Load prediction models in Web-based systems

Mauro Andreolini
Department of Information Engineering
University of Modena and Reggio Emilia
andreolini.mauro@unimore.it

Sara Casolari
Department of Information Engineering
University of Modena and Reggio Emilia
casolari.sara@unimore.it

ABSTRACT

Run-time management of modern Web-based services requires the integration of several algorithms and mechanisms for job dispatching, load sharing, admission control, overload detection. All these algorithms should take decisions on the basis of present and/or future load conditions of the system resources. In particular, we address the issue of predicting future resource loads under real-time constraints in the context of Internet-based systems. In this situation, it is extremely difficult to deduce a representative view of a system resource from collected raw measures that show very large variability even at different time scales. For this reason, we propose a two-step approach that first aims to get a representative view of the load trend from measured raw data, and then applies a load prediction algorithm to load trends. This approach is suitable to support different decision systems even for highly variable contexts and is characterized by a computational complexity that is compatible to run-time decisions. The proposed models are applied to a multi-tier Web-based system, but the results can be extended to other Internet-based contexts where the systems are characterized by similar workloads and resource behaviors.

1. INTRODUCTION

The most critical Internet-based services are provided by distributed infrastructures that have to satisfy scalability and availability requirements, and have to avoid performance degradation and system overload. Managing these systems requires a large set of algorithms, for load balancing and load sharing [2, 8, 32], overload and admission control [14, 20, 28], job dispatching and redirection even at a geographical scale [9]. The advent of self-adaptive systems and autonomic computing [21, 25, 39] will further increase the necessity for management algorithms requiring a run-time evaluation of the load conditions of hardware and software system resources, and the possibility of predicting future load values of the system components. Almost all algorithms and mechanisms evaluate the load conditions of a system resource through the periodic sampling of raw data that we call *resource measures*. While a measure offers an instantaneous view of the load conditions of a resource, it is of little help for distinguishing overload conditions from transient peaks, for understanding load trends and for anticipating future conditions, that are of utmost importance for taking correct decisions.

In this paper, we address the important issue of predicting future load conditions of a resource, that is at the basis of several run-time management tasks. We will see that the direct use of measured raw data does not allow us to solve this problem, because resource measures obtained from load monitors of Internet-based servers are extremely variable even at different time scales, and tend to become obsolete rather quickly [16]. Consequently, long time measurement intervals reduce the effectiveness of the resource measure as a suitable indicator of the real load conditions. Moreover, the workload reaching the Internet-based systems that we consider in this paper is typically characterized by heavy-tailed distributions [4, 11, 15] and by flash crowds [23] that contribute to augment the skewness of the raw data. These characteristics of load resource measures in Internet-based systems make really difficult, if not impossible, to forecast the behavior of future resource measures because they appear as unrelated to the previous sample, and to deduce a clear trend about the load behavior of a resource, for example to find out whether a resource is offloading, overloading or stabilizing.

We have verified that, in a heavy-tailed context, it is of little value to let a load predictor work directly on resource measures, because they give only a limited and instantaneous view of the resource and do not capture its behavioral trend that is of major interest for prediction. Hence, we can anticipate that in Web-based systems it is not convenient to base load prediction directly on resource measures, as done in different contexts [1, 5, 12], but it is preferable to operate on a "representation" of the load behavior of system resources. To this purpose, we propose and compare different functions, called *load tracker*s, that may offer a representative view of the load conditions to the load predictors, thus achieving the two-step approach shown in Figure 1. In this paper, we consider three classes of load trackers that are suitable to support different decision systems and are characterized by a computational complexity that is compatible to run-time decisions: two of them are based on linear models (*simple* and *exponential moving average*), and one is based on a non-linear model (*cubic spline*). We compare the efficacy of the proposed load trackers for the support of load prediction goals. To this purpose, we define also metrics for evaluating the efficacy of a load tracker that supports *load prediction*, according to the accuracy of the load tracker to represent the real load, and to the error between the real and predicted load values. Our results show that the precision of the predictor depends largely on the choice of the load tracker. Smoother load trackers typically lead to a better prediction because of their small prediction error and low computational costs. Thanks to the two-step approach, and an adequate load tracker, even simple linear-based predictor models are able to

Valuetools'06, October 11-13, 2006, Pisa, Italy

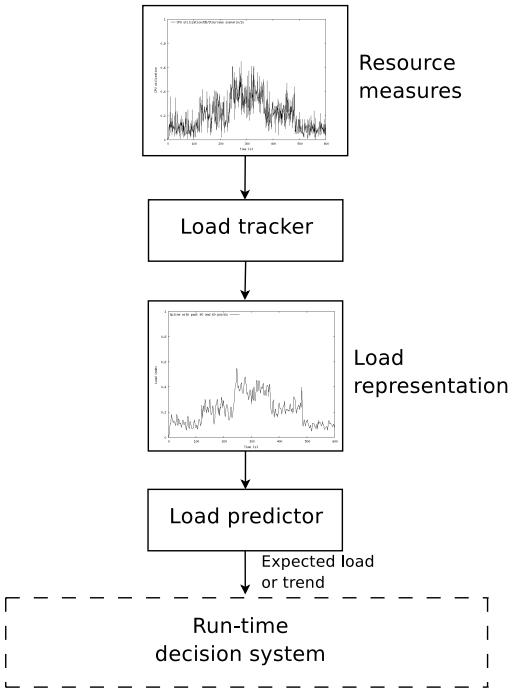


Figure 1: A framework for supporting run-time decisions in Web-based systems

achieve good predictions.
Previous literature [5, 12, 17, 38] uses linear models to predict real resource load measures. However, we should consider that linear models have various drawbacks in the context of highly variable Web-based systems where decisions should be taken in real-time. For example, linear (auto)regression works well provided that the resource measures are highly correlated, which is not always the case in heavy-tailed systems. Furthermore, linear (auto)regressive models require a training period to compute the parameters that may not be available to decision systems subject to run-time constraints.
The experimental results are based on realistic scenarios in the context of a multi-tier Web-based system, but the proposed methodology is not tied to specific application contexts (such as in [12, 13]) and can be applied to other Internet-based systems as well.
The paper is organized as follows. Section 2 gives a motivation for this paper by showing the extreme variability of resource measures at different time scales and under different Web related workload scenarios. It also describes the test-bed architecture and workload models that we use in this paper. Section 3 proposes the load trackers and their properties for load prediction goals that are detailed in Section 4. Section 5 shows the experimental results. Section 6 compares the results of this paper with respect to the state of the art. Section 7 concludes the paper with some final remarks.

2. RESOURCE MEASURES

In this section, we consider the behavior of measured resource loads in the context of a dynamic Web-based system from which we can see that the view of a resource load obtained by monitors is extremely oscillatory at different time scales and for different workload scenarios. The considered test-bed refers to a typical multi-tier logical architecture (Figure 2) that is based on the implementation presented in [7]. The workload refers to the TPC-W model [37] that is becoming a de facto standard for the performance analysis of Web-based systems (e.g., [7, 10, 18]). The first node executes the HTTP server and the application server that is deployed through the Tomcat [36] servlet container; the second node runs the MySQL [29] database server. Requests are generated through a set of *emulated browsers*, where each browser is implemented as a Java thread that emulates an entire user session. We use three TPC-W like workload mixes that represent different scenarios shown in Figure 3.

