Integrated Estimation and Tracking of Performance Model Parameters with Autoregressive Trends

Tao Zheng
Dept. of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada
zhengtao@sce.carleton.ca

Marin Litoiu
Dept. of Information Science
York University
Toronto, Ontario, Canada
mlitoiu@yorku.ca

Murray Woodside
Dept. of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada
cmw@sce.carleton.ca

ABSTRACT
Adaptive management of a software service system can take advantage of a performance model which can predict the effect of proposed changes, before they are deployed. As the system varies over time the model parameters can be tracked by an estimator such as a Kalman Filter, so that decisions can be updated. The filter is valuable when parameters are "hidden" and cannot be directly measured without excessive cost (as is usually the case for the CPU time of a service). Because there may be significant delays in some management control actions (especially in deploying a new replica of a service), it is also important to be able to predict the changes ahead somewhat in time, that is, to predict the trends. The trend predictor itself needs to be estimated from observed trends in the model parameters. This work uses an autoregressive model for trend prediction and integrates it with the parameter estimator, in a single Kalman Filter, using auxiliary states for the parameter evolution process. This paper describes how the trend model is constructed, and evaluates its effectiveness. It compares the overall performance predictions to a simpler trend predictor using linear extrapolation of the fitted parameter time-series, which turns out to be almost as good. The approach is validated on a real system running a benchmark web application.

Categories and Subject Descriptors
D.4.8 [**Performance**]: *trend prediction, tracking, queuing theory, simulation*

General Terms
Experimentation, Measurement, Performance and Theory.

Keywords
Performance model, Layered Queueing, Autoregressive model, Tracking, Estimation, Extended Kalman filter and Simulation

ICPE'11, March 14–16, 2011, Karlsruhe, Germany.

1. INTRODUCTION
Large-scale service systems are dynamic and must adapt to maintain adequate quality of service (QoS) [14][4][11] under changing operating conditions. Changes occur in traffic volumes and in the operational profile of services, due to shifting needs of users and to the replacement of some services by alternatives in a service-oriented architecture. Adaptation may require provisioning of services, allocation of storage or communications resources, or re-tuning of a server. Adaptation decisions may be based directly on system measures, with a change triggered by the passing of a threshold, or by considerations involving a performance model [15][16], which is assumed here. The performance model is initially derived from the system specification (e.g. [23]) or from other knowledge of the system.

A performance model provides a powerful means to evaluate complex decisions, such as trading off QoS in one service against another, or considering additional buffer space versus additional processing power. However, in a dynamic system, the performance model can quickly become outdated if it is not adapted automatically to keep up with the changes in the system parameters or structure. This requires a run time observation of the changes in system parameters. However, some performance parameters may not be observed directly because it is too expensive or too disruptive (an example is the CPU demand of an operation). Because they are not observed, they called here *hidden*. In [22][24] such parameters were estimated indirectly through their effect on the performance calculation, by an Extended Kalman Filter (EKF) estimator, which is a kind of Bayesian estimator [3][10]. Combining direct observation with estimation of hidden parameters, as described in [15][22][24], yields an up to date *tracked* model. Management using this model, to make decisions by evaluating the system performance under different options was shown to be effective in [24].

In many practical situations, there is a substantial delay in applying some of the control decisions, particularly in provisioning servers for a clustered application [6]. In those situations, it is desirable to also *predict* the performance forward in time and thus improve the quality of control. Predictions with no parameter trend model, as in [22], are simply the value of the current estimate, constant into the future.

For parameters which are directly measurable (not hidden) there are simple approaches. In [24][28] the arrival rate of requests (which was measured directly) was predicted by fitting a straight-line trend to the measured values, which significantly enhanced the

model predictions for future times. A more sophisticated predictor such as an autoregressive model [2] could also be applied. In [7] and [18], diurnal and weekly variations of workload were represented by a "trend" term derived from recent observations (a different meaning for "trend"). A term is added from a predictive time-series model fitted to the residual. This approach is excellent but it cannot predict hidden parameters and it responds very slowly (over weeks) to changes in user behaviour.

This paper describes a comprehensive method for predicting the performance trends of a system with hidden parameters. It combines a performance model to estimate performance, with an autoregressive model to predict hidden parameters ahead in time, with estimation and prediction via an Extended Kalman Filter.

An AR model predicts a future value of a time-series as a linear or non-linear function of its own past values. The estimation of an AR model for an observed process is well documented in [2]. It can be based on regression techniques, or on a tracking filter which can update the AR model over time. An example of estimation of an AR model by a Kalman Filter [27] describes tracking changes in gear vibrations, which were observed by sensors.

In the present work the estimators for the AR trend model and for the performance model parameters are combined in a single EKF. The combination greatly improves the prediction of future values, as will be shown. The approximations that justify the combination depend on the adequate separation of the two time scales, one for measuring performance, and the second for time variation of the parameters.

- Over a measurement interval (called here a *step*), we assume the system properties are approximately invariant, with stationary random processes governing arrivals, execution demands and user behaviour. This justifies using a stationary performance model for estimation and prediction, conditional on values for the parameters.
- Over a longer time scale of several steps, the time-varying parameters of the system are assumed to be represented by a *stationary autoregressive (AR) process* and we will call this time-variation the *local trend* (or simply trend) of the attributes (this trend is in general not linear in time).

1.1 Related work

Trend prediction has an enormous literature; a recent survey is [8]. Autoregressive (AR) and related predictors [2] are the most successful, and are fitted to measured data by various methods, including an EKF [27]. Similarly the use of the EKF to estimate hidden states in dynamic systems has a substantial literature. The authors are not aware of any previous attempt to combine these, as in the present work: the EKF estimates the parameters of the performance part, and also the dynamic trend parameters of the AR part of the model.

However the resulting estimator is similar to many studies in dynamic systems in which an EKF is used to estimate the parameters of the system as well as its states. Compared to such studies, the performance model parameters are the states, the AR submodel represents the dynamics, and the performance model is an "output function" computing outputs (performance measures) from the states.

An AR trend model is an approximation. Even if the actual process is not AR, a tracked AR model may be adequate. For example long-range dependent time series have been shown to be common in web services requests and responses [1]. There is some evidence that an AR approximation which adapts to the process may be satisfactory. In [7] the coefficients of an AR model for arrival rates in long-range-dependent traffic were tracked, and gave adequate predictions. In [26], a useful AR model was generated for a long-range dependent request rate, expressing the rate as a linear function of its immediate past values, with parameters that depended on the local average rate. In [13] a fixed function for diurnal and weekly variations was found, and an AR model fitted successfully to the residuals. It was capable of predicting over a few steps of 5 minutes each, but at longer prediction horizons the long-range dependency become important and the predictions had low accuracy. Reference [19] considers the selection of the best order for an autoregressive model.

2. MODELS OF CHANGES IN A SERVICE SYSTEM

A time-varying performance model will be written as a vector function $\mathbf{y}(k) = \mathbf{h}(\mathbf{x}(k))$, for the kth time interval, with changes occuring between time intervals. In interval k the elements of $\mathbf{y}(k)$ may be average response times, rates or throughputs, and resource utilizations; the elements of $\mathbf{x}(k)$ may be average performance parameter values such as CPU demands of service operations, mean storage operations required for a service operation, and arrival rates of service requests, or they may be parameters of probability distributions for these and similar quantities. The parameter vector $\mathbf{x}(k)$ changes with time and $\mathbf{h}$ is a non-linear function representing a performance model such as a Queuing Network Model (QNM) or Layered Queuing Model (LQM). Within the kth time interval we assume that the system is effectively in a steady state, with a constant parameter vector $\mathbf{x}(k)$.

