar results are obtained for completely different workload scenarios.

Another important factor is the sensitivity of the results to the temporal size of the prediction window k. As we consider short-term prediction, we focus on the [1, 120] interval (in lags). Figure 4 reports the results for the stable and real scenarios. In the ideal scenario, all prediction models are valid and share similar behavior. In the more realistic scenarios, it is important that the precision error of the TAR model does not exceed 15% even for $k = 120$ lags. The results of the LR model are unacceptable, and even the AR and ARIMA models show a twofold precision error if compared to the TAR error. We observe that all the previous results are achieved for fixed values of the parameters of the TAR algorithm. In a dynamic context, it is important to measure the robustness of the TAR model by evaluating its sensitivity to its main parameters, that is, the past values $(m-1)q$ and the number of trend coefficients m. For this analysis, we consider the real scenario and heavy service demand, and the filtered data set with noise index $\delta = 0.1$. From the results in Figure 5 we can appreciate that the prediction quality of the TAR model is preserved for a wide range of q and m values. This stability in a highly variable context is an important feature because no other prediction algorithm guarantees a similar robustness.



a) Sensitivity to the frequency of selected samples



b) Sensitivity to the number of considered trend coefficients

Figure 5. Precision error as a function of the parameters of the TAR model

5 Related work

Predictive algorithms would play a crucial role in managing the resources of Web systems, but the majority of prediction models is based on off-line analyses of user accesses and resource logs [4, 9, 15, 18, 21]. Many other prediction models, such as genetic algorithms, support vector machines and Fuzzy systems, are suitable to off-line applications and medium-long term predictions [10]. Some models on short-term prediction for on-line decisions make strong assumptions on the mathematical characteristics of the workload. A feedback control prediction mechanism that works well for mildly oscillating or stationary workloads is presented in [16], while different autoregressive models for the predictability of the CPU load average are presented in [11]. The authors in [19] explore the benefits of the integration of non-linear corrective factors in a linear regression to predict FTP transfer times. Sang and Li compute precise lower and upper bounds for the prediction interval and error of network traffic, which is assumed to be a Gaussian process [18]. All these assumptions do not hold for the workloads characterizing the Web systems considered in this paper. This is a huge difference with respect to previous work because when you move from ideal to realistic workload conditions, all the previous prediction models exhibit high precision errors. The proposed TAR prediction algorithm works finely even in the typical conditions of real workload scenarios that exhibit specific statistical properties, such as heavy-tailed distributions [3, 8], bursty arrivals [13], and hot spots [4].

Other results oriented to the short-term prediction in the presence of non stationary workloads [12, 16] are mainly focused on external factors (such as the arrival frequency) which are turned into a measure of the future internal system load. Forecasting the future resource state or future resource demands exclusively from the offered load may lead to wrong internal decisions, especially in Web systems where each arrival may activate a high number of requests to the internal hardware and software resources with unpredictable mutual interactions and interdependencies. The prediction of load demands to internal resources of Web systems has not received much attention yet. Pacifici et al. [17] apply filtering techniques to predict the CPU demand of Web applications. The authors in [2] discuss the effects of a typical Web workload on the system resources and propose and compare different representations of the resource load with the main purpose of detecting load state changes. This paper has an innovative focus on the short-term prediction of the internal resource state. To this purpose, we apply an original trend-based prediction model to a representation of the data points referring to the internal resources.

6 Conclusions

The Web systems are expected to satisfy scalability, availability, and short-time servicing requirements under critical workload conditions. These external characteristics together with the increasing complexity of the hardware/software architectures make resource management tasks really difficult because the decision algorithms cannot know or predict from the observed data points whether a resource is offloading or overloading. This paper represents a first contribution in the direction of investigating and predicting the performance of the internal resources of Web architectures in a short-term horizon. We have shown that observed data points do not allow any prediction, but an adequate online filter can reduce noises and exhibit data sets with some short-term dependency. However, some popular models that could be used for runtime prediction achieve unacceptable results even on filtered data set. For this reason, we propose a new Trend-Aware Regression algorithm that shows the best precision for different scenarios, and stable results for a wide range of parameters and workloads. There are several exploitations of the proposed model because short-term prediction is at the basis of most runtime management decisions, such as load sharing, admission control, hot spot control, request redirection. From a modeling point of view, the precision of the TAR model can be even improved by some dynamic adaptation of its parameters to the statistical variations of the observed data points.