- **Step scenario.** This scenario describes a sudden load increment from a relatively unloaded to a more loaded system. The population is kept at 300 emulated browsers for 5 minutes, then suddenly increased to 700 emulated browsers for other 5 minutes.

- **Staircase scenario.** This scenario represents a gradual increment in the population up to 600 emulated browsers, that is followed by a similar decrease.

- **Alternating scenario.** This scenario describes an alternating increment and decrease of the load between 300 and 600 emulated browsers, every two minutes.

There are many critical resources in each server node that can be measured through several system monitors [22, 31, 34]. These tools yield instantaneous resource measures at regular time intervals. We have carried out a very large set of experiments for analyzing the typical behavior of commonly measured resources, such as CPU utilization, disk and network throughput, number of open sockets, number of open files, UNIX load, amount of used main memory, for different resource measure intervals and workload scenarios. Here, we report a subset of results that are representative of the main observations and conclusions about the behavior of resource measures. In the Figures 4 we describe the CPU and disk measures of the back-end node of the multi-tier architecture shown in Figure 2. In particular, we report:

- two rates for resource measurement: 1 second and 5 seconds (Figure 4 (a) and (b), respectively);

- two resource metrics: disk throughput (blocks/sec.) and CPU utilization (Figure 4 (c) and (d), respectively);

- three user scenarios: step, staircase, alternating (Figure 4 (c), (e) and (f)).

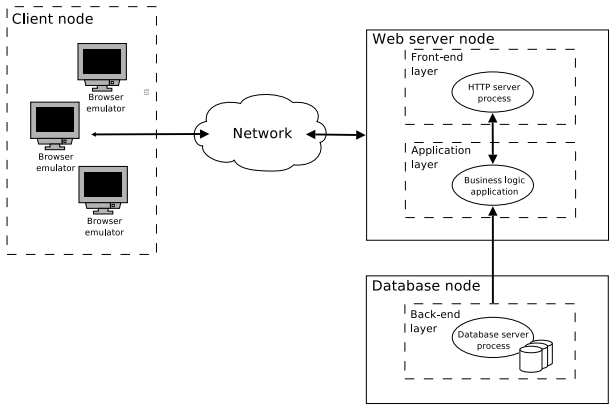
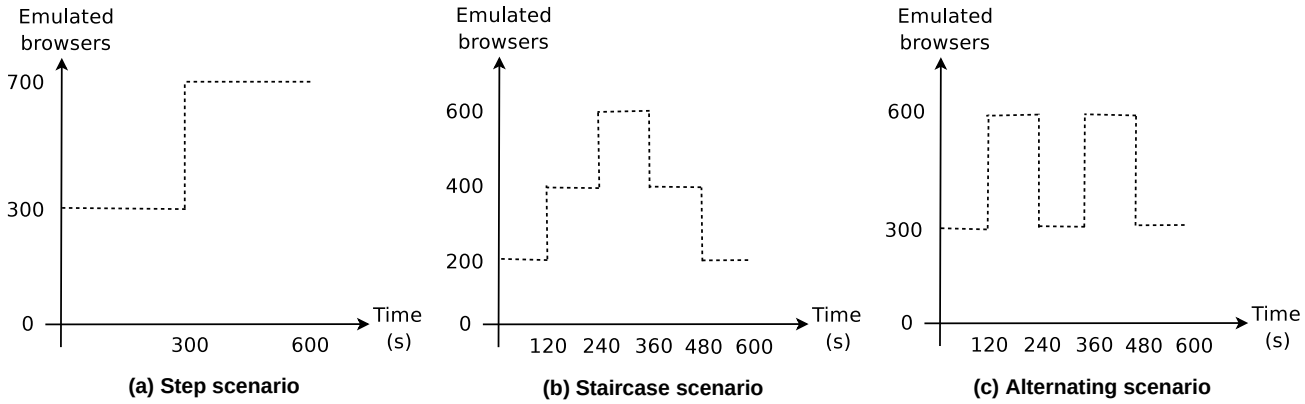
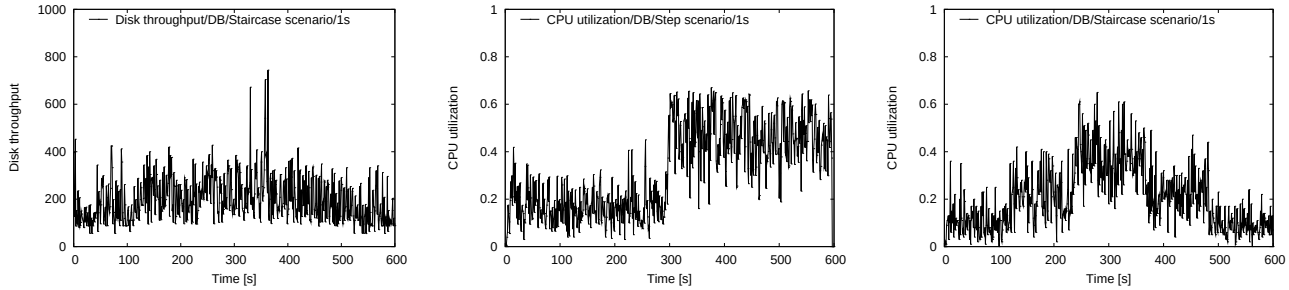


Figure 2: Architecture of a multi-tier Web-based system

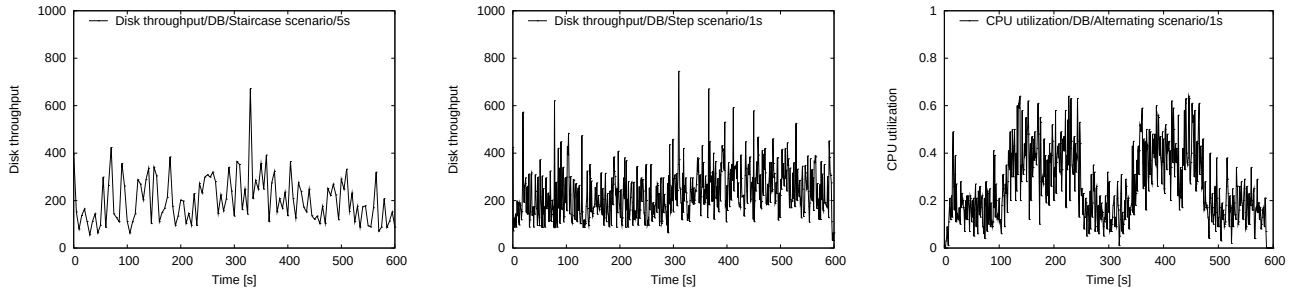


(a) Step scenario **(b) Staircase scenario** **(c) Alternating scenario**

Figure 3: Considered workload scenarios



(a) Sample rate: 1s **(c) Resource: CPU** **(e) Scenario: prediction**



(b) Sample rate: 5s **(d) Resource: disk** **(f) Scenario: trend**

Figure 4: Resource measurements

All these figures share the common trait that the view of a resource obtained from monitored measures is extremely variable to the extent that run-time decisions based on these values may be risky when not completely wrong. Even if a load predictor is able to guess precisely a load value in the future, is this value really useful for taking a correct decision? For example, let us suppose that a load predictor evaluates a future CPU utilization measure with high precision, and this predicted value is used by an admission controller to decide about the acceptance/refusal of requests for the satisfaction of some service level agreements. Actually, a predicted value is a single point in the space, and no one in a highly variable context would have faith in it for taking critical decisions. If we do not figure out the behavioral trend from a sequence of measures, even accurate predicted values share the same characteristics of the resource measures: instability, variability, low autocorrelation. This variability is critical even for some performance control mechanisms [1, 24] that follow a feedback control theory approach, where the major risk is represented by the instability and self-induced oscillations.