To model the random variation of $\mathbf{x}(k)$ over time, we assume that $\mathbf{x}(k)$ evolves according to some law which in general could be written $\mathbf{x}(k+1) = \boldsymbol{\phi}(\mathbf{x}(k), \mathbf{x}(k-1), ..., \mathbf{x}(k-\mu+1), \boldsymbol{\xi}_\mathbf{x}(k))$ where $\boldsymbol{\xi}_\mathbf{x}$ is an independent random vector and μ is the maximum lag. This defines $\mathbf{x}$ as a discrete-time random process. Here we will consider each element x_i of $\mathbf{x}$ to be governed by an approximate time-varying auto-regressive model of order μ

$$x_i(k+1) = \sum_{\tau=1}^{\mu} g_{\tau+\mu(i-1)}(k)\, x_i(k+1-\tau) + \xi_{\mathbf{x},i}(k), \quad i = 1, \ldots, n \qquad (1)$$

where the AR weights are gathered in a vector $\mathbf{g}$, in which the ith group of μ elements gives the AR weights for past values of x_i.

The trend model may change over time, and to represent this $\mathbf{g}$ is modeled by a random walk defined by:

$$\mathbf{g}(k+1) = \mathbf{g}(k) + \boldsymbol{\xi}_\mathbf{g}(k) \qquad (2)$$

where $\boldsymbol{\xi}_\mathbf{g}$ is an independent normal process. Together, $\mathbf{x}$ and $\mathbf{g}$ are combined in a state vector $\boldsymbol{\alpha}$, of size 2L, where L=nμ:

$$\boldsymbol{\alpha}^T(k) = (\mathbf{x}^T(k), \mathbf{x}^T(k-1), \ldots \mathbf{x}^T(k-\mu+1), \mathbf{g}^T(k)) \qquad (3)$$

and the state-transition equations (1) and (2) can be expressed as

$$\boldsymbol{\alpha}(k+1) = \mathbf{f}(\boldsymbol{\alpha}(k)) + \boldsymbol{\xi}(k)) \qquad (4)$$

where $\boldsymbol{\xi}(k)^T = (\boldsymbol{\xi}_\mathbf{x}(k)^T, \mathbf{0}, \mathbf{0}, ..., \mathbf{0}, \boldsymbol{\xi}_\mathbf{g}(k)^T)$.

The AR model (1) is expressed by the first n state equations as:

$$\alpha_i(k+1) = \Sigma_{\tau=1 \text{ to } \mu}\, \alpha_{\mu(n+i-1)+\tau}(k)\, \alpha_{n(\tau-1)+i}(k);\ i = 1, \ldots, n$$

and the time-delayed values of **x** are included by equations of the form $\alpha_{i+n\tau}(k+1) = \alpha_{i+n(\tau-1)}(k)$, for $i = 1, \ldots, n$ and $\tau = 1, \ldots, \mu$

Then the assumed time evolution of the performance model is described by the random process (4) together with:

$$\mathbf{x}(k+1) = \text{Sel}(\boldsymbol{\alpha}(k+1)) = \text{the first } n \text{ elements of } \boldsymbol{\alpha}(k+1) \quad (5)$$

$$\mathbf{y}(k+1) = \mathbf{h}(\mathbf{x}(k+1)).$$

Sel($\boldsymbol{\alpha}$) is a selector function that returns the first n elements of any larger vector. $\mathbf{h}(\mathbf{x})$ is the performance model calculation. Equations (4), (5) describe a *nonlinear* model because **f**(.) and **h**(.) are nonlinear functions of $\boldsymbol{\alpha}(k)$.

2.1 State-transition Jacobian Matrix

The state-transition function in (4) is bilinear, that is it is linear in each component of $\boldsymbol{\alpha}(k)$. Thus it can be written in the form

$$\boldsymbol{\alpha}(k+1) = \mathbf{F}(\boldsymbol{\alpha}(k))\, \boldsymbol{\alpha}(k) + \boldsymbol{\xi}(k),$$

with a matrix $\mathbf{F}(\boldsymbol{\alpha}(k))$ that has terms that are linear in $\boldsymbol{\alpha}$. The $2L$ x $2L$ matrix $\mathbf{F}(\boldsymbol{\alpha}(k))$ is (remember, $L = n\mu$):

$$\mathbf{F}(\boldsymbol{\alpha}(k))= \left[\begin{array}{cccc|c} \mathbf{G}^*_1(\boldsymbol{\alpha}(k)) & \ldots\ldots & & \mathbf{G}^*_\mu(\boldsymbol{\alpha}(k)) & \mathbf{0}_{LxL} \\ \mathbf{I}_{nxn} & \mathbf{0}_{nxn} & \ldots\ldots & \mathbf{0}_{nxn} \quad \mathbf{0}_{nxn} & \\ \mathbf{0}_{nxn} & \mathbf{I}_{nxn} & \ldots\ldots & \mathbf{0}_{nxn} \quad \mathbf{0}_{nxn} & \\ & \ldots\ldots\ldots\ldots & & & \\ \mathbf{0}_{nxn} & \mathbf{0}_{nxn} & \ldots\ldots & \mathbf{I}_{nxn} \quad \mathbf{0}_{nxn} & \\ \hline \multicolumn{4}{c|}{\mathbf{0}_{LxL}} & \mathbf{I}_{LxL} \end{array}\right] \quad (6)$$

in which the upper left quadrant of **F** has $\mu x \mu$ blocks of size nxn and the submatrices $\mathbf{G}^*_i(\boldsymbol{\alpha}(k))$ depend on the AR coefficients for $x_i(k+1)$). Putting in the equivalent AR parameters **g**, we see that:

$$\mathbf{G}^*_i\,(\boldsymbol{\alpha}(k)) = \text{diag}(g_{\mu(i-1)+\tau}(k))\ ,\ \tau = 1, \ldots, \mu, \text{ and } i = 1, \ldots, n \quad (7)$$

The estimator below uses the Jacobian or *sensitivity* matrix of $\mathbf{f}(\boldsymbol{\alpha}(k))$, denoted by $\mathbf{A}(\boldsymbol{\alpha}(k))$ but for brevity written as $\mathbf{A}(k)$:

$$\mathbf{A}(k) = \partial \mathbf{f}(\boldsymbol{\alpha}(k))/\partial \boldsymbol{\alpha}, = \mathbf{F}(\boldsymbol{\alpha}(k)) + \partial \mathbf{F}(\boldsymbol{\alpha}(k))/\partial \boldsymbol{\alpha}. \quad (8)$$

The matrix $\mathbf{A}(k)$ is obtained from (8), as:

$$\mathbf{A}(k) = \left[\begin{array}{ccccccc} \mathbf{G}^*_1(k) & \mathbf{G}^*_2(k) & & \ldots\ldots & \mathbf{G}^*_\mu(k) & \mathbf{X}^{**}(k) \\ \mathbf{I}_{nxn} & \mathbf{0}_{nxn} & \ldots\ldots & \mathbf{0}_{nxn} & \mathbf{0}_{nxn} & \mathbf{0}_{(L-1)xL} \\ \mathbf{0}_{nxn} & \mathbf{I}_{nxn} & \ldots\ldots & \mathbf{0}_{nxn} & \mathbf{0}_{nxn} & \\ & & \ldots\ldots\ldots\ldots & & & \\ \mathbf{0}_{nxn} & \mathbf{0}_{nxn} & \ldots\ldots & \mathbf{I}_{nxn} & \mathbf{0}_{nxn} & \\ \mathbf{0}_{Lxn} & \mathbf{0}_{Lxn} & \ldots\ldots & \mathbf{0}_{Lxn} & \mathbf{I}_{LxL} & \end{array}\right] \quad (8)$$

where $\mathbf{X}^{**}$ is a nxL matrix, all zeros except for these entries:

$$\mathbf{X}^{**}_{i,j}(k) = \alpha_{\mu(i-1)+j}(k) = x_i(k-j+\mu(i-1))\ ,\quad i = 1, \ldots, n;\ \mu(i-1)+1 < j < \mu i$$

That is, the ith row contains a subvector of the μ-1 past values of x_i, which are the derivatives of $x_i(k)$ with respect to the AR weights.