References

[1] T. Abdelzaher, K. G. Shin, and N. Bhatti. Performance guarantees for Web server end-systems: A control-theoretical approach. *IEEE Trans. Parallel and Distributed Systems*, 13(1):80–96, Jan. 2002.

[2] M. Andreolini, S. Casolari, and M. Colajanni. Models and framework for supporting run-time decisions in web-based systems. *ACM Trans. on the Web*, 2(3), Aug. 2008.

[3] M. Arlitt, D. Krishnamurthy, and J. Rolia. Characterizing the scalability of a large web-based shopping system. *IEEE Trans. Internet Technology*, 1(1):44–69, Aug. 2001.

[4] Y. Baryshnikov, E. Coffman, G. Pierre, D. Rubenstein, M. Squillante, and T. Yimwadsana. Predictability of Web server traffic congestion. In *Proc. of the 10th WCW2005*, Sophia Antipolis, FR, Sept. 2005.

[5] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall.

[6] B. L. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer-Verlag, 1987.

[7] 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*, Monterrey, ME, Jan. 2001.

[8] J. Challenger, P. Dantzig, A. Iyengar, M. Squillante, and L. Zhang. Efficiently serving dynamic data at highly accessed Web sites. *IEEE/ACM Trans. on Networking*, 12(2):233–246, Apr. 2004.

[9] B. Choi, J. Park, and Z. Zhang. Adaptive random sampling for load change detection. In *Proc. of 16th IEEE Int. Conf. on Communications*, Anchorage, AL, May 2003.

[10] L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, 1996.

[11] P. Dinda and D. O'Hallaron. Host load prediction using linear models. *Cluster Computing*, 3(4):265–280, Dec. 2000.

[12] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Capacity management and demand prediction for next generation data centers. In *Proc. of 5th IEEE Int. Conf. on Web Services*, Salt Lake City, UT, 2007.

[13] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: characterization and implications for CDNs and Web sites. In *Proc. of WWW2002*, Honolulu, HW, May 2002.

[14] D. J. Lilja. *Measuring computer performance. A practitioner's guide*. Cambridge University Press, 2000.

[15] Y. Lingyun, I. Foster, and J. M. Schopf. Homeostatic and tendency-based CPU load predictions. In *Proc. of 17th Parallel and Distributed Processing Symposium*, Nice, FR, 2003.

[16] Y. Lu, T. Abdelzaher, L. Chenyang, S. Lui, and L. Xue. Feedback control with queueing-theoretic prediction for relative delay guarantees in Web servers. In *Proc. of 9th IEEE real-time and embedded technology and Applications Symposium*, Charlottesville, VA, May 2003.

[17] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi. Cpu demand for web serving: Measurement analysis and dynamic estimation. *Performance Evaluation*, Dec. 2007.

[18] A. Sang and S. Li. A predictability analysis of network traffic. In *Proc. of INFOCOM2000*, Tel Aviv, Mar. 2000.

[19] S. Vazhkudai and J. Schopf. Predict sporadic grid data transfers. In *Proc. of 11th IEEE Symp. on High Performance Distributed Computing*, Edinburgh, UK, July 2002.

[20] R. Vilalte, C. V. Apte, J. L. Hellerstei, S. Ma, and S. M. Weiss. Predictive algorithms in the management of computer systems. *Tech. Rep. HPL-2003,23*, Feb. 2003.

[21] X. Wu, V. Taylor, and J. Paris. A web-based prophesy automated performance modeling system. In *Proc. of Web Tech., Applications and Services*, Calgary, Canada, July 2006.