Another question arises about the computational complexity of the load predictor models. In some literature, the high skewness of load measurements is addressed by sophisticated load predictor models that may require training periods and/or off-line analyses. These models are not suitable to support the run-time decision systems considered in this paper. In other words, we should not look for the most precise load predictor, but for the best predictor that is able to work subject to real time constraints.

3. LOAD TRACKER FUNCTIONS

In this section, we describe the first phase of the two-step approach that aims to get a representative view of the load trend from measured raw data instead of letting a load predictor work on raw measures of resource loads. To this purpose, we propose a new function, called *load tracker*, that filters out the noises presented by a sequence of uncorrelated resource measures and yields a more regular view about the load trend of a resource.

We consider a resource measure s_i sampled at time t_i, and a set of previously collected n measures, that is, $\overrightarrow{S_n}(t_i) = (s_{i-n}, \ldots, s_i)$. We define *load tracker* a function $LT(\overrightarrow{S_n}(t_i)) : \mathbb{R}^n \rightarrow \mathbb{R}$ that takes as its input $\overrightarrow{S_n}(t_i)$ and gives a "representation" of the resource load conditions l_i at time t_i. Multiple applications of the load tracker give a sequence of load values that should exclude out-of-scale resource measures and yield a regular trend of the resource load conditions. For the purposes of this paper, we consider and compare two linear and one non-linear load trackers as supports for load predictors.

3.1 Linear load trackers

We consider the class of *moving averages* as linear load trackers. Indeed, moving averages smooth out resource measures, reduce the effect of out-of-scale values, are fairly easy to compute at run-time, and are commonly used as trend indicators [26]. We focus on two classes of moving average: the *Simple Moving Average* (SMA) and the *Exponential Moving Average* (EMA) that use uniform and non-uniform weighted distributions of the past measures, respectively. We do not consider other popular linear auto regressive models, such as ARMA and ARIMA [17, 38], because they would require frequent updates of their parameters in the case of highly variable systems. For these reasons, the auto regressive models are typically created offline after examination of all available data, or applied to workloads characterized by smaller variability and high auto-correlation of load measures. These operations are computationally too expensive and inadequate to support run-time decision systems, especially in a context where auto-correlation values are low.

Simple Moving Average (SMA). It is the unweighted mean of the n resource measures of the vector $\overrightarrow{S_n}(t_i)$, evaluated at time t_i:

$$SMA(\overrightarrow{S_n}(t_i)) = \frac{\sum_{i-n \leq j \leq i} s_j}{n} \tag{1}$$

A SMA-based load tracker evaluates $SMA(\overrightarrow{S_n}(t_i))$ for each considered sample s_i during the entire observation period. As SMAs assign an equal weight to every resource measure, they have to solve a clear trade-off. Load trackers based on short-term moving averages (that is, working on a small set of load measures $\overrightarrow{S_n}(t_i)$) are more responsive to variations of the load conditions, but at the expense of increased oscillations. On the other hand, long-term moving averages are able to smooth out all minor fluctuations, and tend to show only long-term trends [3]. Typically, the SMA models tend to introduce a significant delay in the trend representation when the size of the set $|\overrightarrow{S_n}(t_i)|$ increases. The EMA models are often proposed to limit this delay effect, without incurring in oscillation risks characterizing short-term moving averages.

Exponential Moving Average (EMA). It is the weighted mean of the n resource measures of the set $\overrightarrow{S_n}(t_i)$, where the weights decrease exponentially. A EMA-based load tracker $LT(\overrightarrow{S_n}(t_i))$, for each time t_i where $i > n$, is equal to:

$$EMA(\overrightarrow{S_n}(t_i)) = \alpha * s_i + (1 - \alpha) * EMA(\overrightarrow{S_n}(t_{i-1})) \tag{2}$$

where the constant $\alpha = \frac{2}{n+1}$ is the *smoothing factor*. The initial $EMA(\overrightarrow{S_n}(t_n))$ value is initialized to the arithmetical mean of the first n measures:

$$EMA(\overrightarrow{S_n}(t_n)) = \frac{\sum_{0 \leq j \leq n} s_j}{n} \tag{3}$$

As the last resource measures give a contribution higher than the first measures of the set $\overrightarrow{S_n}(t_i)$, a load tracker based on a EMA model is able to react rather quickly to changes in the resource load conditions similarly to a short-term SMA, with the advantage that EMA is subject to less oscillations than a short-term SMA.

3.2 Non-linear load trackers

The necessity to consider a non-linear tracker depends on some limits shown by the linear load descriptors. We will see that the simple SMA and EMA models introduce a delay in load trend description when $|\overrightarrow{S_n}(t_i)|$ increases. Even the more sophisticated ARMA and ARIMA models are not a valid alternative because they are strongly dependent on the considered resource metrics and workload characteristics.

In this paper, we consider the *cubic spline* function [33] as a representative example of a non-linear tracker. Lower-order curves (with a degree less than 3) do not react quickly to load changes, while higher-order curves (with a degree higher than 3) are considered unnecessary complex, introduce undesired wiggles and are computationally too expensive for a run-time context. For the definition of the cubic spline, let us choose some *control points* (t_j, s_j) in the set of measured load values, where t_j is the time of measurement of s_j. A cubic spline function $CS^J(t)$, based on J control points, is a set of $J-1$ piecewise third-order polynomials $p_j(t)$, where $j \in [1, J-1]$, that satisfy the following properties.

Property 1. The control points are connected through third-order polynomials:

$$\begin{cases} CS^J(t_j) = s_j & j = 1, \ldots, J, \\ CS^J(t) = p_j(t) & t_j < t < t_{j+1},\ j = 1, \ldots, J \end{cases} \tag{4}$$

Property 2. To guarantee a C^2 behavior at each control point, the first order and second order derivatives of $p_j(t)$ and $p_{j+1}(t)$ are set equal at time t_{j+1}:

$$\begin{cases} \frac{dp_j(t_{j+1})}{dt} = \frac{dp_{j+1}(t_{j+1})}{dt}, \\ \frac{d^2p_j(t_{j+1})}{dt^2} = \frac{d^2p_{j+1}(t_{j+1})}{dt^2} \end{cases} \tag{5}$$

If we apply both properties, we obtain the following definition of $CS^J(t)$:

$$\begin{aligned} CS^J(t) = & \frac{z_{j+1}(t-t_j)^3 + z_j(t_{j+1}-t)^3}{6h_i} \\ & + (\frac{s_{j+1}}{h_i} - \frac{h_j}{6}z_{j+1})(t-t_j) \\ & + (\frac{s_j}{h_j} - \frac{h_j}{6}z_j)(t_{j+1}-t) \end{aligned} \tag{6}$$

where $h_j = t_{j+1} - t_j$. The z_j coefficients are solved by the following system of equations:

$$\begin{cases} z_0 = 0 \\ h_{j-1}z_{j-1} + 2(h_{j-1}+h_j)z_j + h_j z_{j+1} = 6(\frac{s_{j+1}-s_j}{h_j} - \frac{s_j - s_{j-1}}{h_{j-1}}) \\ z_n = 0 \end{cases} \tag{7}$$

The spline-based load tracker $LT(\overrightarrow{S_n}(t_i))$, at time t_i is defined as the cubic spline $CS_n^J(t_i)$ that is obtained through a subset of J

control points from the vector of the load measures $\overrightarrow{S_n}(t_i)$, having dimension n.
While the cubic spline is more expensive to compute than the SMA and EMA models, it is commonly used in approximation and smoothing contexts [19, 33, 40], because its computational complexity is compatible to support run-time decision systems. Indeed, load trackers based on moving average models compute a load tracker value at each resource measure, while the proposed load tracker based on the cubic spline function CS_n^J returns a new value after n resource measures. Moreover, the cubic spline has the advantage of being reactive to load changes and independent of resource metrics and workload characteristics.