2.2 Some Special cases

No Trend: Without dependence on the past, there is no **g**. Each x_i is an independent random walk, $x_i\ (k+1)= x_i(k) + \xi_i(k)$, making $\mathbf{A}(k) = \mathbf{I}$ (as used in [28]). This assumes the least about the nature of the parameter change from one step to the next, and in particular it does not model any trends that extend over multiple steps.

One Parameter, First Order AR Forecast: For one estimated parameter $x(k)$, a first-order AR model $x(k+1) = g(k)\, x(k) + \xi_1(k)$, and a random walk for $g(k)$, the state model is:

$$\alpha_1(k+1) = x(k+1) = g(k)\, x(k) + \xi_1(k) = \alpha_2(k)\, \alpha_1(k) + \xi_1(k)$$

$$\alpha_2(k+1) = g(k+1) = g(k) + \xi_2(k) = \alpha_2(k) + \xi_2(k)$$

The sensitivity matrix **A** is:

$$\mathbf{A} = \begin{bmatrix} \alpha_2 & \alpha_1 \\ 0 & 1 \end{bmatrix}$$

One Parameter, Second Order AR Forecast: For a single estimated parameter and trend model $x(k+1) = g_1(k)\, x(k) + g_2(x(k-1)) + \xi_1(k)$, the state model is:

$$\alpha_1(k+1) = x(k+1) = g_1(k)\, x(k) + g_2(k)\, x(k-1) + \xi_1(k)$$

$$= \alpha_3(k)\, \alpha_1(k) + \alpha_4(k)\, \alpha_2(k) + \xi_1(k)$$

$$\alpha_2(k+1) = x(k) = \alpha_1(k)$$

$$\alpha_3(k+1) = g_1(k+1) = g_1(k) + \xi_3(k) = \alpha_3(k) + \xi_3(k)$$

$$\alpha_4(k+1) = g_2(k+1) = g_2(k) + \xi_4(k) = \alpha_4(k) + \xi_4(k)$$

The sensitivity matrix $\mathbf{A}(k)$ is:

$$\mathbf{A}(k) = [\alpha_3(k)\ \ \alpha_4(k)\ \ \alpha_1(k)\ \alpha_2(k)) = \begin{array}{cccc} [g_1(k) & g_2(k) & x(k) & x(k-1) \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1\ ] \end{array} \quad \begin{array}{cccc} & & & \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1\ ] \end{array}$$

3. TRACKING FILTER WITH AR PARAMETER FORECASTING

The tracking filter assumes a known covariance matrix $\mathbf{Q}(k)$ for $\boldsymbol{\xi}(k)$ (see [24] for how to determine **Q**). The measured performance values $\mathbf{z}(k)$ can reasonably be assumed to include unbiased independent normal errors of measurement $\mathbf{v}(k)$ with a known covariance matrix $\mathbf{R}(k)$ (found from sample measurements):

$$\mathbf{z}(k) = \mathbf{y}(k) + \mathbf{v}(k), \quad \text{Cov}(\mathbf{v}(k)) = \mathbf{R}(k) \quad (9)$$

R may not in fact be constant in time, this is discussed in [24].

Estimation of **x** and **y** requires an Extended Kalman filter (EKF) because the state change **f** in equations (4)(5) and the output function **h** are both nonlinear. The EKF is described in [3][10]. The EKF estimator for $\mathbf{y}(k)$ is given by the following steps, in which $\hat{\boldsymbol{\alpha}}(k)$ and $\hat{\mathbf{y}}(k)$ are the current estimates of the state $\boldsymbol{\alpha}$ and

the performance value **y**, found after processing the measurements **z**(k).

Step 1. project the state one step ahead and select out the projected performance model parameters:

$$\hat{\boldsymbol{\alpha}}^{-}(k+1) = \mathbf{f}(\hat{\boldsymbol{\alpha}}(k)) = \mathbf{F}(k)\,\hat{\boldsymbol{\alpha}}(k)$$

$$\hat{\mathbf{x}}^{-}(k+1) = \mathrm{Sel}(\hat{\boldsymbol{\alpha}}^{-}(k+1)) \quad \textit{(select the first n elements)}$$

Step 2. project $\mathbf{P}(k+1)$, the estimated covariance matrix for the estimates of $\boldsymbol{\alpha}$:

$$\mathbf{P}^{-}(k+1) = \mathbf{A}(k+1)\mathbf{P}(k+1)\mathbf{A}^{\mathrm{T}}(k+1) + \mathbf{Q}(k+1)$$

Step 3. compute the Kalman gain $\mathbf{K}(k+1)$ as:

$$\mathbf{K}(k+1) = \mathbf{P}^{-}(k+1)\mathbf{H}(k+1)^{\mathbf{T}}(\mathbf{H}(k+1)\mathbf{P}^{-}(k)\mathbf{H}(k+1)^{\mathbf{T}} + \mathbf{R}(k+1))^{-1}$$

Step 4. using the measurement $\mathbf{z}(k+1)$, correct the state vector, select the parameters, and find the performance prediction:

$$\hat{\boldsymbol{\alpha}}(k+1) = \hat{\boldsymbol{\alpha}}^{-}(k+1) + \mathbf{K}(k+1)(\mathbf{z}(k+1) - \mathbf{h}(\hat{\mathbf{x}}^{-}(k+1)))$$

$$\hat{\mathbf{x}}(k+1) = \mathrm{Sel}(\hat{\boldsymbol{\alpha}}(k+1)) \quad \textit{(parameter estimates)}$$

$$\hat{\mathbf{y}}(k+1) = \mathbf{h}(\hat{\mathbf{x}}(k+1)) \quad \textit{(performance calculations)}$$

Step 5. correct the error covariance $\mathbf{P}_{k+1}$:

$$\mathbf{P}(k+1) = (\mathbf{I} - \mathbf{K}(k+1)\mathbf{H}(k+1))\,\mathbf{P}^{-}(k+1)$$

In these steps,

- the matrix $\mathbf{A}(k)$ is the Jacobian of $\mathbf{f}(k,\boldsymbol{\alpha})$ as derived above, evaluated at $\hat{\boldsymbol{\alpha}}(k)$.
- the matrix $\mathbf{H}(k)$ is the Jacobian of $\mathbf{h}(\mathbf{x})$ evaluated at $\hat{\mathbf{x}}(k)$.

The optimal predictor $\hat{\mathbf{x}}(k+j|k)$ for values j steps ahead from step k applies the projection step 1, j times [10]:

$$\hat{\boldsymbol{\alpha}}^{-}(k+j|k) = \mathbf{F}(k)\,\hat{\boldsymbol{\alpha}}^{-}(k+j\text{-}1|k), j = 1, 2, 3...$$

$$\hat{\boldsymbol{\alpha}}^{-}(k|k) = \mathbf{F}(k)\,\hat{\boldsymbol{\alpha}}^{-}(k),$$

$$\hat{\mathbf{x}}^{-}(k+j|k) = \mathrm{Sel}(\hat{\boldsymbol{\alpha}}^{-}(k+j|k)) \qquad (10)$$

Without the trend modeling, $\hat{\mathbf{x}}^{-}(k+j|k) = \hat{\mathbf{x}}(k)$.

4. A TRACKING EXPERIMENT

Our research uses a layered queueing network (LQN) performance model, a kind of extended queueing network which explicitly includes attributes of service systems. Figure 1 shows a diagram of an example of an LQN. There are hardware servers or *processors* (shown as ovals) and software servers called *tasks* (shown as bold rectangles, with attached rectangles called *entries* describing their classes of service). Tasks and processors are labeled to show their multiplicity, e.g. (100), entries are labeled to show their processor demand, e.g. [50 ms], and arrows indicating requests made by one entry for service by another are labeled by the mean number of requests, e.g. (0.4). LQNs are described in [17][21] and references to applications may be found in [5].

The tracking parameters are those three parameters, i.e. $\mathbf{x}^{\mathrm{T}} = (S_d, S_w, Z)$ and $n = 3$. The measurements include the response times and cpu utilizations of the three related components, and the throughput of the system.

The performance measures taken from the simulation were:

z_1 = response time of retrievePage, as seen by the Users

z_2 = response time of dbOp

z_3, z_4 = processor utilizations of WebServer and Database, respectively

$z_{5,6}$ = throughput of WebServer and Database, respectively

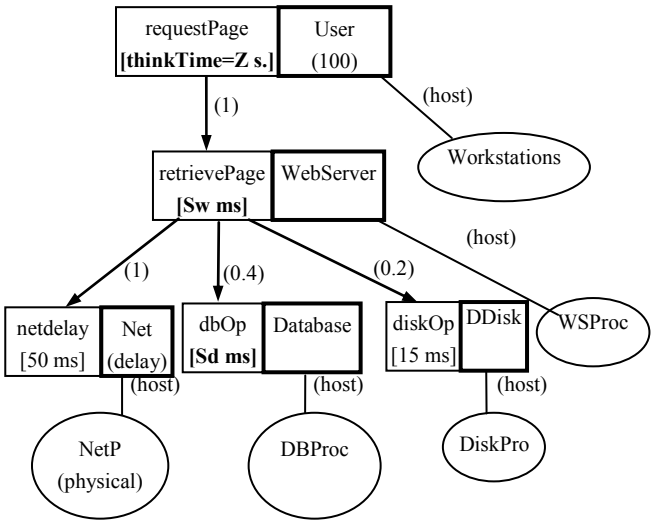

Figure 1. The LQN model of a web server system

and these were measured as averages over a step. The step length was chosen so the estimation errors in the mean values gave a 95% confidence of no more than 5% of the mean value, across all the estimates in a trial experiment.

4.1 Case 1: Periodic Trend Variation

The web server system shown in Fig. 1 was simulated with a periodic variation imposed on the three parameters which are shown by symbols: the user think time (Z) and the execution demands of the web server (S_w) and the database server (S_d).

$$\mathbf{x} = (S_d, S_w, Z) \qquad (10')$$

$$S_d = 20 + 10\cos(k\pi/50) \text{ ms}$$

$$S_w = 25 + 15\sin(k\pi/50 - \pi/2) \text{ ms}$$

$$Z = 5000 + 1000\sin(k\pi/250) \text{ ms}$$

This made S_d vary from 10 to 30 and S_w vary from 10 to 40 in a cycle of 100 steps, and Z to vary from 4000 to 6000 with a cycle of 500 steps. The CPU utilization of the WebServer is up to 0.7.

The average values of measurements have a sampling error due to the finite step length, which gives the measurement errors $\mathbf{v}(k)$ in (9). The variance matrix **R** was made a diagonal matrix (assuming independent sampling errors) and step size was chosen so its

entries correspond to a 95% confidence interval which is about ±5% of the mean:

$$\mathbf{R}_{ii} = ((95\% \text{ confidence interval half-width})/1.96)^2 \quad (11)$$

A first-order autoregressive model was assumed for the time-variation of $\mathbf{x}(k) = (x_1(k), x_2(k), x_3(k))^T$, of the form

$$x_i(k+1) = g_i(k)\, x_i(k) + \xi_i(k), \quad i = 1, 2, 3$$

Then the state-transition matrix **F** is

$$\mathbf{F} = \begin{bmatrix} \mathbf{G^*_{3x3}} & \mathbf{0_{3x3}} \\ \mathbf{0_{3x3}} & \mathbf{I_{3x3}} \end{bmatrix}$$

where

$$\mathbf{G^*_{3x3}} = \text{diag}\,(g_1(k), g_2(k), g_3(k))$$

During an increasing trend of $x_i(k)$, $g_i(k)$ will be estimated as greater than 1, and in a decreasing trend, less than 1.

In the tracking filter, the covariance matrix $\mathbf{Q}(k)$ was a diagonal matrix set according to:

$$\mathbf{Q}_{ii} = (Qfac * x_i(0))^2 \quad (12)$$

with default value of $Qfac = 0.02$. $x_i(0)$ is the initial value of the i-th parameter, where $\mathbf{x}^T(0) = [30, 10, 5000]$. This models a standard deviation of ξ_i which is about 0.02 times the initial value of parameter x_i. It prepares the filter for substantial parameter changes, but significantly less than the current parameter value. Various values of *Qfac* were used to evaluate their impact.

The output function $\mathbf{h}(\mathbf{x})$ was the LQN solution for the model with parameter values **x**. It was found by applying the LQNS solver [5] to the LQN with the estimated parameters.

The tracking and prediction errors found by simulation experiments will be summarized using the measures

- *MARE* (mean absolute relative error), for tracking and
- *fMARE* (forwarding looking *MARE*), for prediction,

We first define the j-steps-ahead prediction of the parameters **x** and the performance **y** as follows:

$\hat{\boldsymbol{\alpha}}(k+j|k)$ = prediction of $\boldsymbol{\alpha}(k+j)$ based on data up to step k.

$$= \mathbf{F}(k)\, \hat{\boldsymbol{\alpha}}(k+j-1|k)$$

$$\hat{\boldsymbol{\alpha}}(k+0|k) = \hat{\boldsymbol{\alpha}}(k)$$

$$\hat{\mathbf{x}}(k+j|k) = \text{Sel}(\hat{\boldsymbol{\alpha}}(k+j|k))$$

$\mathbf{y}(k+j|k)$ = prediction of $\mathbf{y}(k+j)$ based on data up to step k.

$$= \mathbf{h}(\hat{\mathbf{x}}(k+j|k))$$

For errors in performance prediction over a trace of K steps, $MARE(z_i)$ for the i-th measurement component z_i is defined as

$$MARE\,(z_i) = (1/K)\sum_{k=1}^{K} |\, y_i(k) - z_i(k)\,| \,/\, z_i(k)$$

For j-steps-ahead predictions, fMARE(z_i, j) is defined as

$$fMARE\,(z_i, j) = (1/K)\sum_{k=1}^{K} |\, y_i(k+j\,|\,k) - z_i(k+j)\,| \,/\, z_i(k+j)$$

Similarly, for the model parameters x_i:

$$MARE\,(x_i) = (1/K)\sum_{k=1}^{K} |\, \hat{x}_i(k) - x_i(k)\,| \,/\, x_i(k)$$

$$fMARE\,(x_i, j) = (1/K)\sum_{k=1}^{K} |\, \hat{x}_i(k+j\,|\,k) - x_i(k+j)\,| \,/\, x_i(k+j)$$

4.2 Results for Slow Sinusoidal Variation

The system was simulated with the slow sinusoidal variation of parameters as in Eq (10') above, and the three parameters and three AR coefficients were tracked. Figure 2 shows a trace of the S_d, and S_w parameters and their tracked values. Tracking is excellent. Remember that, although the parameter variation is deterministic, the measurements have errors due to statistical sampling, so the estimator must filter these errors.