3.3 Accuracy of load trackers

All the considered load trackers have the common goal of representing the trend of a set of resource measures obtained from some load monitor. A load tracker is accurate if it is able to perfectly follow the *ideal trend*. For the purposes of this paper, we consider as the ideal trend the indicator of the *central tendency* of the resource measures in specific *intervals* of the experiment where the load is subject to significant changes. If you do not control the load generators, the choice of the most suitable intervals where to evaluate the central tendency is a problem by itself. Here, we consider as a reference interval the period of time during which we generate the same number of users (i.e., emulated browsers). For example, in the step scenario, we have two reference intervals ([0,300] and [301, 600]); in the staircase scenario, we have five reference intervals ([0,120], [121,240], [241, 360], [361,480] and [481,600]), and similarly for the alternating scenario.
The simple mean is a good indicator of the central tendency of a set of data [26], and we use this value as the *ideal trend*. In Figures 5, we plot the ideal trends for the three considered scenarios, where the monitored resource is the CPU utilization of the DB server measured every second. We define the accuracy of a load tracker in representing the trend of the resource measures as the sum of the *distances* between each load tracker value l_i computed at time t_i and the corresponding value of the ideal trend id_i at the same instant, that is:

$$\Delta_{LT} = \sum_i |l_i - id_i| \tag{8}$$

where i spans over the load tracker values computed during the observation period. Load trackers with small distance values follow the ideal trend with high accuracy. On the other hand, high distances from the ideal trend values are due to representation delays or oscillations. These behaviors lead to load trackers that are less accurate and even less suitable to support load predictors.

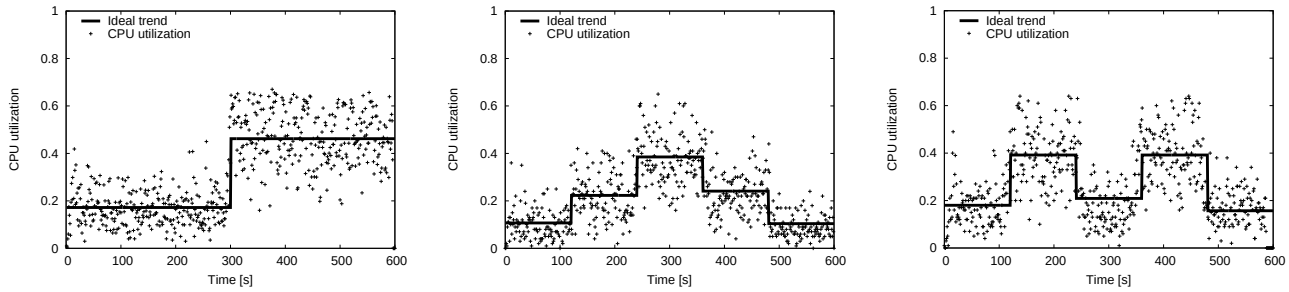
4. LOAD PREDICTION

A load predictor is a function $LP_k(\overrightarrow{L_q}(t_i)) : \mathbb{R}^q \rightarrow \mathbb{R}$ that takes as its input the set of q values $\overrightarrow{L_q}(t_i) = (l_{i-q}, \ldots, l_i)$, and returns a real number that is the predicted value at time t_{i+k}, where $k > 0$. In previous studies, the vector $\overrightarrow{L_q}(t_i)$ consists of a set of real resource measures and the load predictors aim to forecast the future resource measure, at time t_{i+k}. On the other hand, we propose that our load predictor takes as its input a set of load tracker values and returns a future load tracker value. In a context, where the resource measures obtained from the load monitors of the Internet-based servers are extremely variable, there are two reasons that justify our choice.

- The behavior of a measured resource appears extremely variable to the extent that run-time decisions based on these values may be risky when not completely wrong; hence, in our context we think that the prediction of real resource measures is not really useful for taking correct decisions.

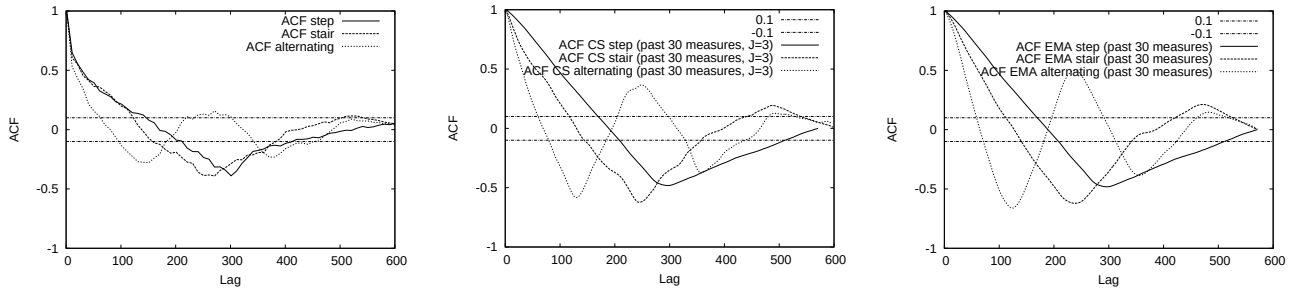
- Many proposed load predictors working on real measures are not suitable to support a run-time decision system because of their computational complexity.