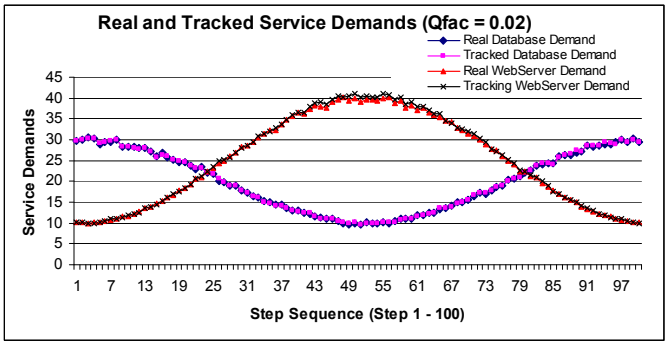

Figure 2. Real and tracked service demands (Qfac = 0.02)

Figure 3 and 4 show 3-step predictions of service demands and the user response time against the future simulated values. Clearly the errors are small and the predictions are good, with slightly larger errors as the trend reverses, at the peak and trough of the curve.

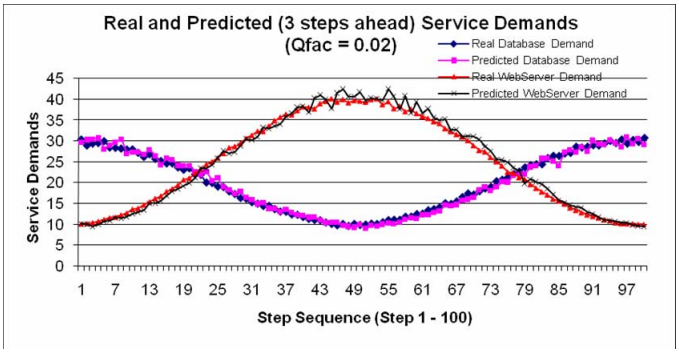

Figure 3. Real and predicted service demands (Qfac = 0.02)

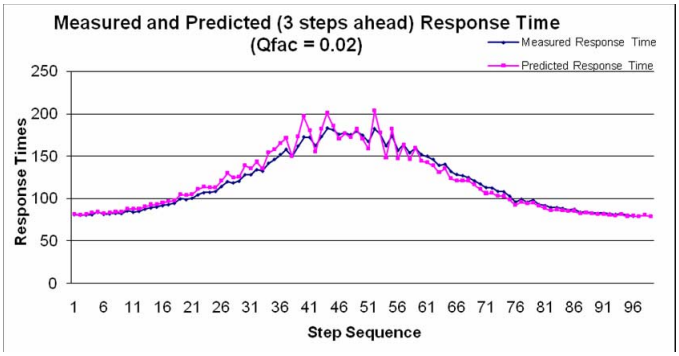

Figure 4. Measured and predicted response time (Qfac = 0.02)

Figure 5 shows the estimates found for the autoregressive coefficients *g* as well, showing how they oscillate around 1.0 as the trend goes from increasing to decreasing and back.

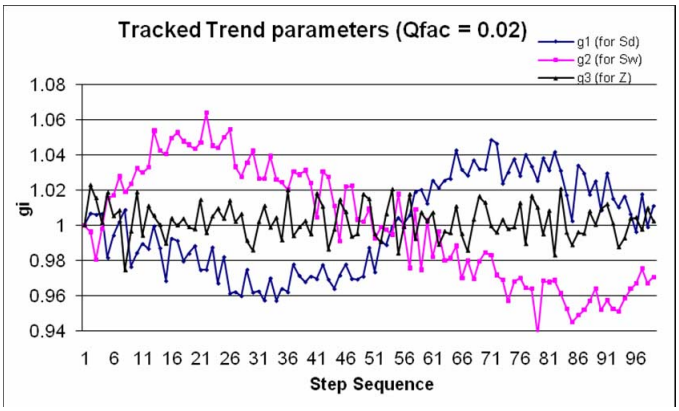

Figure 5. Tracked trend parameters g_i (Qfac = 0.02)

Table 1 shows quantitative results (*MARE* and *fMARE*) for the tracking and prediction errors. The response time error (MARE) is less than 1% for tracking, and for 3-step prediction (fMARE) the trend estimation improves the quality nearly 20% (from nearly 6.7% error without trend tracking, down to 5.5%). Thus, trend tracking is effective. The parameter tracking accuracies show similar relationships, with smaller relative errors. The table also shows that the improvement for 3-step prediction is more significant than that for 1-step prediction.

An alternative way to predict the performance for some steps ahead is to extrapolate from the recent filter estimates with no prediction of the hidden parameters. This was used effectively to predict, in lab-scale experiments reported in [24]. We used linear regression over the 10 preceding arrival rate estimates to obtain the prediction for the arrival rates with an external predictor. These predicted arrival rates were then used as input for the LQM, that is, the function **h**(**x**), to calculate the response time for 3 steps ahead. The errors of the external predictor and of integrated one (proposed by this paper) are shown in Table 2. By using our proposed integrated predictor, the prediction errors are reduced by about 50%, slightly smaller for one step, larger for three steps (Table 2, line 1). Thus on relatively clean data and slow parameter variation, the separate regression is worse. It is considered again in Section 4.5 and 4.6 below, for other cases.

TABLE 1. The Mean Absolute Relative Errors of estimation and prediction, for Qfac=0.02

What is Est./Pred		fMARE 1 step ahead			fMARE 3 steps ahead		
	MARE	no trend tracking	trend tracking	benefit %	no trend tracking	trend tracking	benifit %
User Resp.	0.0077	0.03039	0.02900	4.5872	0.06753	0.05419	19.7459
S_d	0.0053	0.02734	0.02268	17.0690	0.06808	0.03709	45.5273
S_w	0.0140	0.03489	0.02316	33.6362	0.08776	0.03984	54.6021

TABLE 2. fMARE for errors in user response time predicted by linear regression over the 10 previous estimates

	Compare: Integrated estimator vs separate regression (Qfac = 0.02)		
Case and Section	fMARE, 1 step	fMARE, 2 steps	fMARE, 3 steps
1. Sinusoidal Demand, Section 4.2	0.029 vs 0.061	0.041 vs 0.079	0.054 vs 0.098
2. Simplified Model, Section 4.5	0.032 vs 0.068	0.044 vs 0.091	0.056 vs 0.114
3.Auto Regressive Workload, Section 4.6	0.013 vs 0.029	0.016 vs 0.038	0.022 vs 0.047

TABLE 3. Errors in response time prediction, when Qfac is varied

	Errors in tracked and predicted User response time						
		No trend tracking (A = I)			***With trend tracking (A varying)***		
QFac	***MARE***	***fMARE (1 step)***	***fMARE (2 steps)***	***fMARE (3 steps)***	***fMARE (1step)***	***fMARE (2 steps)***	***fMARE (3 steps)***
0.001	0.0185	0.0323	0.0505	0.0707	0.0254	0.0317	0.0401
0.005	0.0118	0.0301	0.0475	0.0673	0.0252	0.0330	0.0402
0.02	0.0077	0.0304	0.0478	0.0675	0.0290	0.0414	0.0542
0.1	0.0064	0.0307	0.0479	0.0678	0.0313	0.0465	0.0626
0.5	0.0063	0.0311	0.0491	0.0696	0.0314	0.0470	0.0629

TABLE 4. Errors in response time prediction, with 2nd order AR trend applied to various cases, and Qfac = 0.02

	Errors in tracked and predicted User response time						
		fMARE 1 step ahead			fMARE 3 steps ahead		
Case and Section	MARE	no trend tracking	trend tracking	benefit %	no trend tracking	trend tracking	benefit %
1. Full model of Sec. 4.2	0.0069	0.0302	0.0262	13.27	0.0674	0.0438	35.01
2. Fast change workload, Sec 4.3	0.0055	0.0546	0.0251	54.04	0.1594	0.0824	48.29
3. Fast change workload, Sec 4.3, (first order AR trend, to compare)	0.0059	0.0553	0.0242	56.32	0.1605	0.0939	41.46
4. Autoregressive random workload trace of Sec 4.6	0.0067	0.0152	0.0125	17.66	0.0325	0.0203	37.44
5. AR random workload Sec 4.6, (first order AR trend, to compare)	0.0070	0.0151	0.0131	13.30	0.0325	0.0216	33.58

The choice of the drift variance parameter *Qfac* for the filter needs to be examined. With a small value, the tracked and predicted values are smoother, but if it is too small the filter becomes sluggish. When the value of *Qfac* was varied over a wide range, from 0.001 to 0.1, the results for the performance prediction were as shown in Table 3. The results do vary, and there appears to be a shallow optimum around *Qfac* = 0.005. Thus for this system our default value of 0.02 overestimates the change in parameter values over a single step. Reflection confirms this is the case, since the largest change in a parameter (at the point where the sinusoid is zero) is less than 1% of the parameter value. Still, a larger value such as 0.02 is a sound defensive strategy against more rapid changes.

4.3 Rapid Sinusoidal Variation

It is the separation of the intermediate time scale (a few steps) and the long time scale in which the sinusoidal variation is imposed (a period of 100 steps in Section 4.1) that makes it possible to track the trends. The rate of variation of the parameters in the simulation was speeded up by a factor of 2.5, giving a period of 40 steps.

The errors are larger for the faster changes, as shown by line 3 in Table 4. This rate of change begins to violate the separation of the "intermediate" time scale (at which the trend can be approximated by the AR model) of the actual variation of **x**.

4.4 A Second-order Time-Series Model

The first trend model used a first-order AR model for each of S_w, S_d, and Z. Here we use the same data as in Section 4.2 but employ a second-order AR model, giving:

$$\mathbf{x} = [S_d, S_w, Z]^T$$

$$x_i(k+1) = g_{(i-1)*2+1}\, x_i(k) + g_{(i-1)*2+2}\, x_i(k-1) + \xi_i(k), \quad i = 1,..,3;$$

When the six autoregressive coefficients are gathered into the vector **g**, we obtain

$$\boldsymbol{\alpha}(k)^T = (\mathbf{x}(k)^T, \mathbf{x}^T(k-1), \mathbf{g}^T)$$

and in the Kalman Filter, the state projection equation $\boldsymbol{\alpha}(k+1) = \mathbf{F}(k)\boldsymbol{\alpha}(k)$ has the 12x12 matrix **F**:

$$\mathbf{F}(k) = \begin{bmatrix} \mathbf{G^*_{3x6}} & \mathbf{0_{3x6}} \\ \mathbf{I}_{3x3} \quad \mathbf{0}_{3x3} & \mathbf{0}_{3x6} \\ \mathbf{0}_{6x6} & \mathbf{I}_{6x6} \end{bmatrix}$$

where

$$\mathbf{G^*_{3x6}} = \begin{bmatrix} g_1(k) & 0 & 0 & g_2(k) & 0 & 0 \\ 0 & g_3(k) & 0 & 0 & g_4(k) & 0 \\ 0 & 0 & g_5(k) & 0 & 0 & g_6(k) \end{bmatrix}$$

and **I** and **0** are the identity and null matrix.

The sensitivity matrix **A** is:

$$\mathbf{A}(k) = \begin{bmatrix} \mathbf{G^*_{3x6}} & \mathbf{X^*_{3x6}} \\ \mathbf{I}_{3x3} \quad \mathbf{0}_{3x3} & \mathbf{0}_{3x6} \\ \mathbf{0}_{6x6} & \mathbf{I}_{6x6} \end{bmatrix}$$

where

$$\mathbf{X^*}(k) = \begin{bmatrix} x_1(k) & x_1(k\text{-}1) & 0 & 0 & 0 & 0 \\ 0 & 0 & x_2(k) & x_2(k\text{-}1) & 0 & 0 \\ 0 & 0 & 0 & 0 & x_3(k) & x_3(k\text{-}1) \end{bmatrix}$$

Table 4 shows the errors in response time prediction when the 2nd order model was applied to all the cases in the Section. Comparing row 1 to Table 1 where the first order model was used, the second order model performs always better. This is also true for the other cases in the paper. Because some the changes are sinusoidal and a sinusoid is exactly second order in dynamics, there is no surprise the second order model provides better performance. For 3-step prediction, the improvement of using 2nd order model is 30% - 50%.

4.5 A Simplified Performance Model

The question of how complex a performance model should be is always important. In general we would prefer to work with the simplest model that fits the data.

We can address this question by considering a simplified model for the Web Server. The simulation data is the same as in Section 4.2, but the performance model is simplified as shown in Figure 6, with just the WebServer. That is, the delays due to the database service and disk are folded into the service time of the web server in this model.

In the simplified model, the tracking parameters are **x** = [*Z*, *S*]. The measurement set **z** consists of the response time and throughput of the web server only. The experiment in this part is to investigate if a simplified model is good enough to predict the future response time, when the same simulated data trace is presented to the filter.

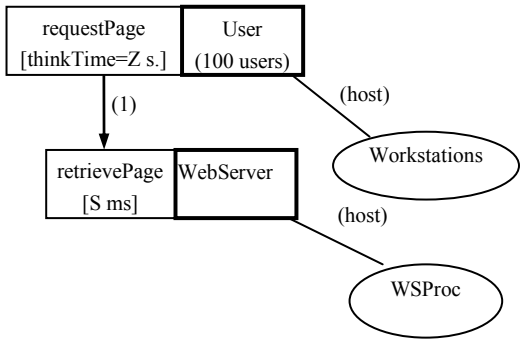

Figure 6. The simplified LQN model of a web server system

Comparing Table 5 to Table 3 shows that the performance prediction error (*fMARE*) is a little larger with the simplified model, for all values of *Qfac*. Thus, the additional detail in Section 4.2 is probably justified.