We confirm our hypothesis through a study of the auto-correlation function of the CPU utilization measures. The accuracy of a prediction algorithm depends on the correlation between consecutive resource measures, and when the auto-correlation functions of the set of analyzed data rapidly fall, it is more difficult to have an accurate prediction [5, 38]. In these scenarios, even an attractive approach for statistical modeling and forecasting of complex temporal series, such as the Box-Jankins's ARIMA (Auto-regressive Integrated Moving Average) method [6], tends to produce models that do not adapt well to highly variable changes in the workloads. In Figure 6(a), we analyze the auto-correlation function (ACF) of the CPU utilization for the three considered scenarios during an observation period of 600 seconds. A point (k, y) in this graph represents the correlation value y between the resource measure s_i at time t_i and the measure s_{i+k} at time t_{i+k}. A high value of the auto-correlation function means that the two resource measures are highly correlated, which suggests that the resource measure at time t_i may be used to predict the load at time t_{i+k}. On the other hand, a low value of the auto-correlation function indicates a jittery behavior and an increased difficulty of prediction. From this figure, we can conclude that the resource measure are uncorrelated for any scenario. In similar contexts, using an auto-regressive model such as ARIMA [38] to predict future resource samples may prove computationally expensive, because the model parameters need to be updated frequently, and are not suitable to support run-time decision systems.
Hence, we propose the load tracker results as the basis for load prediction. In Figures 6(b) and (c), we report the auto-correlation diagram of the load tracker values. Let us suppose that the prediction is carried at the instant $t_i = 0$, and that the prediction window of interest for our study is an interval of 30 seconds. In the interval $[0 - 30]$, the ACF of resource measures decreases much more abruptly than the ACFs of the two proposed load trackers. This result is important, because in a context of consecutive values showing a high correlation degree, the accuracy of the load prediction is more likely. Thanks to the use of a load tracker that provides high auto-correlation among values, even simple linear predictors should be sufficient to forecast the future behavior of resource load. Indeed, previous studies [5, 27, 35] show that simple linear models, such as the AR model or the linear interpolation, are adequate for prediction when the correlation of consecutive resource measures is high. For example, Dinda shows that the UNIX load average can be predicted best through an auto-regressive model that takes into account the last 16 measures ($AR(16)$), because of its good predictive power and low computational cost [17].
In this paper, we consider a set of load predictors $LP_k(\overrightarrow{L_q}(t_i))$ that are based on the linear regression of two available load tracker values. Each predictor in this class is characterized by a couple of values: the predicted window k, that represents the size of the prediction interval, and the past time window q, where q is the size of load tracker vector $\overrightarrow{L_q}(t_i)$, that is the distance between the first and last load tracker value. Let us consider two load tracker values: the first l_{i-q} and the last l_i. The load predictor $LP_k(\overrightarrow{L_q}(t_i))$ of the load tracker is the line that intersects the two points (t_{i-q}, l_{i-q}) and (t_i, l_i) and returns $\hat{l}_{i+k}$ that is the predicted value of the load



(a) Step scenario (b) Staircase scenario (c) Alternating scenario

Figure 5: Ideal trends for the three scenarios.



(a) ACF of the measured values (b) ACF of the CS^3_{30} values (c) ACF of the EMA_{30} values

Figure 6: Autocorrelation function

tracker l_{i+k} at time t_{i+k}:

$$\begin{cases} LP_k(\overrightarrow{L_q}(t_i)) = m * (t_{i+k}) + a \\ m = \frac{l_i - l_{i-q}}{q} \\ a = l_{i-q} - m * t_{i-q} \end{cases} \quad (9)$$

The strength of a load predictor depends on its accuracy to evaluate the future values of the load tracker. The common way to measure the accuracy of a prediction is through the evaluation of the relative error between a load tracker value l_{i+k} and the corresponding predicted value $\hat{l}_{i+k}$. A load predictor characterized by a low prediction error is able to evaluate future load tracker accurately.

Let us consider a load tracker $LT(\overrightarrow{S_n}(t_i))$ and a load predictor $LP_k(\overrightarrow{L_q}(t_i))$ that, at time t_i, forecasts $LT(\overrightarrow{S_n}(t_{i+k}))$ where $k > 0$. We define the *prediction error* ϵ_{i+k} at time t_{i+k} as the relative error between the actual load tracker value l_{i+k} and the predicted value $\hat{l}_{i+k}$:

$$\epsilon_{i+k} = |\frac{l_{i+k} - \hat{l}_{i+k}}{l_{i+k}}| \quad (10)$$

Small values of ϵ_{i+k} indicate a good accordance between l_{i+k} and $\hat{l}_{i+k}$. We will compare the accuracy of the load predictor in Equation (9) applied to the load trackers presented in Section 3 by evaluating the mean Δ_P of all prediction errors throughout the entire observation period:

$$\Delta_P = \frac{\sum_i \epsilon_{i+k}}{M} \quad (11)$$

where M is the length of the entire observation period.

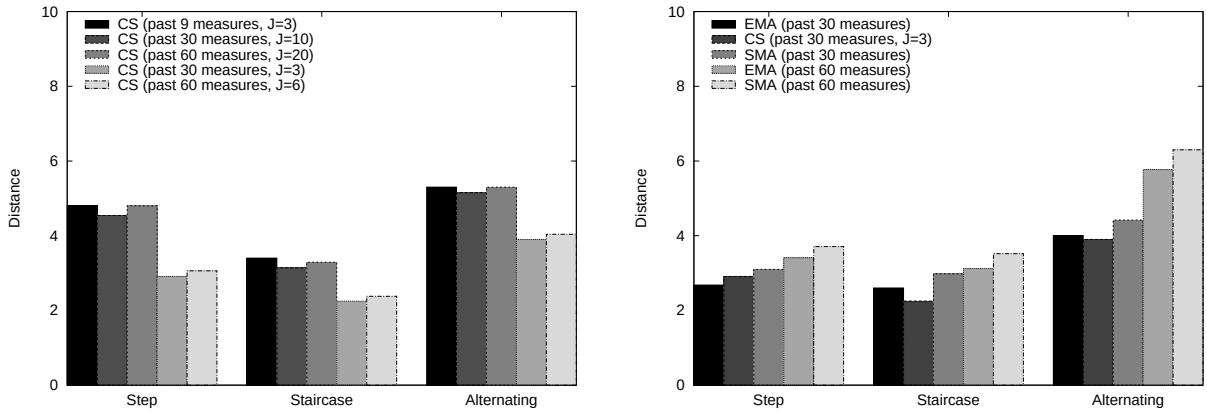
5. RESULTS

5.1 Accuracy of the load trend representation

This section has the twofold goal of analyzing both the accuracy of a load tracker in representing the load trend of a set of resource measures, and the accuracy of the considered load predictor to forecast the future load trend. To this purpose, we use the three classes of load trackers proposed in Section 3 and the linear load predictor proposed in Section 4.

Let us first evaluate which of the considered load trackers are representative of the considered ideal trends. We are also interested to find out the impact of the parameters of the load tracker functions, that is, the size n of the measurement vector $\overrightarrow{S_n}(t_i)$, and the number of control points for the cubic splines. To this purpose, in Figures 7, we report the the distance Δ_{LT} (see Equation (8)) between each load tracker and the corresponding ideal trend for the step, staircase and alternating scenarios. In the four Figures 8 we give a graphical representation of the behavior of some load trackers with respect to the ideal trend values for the staircase scenario.

We first consider in Figure 7(a) the non linear load trackers (CS) that depend on the dimension of the vector of resource measures (n) and on the number of control points (J). We carried out a large set of experiments, and we report here some significant results for $n = 9, 30, 60$ and $J = 3, 6, 10, 20$ (shown between brackets in the figure). In general, the ability of a cubic spline based load tracker to represent the ideal trend, depends on the scenario that it needs to describe. A load tracker that represents perfectly the ideal trend for only one specific scenario and that is inadequate to describe different workloads, can not be considered the "best" load tracker. For this reason, we analyze the distance Δ_{LT} for all the three considered workload scenarios. The best performing load tracker is the cubic spline CS^3_{30}, because it shows the smaller distance Δ_{LT} for every considered workload mix. If we also consider the CS



(a) Cubic spline load trackers **(b) Linear load trackers and best cubic spline**

Figure 7: Distances among the load tracker values and the ideal trend points for the three scenarios

curves in Figures 8(c), we can observe that the behavior of the cubic splines for different sizes of the measure vector is similar. However, when the number of resource measures n is kept constant, the distance Δ_{LT} increases with the number of control points J. With a higher number of control points, the cubic spline is forced to touch more points; as these points are resource measures with high oscillations, the resulting cubic curve presents more ripples. Hence, we conclude that it is better to work with a small number of control points.