The question of whether the additional detail is significantly better should be decided by a statistical test on the residuals for the two models. The calculation requires the sums of squares of the residuals for the performance prediction (the residuals are the amounts in the absolute value expression used in defining *MARE* or *fMARE*). If *SSE1* is the (larger) sum of squares for the smaller model in this section (with n_1 estimated parameters), and *SSE2* is

for the more elaborate model reported in Section 4.2 (with n_2 estimated parameters), and the errors are assumed normally distributed and independent, then an approximate F statistic for the fit is

F = ((*SSE1*-*SSE2*)/(n_2 - n_1)) / (*SSE2*/no. of data points - n_2 -1)

with degrees of freedom (n_2 - n_1, data points - n_2 -1). The results for tracking (zero steps ahead) with *Qfac* = 0.02 and 990 data points in the trace gave:

SSE1 = 8.472 with n_1 = 2, *SSE2* = 7.732 with n_2 - n_1 = 3.

F = (*SSE1*-*SSE2*/1)/(*SSE2*/988) = 94.5

while for the three-step predictions, they gave:

SSE1 = 5.956, *SSE2* = 4.559

F = 302.7

The critical value at the 5% level and (1, 988) degrees of freedom is 3.84. If F>3.84 (as in both of these cases) we can reject the hypothesis that the simpler model is as good as the more elaborate one, with 95% confidence

TABLE 5. The quality of prediction in the simplified model

	Errors in tracked and predicted User response time						
		No trend tracking (A = I)			*With trend tracking (A varying)*		
QFac	***MARE***	***fMARE (1step)***	***fMARE (2 steps)***	***fMARE (3 steps)***	***fMARE (1step)***	***fMARE (2 steps)***	***fMARE (3 steps)***
0.001	0.0254	0.0353	0.0524	0.0718	0.0315	0.0353	0.0403
0.005	0.0293	0.0367	0.0517	0.0695	0.0310	0.0367	0.0439
0.02	0.0354	0.0421	0.0552	0.0720	0.0320	0.0420	0.0553
0.1	0.0387	0.0449	0.0576	0.0738	0.0323	0.0449	0.0622
0.5	0.0394	0.0455	0.0581	0.0743	0.0323	0.0455	0.0639

Since the assumptions required for the F-test cannot be substantiated for this kind of estimation, its application is an approximation and the conclusion is only tentative. However, these results indicate that the simpler model is very probably less accurate, and the conclusion is stronger for the predictions, than for the tracking. That indicates that the trend modeling is effective.

4.6 Random Workload

This case replaces the sinusoidal variation of the arrivals by a randomly varying rate of arrivals. A somewhat smooth random arrival sequence was generated by the third-order AR model

$Arr(k+1)=2.1Arr(k) -1.43\ Arr(k-1) +0.315\ Arr(k-2) +0.015\ \xi(k)$

(with ξ being a normal independent process of unit variance). This rate in arrivals/ms was scaled by a factor of 0.01 and then biased by a constant 0.01 so as to always be positive. A closed workload with 1000 users was assumed.

The think time $Z(k)$ was derived from the sequence using:

- N = 1000 = assumed size of the pool of users
- $Z(k) = N/Arr(k)$ = think time that would give $Arr(k)$ if the response time is much less than $Z(k)$

Since the mean think time $Z(k)$ is around 100000 ms, which is much bigger than the response time (mean value < 300ms), the simulated arrival rate is close the $Arr(k)$.

This series was used as input to the Web Server simulation, with sinusoidal variation of the demand parameters S_w and S_d as in Section 4.2, and the estimation was carried out as before. With a first-order AR trend model, the results in Figure 6 (and in Table 4, line 5) were obtained.

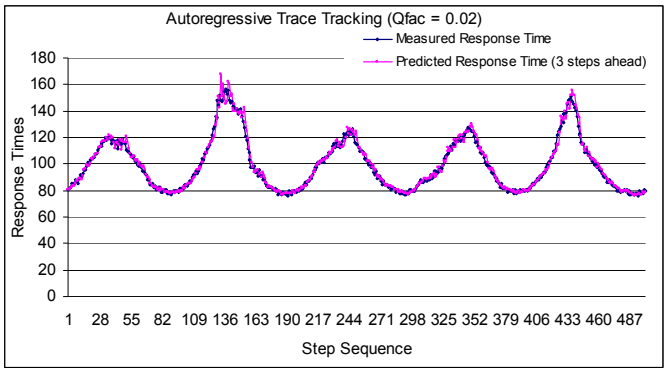

Figure 7. Results for tracking and predicting with autoregressive trace

Generally the tracking results were excellent, similar to the results in Section 4.2. The second-order AR trend gives even better results (Table 4, line 4).

4.7 Separate Regression for Prediction

Over the ensemble of cases described here, the question of separate trend prediction by regression on the history of estimates can be revisited. Table 2 includes results for the cases considered later in the section. The integrated predictor is able to predict the trend of the service demands in addition to the arrival rates, thus it is not surprising that it performs better than a separate predictor.

4.8 Validation on a Real System

Trade 6 benchmark application [29] is a web based end-to-end benchmark and performance sample application developed by IBM. It implements the basic functions for stock trading. By accessing a servlet in the application, different types of user requests will be generated randomly with specific probabilities according to a benchmark. We deployed the Trade 6 application on a platform consisting of IBM WebSphere® Application Server version 6.1 and IBM DB2® Universal Database (DB2 UDB) Version 8.2. WebSphere and DB2 run on two virtual machines using CentOS 5 as the OS. The structure of the performance model of the system is similar to the model shown in Figure 1, while the network delay and disk operation are not considered in this case.

We used a workload generator that generates a traffic that is common to most e-commerce website. In the experiment, the number of users increases gradually to a peak, then decreases

gradually to a minimum number in a repetitive pattern that reflects the daily usage.

The run time performance metric such as the response times, invoke rates, CPU utilizations etc are collected from WebSphere application server using Java Management Extensions (JMX) technology. The sampling interval is 60 minutes.

The fMARE of the tracking with 1 step ahead trend prediction is 0.0245, while the fMARE of the tracking without trend is 0.0306. The accuracy improves by 20%. For 3 steps ahead prediction, the accuracy improves by 37.5% (0.0474 vs 0.0759). The findings are consistent with those found through simulation.

5. CONCLUSIONS

This work represents the first attempt to combine on-line tracking of performance parameters with estimate of their current short-term trends, in the same estimator. This makes it possible to estimate trends for hidden parameters, that is parameters that are fitted but not observed directly.

The results show that an autoregressive model can successfully be fitted to the time variation of hidden parameters using an Extended Kalman Filter estimator. The autoregressive model is used to predict the parameter values forward in time, provided the local trends change slowly enough.

Integrated tracking of trend coefficients provides much better predictions of performance measures such as response time, than separate regression on the history of the tracking estimates or the observations themselves. However its unique capability is, to predict hidden parameter values.

Experiments with smoothly varying parameters and measurement errors compared different filter settings, and investigated the effect of variation speed and model complexity. Prediction errors of a few percent in performance measures, and somewhat smaller errors in parameter values, were obtained. The predictions were not very sensitive to the filter disturbance variance setting (here represented by *Qfac*).

Experiments with large-amplitude random swings in the arrival rate showed an equal capability to make predictions. With long-range dependent variations in the arrival rate, the filter could not predict arrivals accurately in all cases but recovered quickly from momentary errors, indicating a useful robustness.