In Figure 7(b), we report the distances referring to the linear load trackers (EMA and SMA) and the best performing cubic spline, that is, the CS^3_{30} load tracker working on a set of 30 load measures and 3 control points. If we consider Figures 7(b) and 8, we observe a tradeoff between a reduced delay and a reduced degree of oscillations. We also notice that the distance Δ_{LT} increases when the linear load trackers use a wider resource measurement vector ($\overrightarrow{S_{60}}(t_i)$). This implies a reduced accuracy in representing the load trend, which holds true for all the considered load trackers. When a larger vector of resource measures is used, the linear load trackers introduce a delay. Similarly, a reduced number of resource measures reduces the delays, improves the reactivity to load changes, but it does not smooth out oscillations. Hence, for a good representation of the load trend, it is necessary to find a value of n that represents a good tradeoff between a reduced delay and a reduced degree of oscillations. The exponential moving average EMA_{30} is the load tracker that presents the smallest distance Δ_{LT} and has results comparable with those of the cubic spline CS^3_{30}. For example, in the staircase scenario we have $\Delta_{LT}(CS^3_{30}) = 2.25$ and $\Delta_{LT}(EMA_{30}) = 2.6$. In the rest of this section, we evaluate the accuracy of the linear load predictor that is based on these two load trackers.

5.2 Accuracy of the load predictor

The linear load predictor presented in Section 4 is actually a class of load predictors that are based on the linear regression of two load tracker values. Each predictor in this class is characterized by a couple of values: the predicted window k and the past time window q.

First, we aim to choose a good value for the q parameter of the load predictor, because it drives the choice of the second point used to compute the prediction line. To this purpose, in Figures 9 (a) and (b) we show the prediction error as a function of q for two prediction windows equal to $k = 10$ and $k = 30$ seconds, respectively. These figures allow us to conclude that the load predictor based on EMA_{30} performs always better than CS^3_{30}. The reason of its lower prediction error lies in the reduced oscillations of the linear load tracker. When the prediction window is smaller, the accuracy of load predictor based on EMA_{30} is considerably higher than that of load predictor based on CS^3_{30} (almost 3 times when $q = 5$).

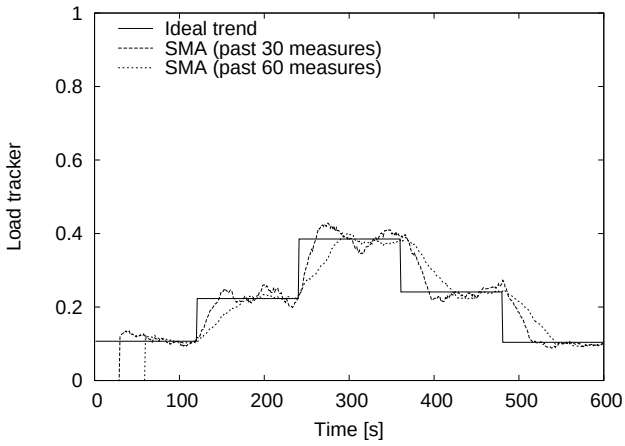
We also note that small values of q (for example, $q = 1$) lead to higher prediction errors Δ_P. The reason is that, with a small value of q, the prediction line takes into account only the very recent trend of the load tracker. If the load tracker is not perfectly smoothed, the prediction error augments. On the other hand, very high values of q lead to prediction lines that weigh past history more than recent history, thus causing another increased prediction error. It seems that the optimal values of q lie in the central area of the Figures 9.

Let us choose two values of q that allow us to obtain a small prediction error that is, $q = 5$ for a prediction window of $k = 10$ seconds, and $q = 15$ for a prediction window of $k = 30$ seconds. Figures 8 show the load tracker values and the predicted values for the two prediction windows. These figures confirm that the load predictor based on an EMA model has a lower prediction error than that based on the cubic spline. The difference is noteworthy for a small prediction window of 10 seconds that is, $\Delta_P = 0.065$ and $\Delta_P = 0.16$ for the EMA-based and the CS-based predictors, respectively.

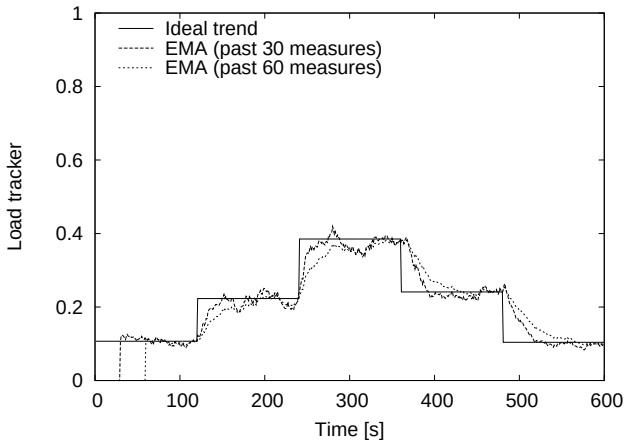
On the other hand, the EMA-based predictor could suffer of higher delays in representing the load trends with respect to the CS model. These delays, that are measured as Δ_{LT} distances, may cause a reduced reactivity to evidence variations in the load conditions. Hence, we can conclude that the best choice for the load predictor should depend on the context to which the decision system is applied. When the highest accuracy is more important than the reactivity, the load predictor based on an EMA load tracker model is preferable; otherwise, we are oriented to recommend a load predictor based on a Cubic Spline load tracker model.

6. RELATED WORK

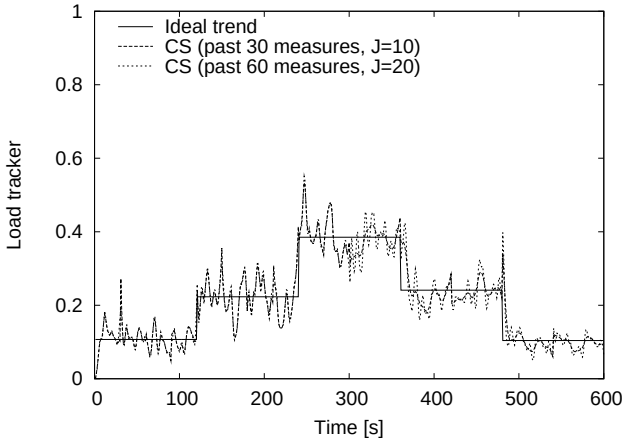
In this work we address the issue of predicting future resource loads under real-time constraints in the context of Internet-based systems, where the prediction is an important task for some runtime decision systems. Previous literature proposes several strategies and algorithms for decision systems such load balancing and load sharing [2, 8, 32], overload and admission control [14, 20, 28],



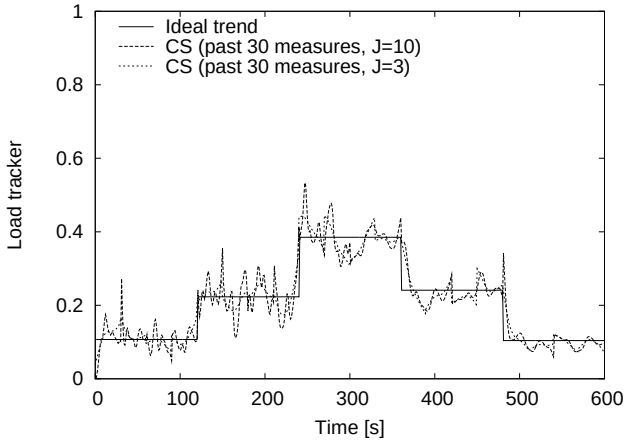
(a) The SMA load trackers



(b) The EMA load trackers

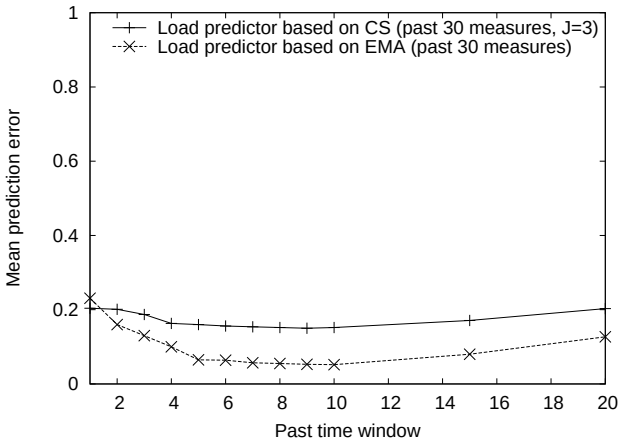


(c) The CS load trackers

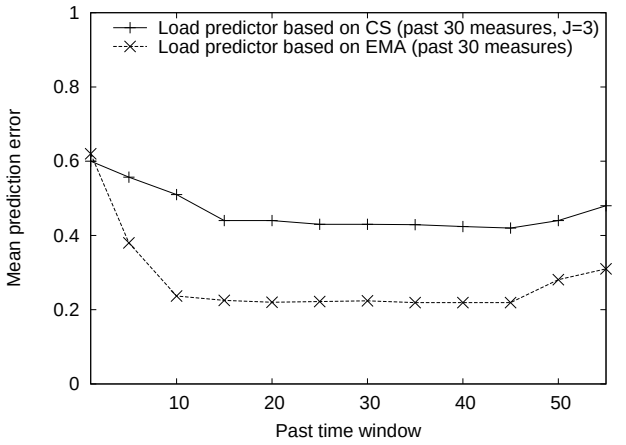


(d) The CS^3_{30} load tracker

Figure 8: Load tracker behaviors with respect to the ideal trend values in the staircase scenario

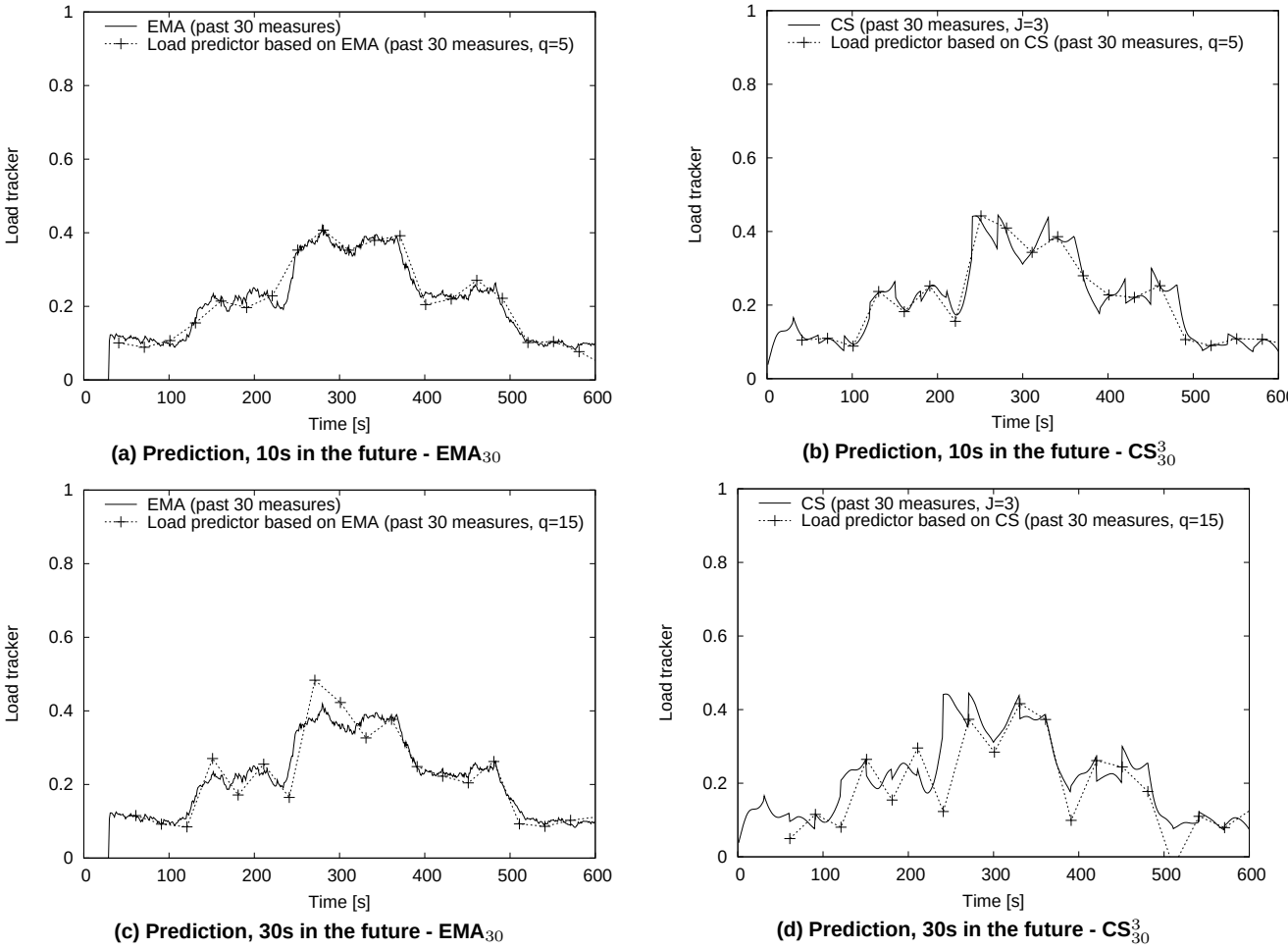


(a) Predicted window: 10 seconds



(b) Prediction window: 30 seconds

Figure 9: Prediction errors in the staircase scenario



(a) Prediction, 10s in the future - $\mathbf{EMA}_{30}$

(b) Prediction, 10s in the future - $\mathbf{CS}^{3}_{30}$

(c) Prediction, 30s in the future - $\mathbf{EMA}_{30}$

(d) Prediction, 30s in the future - $\mathbf{CS}^{3}_{30}$

Figure 10: Load predictors

job dispatching [30] and redirection even at a geographical scale [9], that have the common trait of using the resource measures as the indication of server load. Even the proposals that adopt a control theoretical approach to control the request rate of a system [1, 24] with the intent to prevent possible overload and/or provide guaranteed levels of performance, use direct resource measures (e.g., the CPU utilization or the average Web object response time) as a feedback signal. In this paper, we show that in a context where the resource measures are characterized by high instability and variability, the decisions based on the direct use of these values, may be risky when not completely wrong. In this paper we propose a two-step approach that first aims to represent the load trend of a resource and then it uses this load trend descriptor as the input of the load predictor model.