As described in [28] there is condition for convergence of the Kalman Filter, on an identifiability matrix:

$$\text{rank}\,[\mathbf{H}^{\mathrm{T}}, (\mathbf{HA})^{\mathrm{T}}, \ldots, (\mathbf{HA}^{i})^{\mathrm{T}}, \ldots, (\mathbf{HA}^{n-1})^{\mathrm{T}}] = n.$$

In estimators without trend tracking (as considered in [22][24][28]) this condition requires that there be as many measurements as there are parameters to be estimated. With trends there are more quantities to estimate, but an equal number of additional measurements are not required because of the relationships of estimates across time, through the trends. The condition above should be tested, and may still fail if the model is not sensitive to some parameter, or is sensitive only to a linear combination such as the sum of two parameters. This sometimes limits the amount of system detail that can be modeled and calibrated.

There is a significant limitation to tracking parameters with a trend predictor integrated in the estimator: by assuming gradual trends in parameter values, it makes the filter less agile. There is a necessary separation of time-scales between the parameter variation and the sampling steps for the estimator. In our experiments stable tracking required that a trend should continue for one the order of steps. By contrast, in [24] an estimator without trend tracking was shown capable of tracking large jumps in value separated by just a few steps. The difference is that the filter in [24] assumed a random change in parameter values at every step, and the only smoothness assumption was in the size of the assumed random change.

This agility limitation points up a potential conflict in the goals of tracking: a goal in a smoothly varying regime, to home in on excellent tracking with a capability to make predictions by using trend tracking, versus a different goal in a chaotic regime, to try to keep up with rapid changes. The smoothly varying regime is an important one for routine business operations, and is able to accommodate and predict large changes in behaviour provided they are not too rapid. In those situations the agility limitation is not a serious disadvantage.

6. ACKNOWLEDGMENT

The authors wish to acknowledge useful conversations with Johnny Wong and Cristina Amza, and members of the IBM Center for Advanced Studies, Toronto.

7. REFERENCES

[1] M. Arlitt and C. Williamson. “Web Server Workload Characterization: The Search for Invariants”, *Proc. ACM SIGMETRICS ’96*, May 1996.

[2] G.E.P. Box, G. Jenkins G. Reinsel, *Time Series Analysis: Forecasting and Control,* Prentice Hall, third edition, 1994.

[3] E. Brookner, *Tracking and Kalman Filtering Made Easy*, Wiley Interscience, 1998.

[4] E. DiNitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A journey to highly dynamic self-adaptive service-based applications," *Automated Software Engineering*, vol. 15, no. 3-4 pp. 313-341, 2008.

[5] G. Franks, T. Al-Omari, M. Woodside, O. Das, and S. Derisavi, "Enhanced Modeling and Solution of Layered Queueing Networks," *IEEE Trans. on Software Eng.*, vol. 35, no. 2 pp. 148-161, 2009.

[6] S. Ghanbari, G. Soundararajan, J. Chen, C. Amza, Adaptive Learning of Metric Correlations for Temperature-Aware Database, *Proc. Fourth Int. Conf. on Autonomic Control*, p 26, 2007

[7] Gmach D., Rolia J., Cherkasova L., Kemper A., “Workload Analysis and Demand Prediction of Enterprise Data Center Applications,” *Proc. 10th IEEE Int. Symp. on Workload Characterization,* pp. 171-180, Sept. 2007.

[8] J.G. De Gooijer, R.J. Hyndman, “25 years of time series forecasting”, Int. J. of Forecasting v 22 pp 443– 473, 2006.

[9] R. Jain, *The Art of Computer Systems Performance Analysis*. Wiley, 1991.

[10] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. New York: Academic Press, 1970

[11] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 6, no. 1 pp. 41-50, 2003

[12] W.E. Leland, M.S. Taqqu, W. Willinger, D.V. Wilson, “On the Self-similar Nature of Ethernet Traffic (Extended Version)”, *IEEE/ACM Trans. on Networking*, v. 2 n. 1, 1994

[13] T.-H. Li, "A Hierarchical Framework for Modeling and Forecasting Web Server Workload," *Journal of the American Statistical Association*, vol. 100, no. September 2005 pp. 748-763, 2005.

[14] M. Litoiu, M. Woodside, T. Zheng, “Hierarchical model based autonomic control of software systems”, *Proc. of Design and Evolution of Autonomic Software (DEAS’05) Workshop*, St. Louis, May 2005.

[15] Y. Lu, T. Abdelzaher, C. Lu, L. Sha, and X. Liu, "Feedback Control with Queueing-Theoretic Prediction for Relative Delay Guarantees in Web Servers," *Proc 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS03)*, May 2003.

[16] D.A. Menascé, M.N. Bennani, "On the use of performance models to design self-managing computer systems," *Proc. 2003 Computer Measurement Group Conf.*, Dallas, 2003.

[17] J. A. Rolia and K. A. Sevcik. The method of layers. *IEEE Trans. Software Engineering*, 21(8):689–700, Aug. 1995.

[18] Shen D., Hellerstein J.L., “Predictive Models for Proactive Network Management: Application to a Production Web Server,” *Proceedings of IEEE NOMS 2000,* pp 833-846, 2000.

[19] R. Shibata, "Asymptotically Efficient Selection of the Order of the Model for Estimating Parameters of a Linear Process," *Annals of Statistics*, vol. 8, no. 1 pp. 147-164, 1980.

[20] VMware Capacity Planner. http://www.vmware.com

[21] C.M. Woodside, J. E. Neilson, D. C. Petriu, and S. Majumdar, “The stochastic rendezvous network model for performance of synchronous client-server-like distributed software,” *IEEE Trans. Computers*, vol. 44, no. 8, pp. 20–34, Aug. 1995

[22] M. Woodside, T. Zheng, M. Litoiu , “The use of optimal filters to track parameters of performance models”, *Proc. 2nd Int. Conf. on Quantitative Evaluation of Systems (QEST05)*, Torino, Sept. 2005.

[23] M. Woodside, D. C. Petriu, D. B. Petriu, H. Shen, T. Israr, J. Merseguer, “Performance by Unified Model Analysis (PUMA)”, Proc. 5th Int. Workshop on Software and Performance (WOSP 2005), pp 1-12, July 2005

[24] M. Woodside, T. Zheng, M. Litoiu, “Service System Resource Management Based on a Tracked Layered Performance Model”, *Proc 3rd IEEE International Conference on Autonomic Computing*, 2006.

[25] Xen, http://xen.xensource.com/, a virtual machine monitor.

[26] C. You and K. Chandra, "Time Series Models for Internet Data Traffic," *Proc. Conf on Local Computer Networks 1999* (LCN '99), Lowell, MA, USA, Oct 1999, pp. 164-171.

[27] Y. M. Zhan and A. K. S. Jardine, "Adaptive autoregressive modeling of non-stationary vibration signals under distinct gear states. Part 1: modeling," *Journal of Sound and Vibration*, v. 286, pp. 429–450, 2005.

[28] Zheng T., Woodside C. M., Litoiu M., “Performance Model Estimation and Tracking Using Optimal Filters”, *IEEE Transactions on Software Engineering*, Vol. 34, No 3, pp. 391-406, June

[29] http://www.ibm.com/developerworks/data/tutorials/dm0506lau/section2.html, IBM Trade 6 Performance Benchmark