Other papers [5,17,27,35], in non heavy-tail contexts or in a context where the correlation of subsequent resource measures is high, propose a simple linear model to predict the resource load. However, in the context of heavy-tailed systems such as the one considered in this paper and when resource measures present low correlation, more sophisticated approaches are needed to predict the resource measures, such as ARIMA [38]. These models are more expensive to compute and update, hence they are difficult to use in a real-time setting. Moreover, also the theoretical approach [1, 24] that uses the resource measures to prevent overload, may be inadequate in a heavy-tail scenario where the resource measures have an extremely noisy nature and oscillatory behavior. In this paper, we show that in heavy-tailed systems, the decisions based on the direct use of these values may be risky when not completely wrong.

7. CONCLUSIONS

In this paper, we have investigated the properties that must be possessed by load predictors to support run-time decisions in the context of a multi-tier Web-based system that is subject to heavy-tailed workloads. Our results show that a more regular representation of the load conditions of a resource is necessary because the measures deriving from load monitors show high oscillations. In this context, even the most sophisticated time series models are inadequate for run-time decision systems, because they would require a frequent update of their parameters. For this reason, we propose a two-step approach that first evaluates the load trend through a load tracker function, and then applies the load predictor to the load trend results, instead of working on direct resource measures.

8. 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, M. Colajanni, and M. Nuccio. Scalability of content-aware server switches for cluster-based Web information systems. In *Proc. of 12th Int'l World Wide Web Conf. (WWW2003)*, Budapest, HU, May 2003.

[3] K. Arun, M. S. Squillante, L. Zhang, and J. Poirier. Analysis and characterization of large-scale Web server access patterns and performance. *World Wide Web*, 2(1-2):85–100, Mar. 1999.

[4] P. Barford and M. E. Crovella. Generating representative Web workloads for network and server performance evaluation. In *Proceedings of the Joint International Conference on Measurement and modeling of computer systems (ACM SIGMETRICS 1998/Performance 1998)*, pages 151–160, Madison, WI, July 1998.

[5] Y. Baryshnikov, E. Coffman, G. Pierre, D. Rubenstein, M. Squillante, and T. Yimwadsana. Predictability of Web server traffic congestion. In *Proc. of 10th Int'l Workshop of Web Content Caching and Distribution (WCW05)*, Sep. 2005.

[6] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis Forecasting and Control*. Prentice-Hall, 1994.

[7] H. W. Cain, R. Rajwar, M. Marden, and M. H. Lipasti. An architectural evaluation of Java TPC-W. In *Proc. of the 7th Int.l Symposium on High-Performance Computer Architecture (HPCA2001)*, Nuovo Leone, ME, jan 2001.

[8] V. Cardellini, E. Casalicchio, M. Colajanni, and P. Yu. The state of the art in locally distributed Web-server system. *ACM Computing Surveys*, 2002.

[9] V. Cardellini, M. Colajanni, and P. Yu. Request redirection algorithms for distributed Web systems. *IEEE Trans. Parallel and Distributed Systems*, 14(5), May 2003.

[10] E. Cecchet, A. Chanda, S. Elnikety, J. Marguerite, and W. Zwaenepoel. Performance comparison of middleware architectures for generating dynamic Web content. In *Proc. of 4th Middleware Conference*, Jun 2003.

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

[12] X. Chen and J. Heidemann. Flash crowd mitigation via an adaptive admission control based on application-level measurement. Technical Report ISI-TR-557, USC/Information Sciences Institute, May 2002.

[13] L. Cherkasova and P. Phaal. Session based admission control: a mechanism for improving performance of commercial Web sites. In *Proceedings of the International Workshop on Quality of Service*, London, June 1999.

[14] L. Cherkasova and P. Phaal. Session-based admission control: a mechanism for peak load management of commercial web sites. *IEEE Transactions on Computers*, 51(6), June 2002.

[15] M. E. Crovella, M. S. Taqqu, and A. Bestavros. Heavy-tailed probability distributions in the World Wide Web. In *A Practical Guide To Heavy Tails*, pages 3–26. Chapman and Hall, New York, 1998.

[16] M. Dahlin. Interpreting stale load information. *IEEE Trans. Parallel and Distributed Systems*, 11(10):1033–1047, Oct. 2000.

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

[18] R. C. Dodge, D. A. Menascé, and D. Barbará. Testing e-commerce site scalability with TPC-W . In *Proc. of 2001 Computer Measurement Group Conference*, Dec 2001.

[19] R. L. Eubank and E. Eubank. *Non parametric regression and spline smoothing*. Marcel Dekker, 1999.

[20] D. Ferrari and S. Zhou. An empirical investigation of load indices for load balancing applications. In *Proceedings of Performance 1987*, pages 515–528, North-Holland, The Netherlands, 1987.

[21] A. G. Ganek and T. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, 2003.

[22] S. Godard. Sysstat: System performance tools for the Linux OS, 2004. http://perso.wanadoo.fr/sebastien.godard/.

[23] 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 11th Int'l World Wide Web Conference (WWW2002)*, May 2002.

[24] A. Kamra, V. Misra, and E. M. Nahum. Yaksha: a self-tuning controller for managing the performance of 3-tiered sites. In *Proceedings of Twelfth International Workshop on Quality of Service (IWQOS2004)*, pages 47–56, June 2004.

[25] J. O. Kephart and D. M. Chess. The vision of Autonomic Computing. *IEEE Computer*, 36(1):41–50, Jan. 2003.

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

[27] Y. Lingyun, I. Foster, and J. M. Schopf. Homeostatic and tendency-based CPU load predictions. In *Parallel and distributed processing Symposium, 2003*, pages 9–, 2003.

[28] M. Mitzenmacher. How useful is old information. *IEEE Trans. Parallel and Distributed Systems*, 11(1):6–20, Jan. 2000.

[29] MySQL database server, 2005. – http://www.mysql.com/.

[30] Network Weather Service, 2005. – http://nws.cs.ucsb.edu/ewiki/.

[31] T. Oetiker. Rrdtool: a system for displaying time-series data, 2004. http://oss.oetiker.ch/rrdtool/.

[32] 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 *Proceedings of the 8th ACM Conference on Architectural Support for Programming Languages and Operating Systems*, pages 205–216, San Jose, CA, Oct. 1998.

[33] D. J. Poirier. Piecewise regression using cubic spline. *Journal of the American Statistical Association*, 68(343):515–524, Sep. 1973.

[34] procps - the /proc file system utilities, 2005. http://procps.sourceforge.net/.

[35] A. Sang and S. Li. A predictability analysis of network traffic. In *Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM2000)*, pages 342–351, 2000.

[36] The Tomcat Servlet Engine, 2005. – http://jakarta.apache.org/tomcat/.

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

[38] N. Tran and D. Reed. Automatic ARIMA time series modeling for adaptive I/O prefetching. *IEEE transaction on parallel and distributed systems*, 15(4):362–377, Apr. 2004.

[39] J. Wildstrom, P. Stone, E. Witchel, R. Mooney, and M. Dahlin. Towards self-configuring hardware for distributed computer systems. In *Proc. of the Second International Conference on Autonomic Computing (ICAC2005)*, June 2005.

[40] G. Wolber and I. Alfy. Monotonic cubic spline interpolation. In *Computer Graphics International*, pages 188–195, Canmore, Alta., Canada, July 1999.