PAPER

Joint Tracking of Performance Model Parameters and System Behavior Using a Multiple-Model Kalman Filter

Zhen ZHANG[†a)], ***Student Member***, **Shanping LI**[†], ***and*** **Junzan ZHOU**[†], ***Nonmembers***

SUMMARY Online resource management of a software system can take advantage of a performance model to predict the effect of proposed changes. However, the prediction accuracy may degrade if the performance model does not adapt to the changes in the system. This work considers the problem of using Kalman filters to track changes in both performance model parameters and system behavior. We propose a method based on the multiple-model Kalman filter. The method runs a set of Kalman filters, each of which models different system behavior, and adaptively fuses the output of those filters for overall estimates. We conducted case studies to demonstrate how to use the method to track changes in various system behaviors: performance modeling, process modeling, and measurement noise. The experiments show that the method can detect changes in system behavior promptly and significantly improve the tracking and prediction accuracy over the single-model Kalman filter. The influence of model design parameters and mode-model mismatch is evaluated. The results support the usefulness of the multiple-model Kalman filter for tracking performance model parameters in systems with time-varying behavior.

key words: *performance modeling, tracking filter, multiple-model method, queueing theory, resource management*

1. Introduction

As today's Internet-based web applications become more complex, resource management tasks such as resource allocation become increasingly difficult. One of the most complex aspects of these tasks is to evaluate the performance effects of hypothetical changes in system configurations. Performance models such as queueing models provide a powerful means of evaluating these performance effects [1]. However these models quickly become obsolete if they fail to keep up with the changes in system parameters and behavior. These changes can be common, rapid and unexpected, especially in large scale, Internet-based systems. For example, the (average) service time of a request, which is an important parameter of performance models, often changes with software updates, browsing pattern shifts, and virtual machine migration. For this reason, performance models must be updated in a timely manner, preferably automatically and online.

Estimating performance model parameters in run-time is not an easy task. Performance model parameters, such as the CPU service time of a request, are very hard to measure directly. Performance data, such as CPU utilization, comes in streams of noisy data, which must be processed incrementally. One solution is to adapt offline methods of estimating performance parameters, such as regression-based methods, to online environments, using adhoc smoothing techniques to reduce noise. This approach, however, has been shown to be viable only for *rough* estimates [2].

The Kalman filter can be used as an alternative to adapting offline methods to online systems. It is an estimator specific to dynamic systems, and it has been shown to be suitable for the tracking of performance model parameters that vary over time [3]–[6]. The Kalman filter treats performance model parameters such as the service time of a request as *hidden states* and the various types of performance data, such as CPU utilization, as noisy *measurements*. It provides accurate estimates of hidden states by combining past estimations with current noisy measurements in an optimal way [7]. The Kalman filter requires three essential inputs to encode the modeler's knowledge of system behavior: a *measurement model* describes the effect of state changes on the measurement; a *process model* describes expected trends or patterns in the state changes; and *error covariances* specify the belief of how accurate is the above measurement and process model. For example, in one previous study a layered queueing network model was used as the measurement model of a two-tier web application, an autoregressive model was used as the process model to approximate the short-term trend of sinusoid changes in think time, and a small measurement and process noise covariance was used to indicate that the measurement noise and state disturbance were low [6]. Currently, all tracking filter solutions for the estimation of performance parameters assume that these inputs are time-invariant and known to the modeler.

In practice, a computer system may undergo both continuous changes in states and discrete changes in modes. A *mode* is the true system behavior that is to be approximated by the *models*, including measurement model, process model, and error covariances. For example, the service discipline of CPU can be approximated by process sharing (PS) discipline when the request service time is significantly larger than the time-slice of the OS scheduler, but it more closely resembles a first-come first-served (FCFS) discipline, when the service time is smaller than the time-slice [8]. The service time of the system may vary smoothly during normal time, but it undergoes rapid changes during bursty time. The measurement noise is often larger during idle time than during busy time [4]. To

Manuscript received July 2, 2012.
Manuscript revised December 16, 2012.

[†]The authors are with Computer College of Zhejiang University, Hangzhou, Zhejiang, 310027, China.
a) E-mail: zhen@zju.edu.cn

DOI: 10.1587/transinf.E96.D.1309

track and predict states accurately in such system, a conventional *single-model-based filter* must be tuned to compromise among modes. In contrast, *a multiple-model-based filter*, which is a kind of *multiple-model (MM)* method that runs many different filters and combines their results, can give good results in each mode. Such MM methods are also useful for situations in which the mode is unknown but is known to remain constant.

This paper makes two contributions to the knowledge of the field: First, we investigate how to apply MM methods to the tracking of performance model (Sect. 4). We use a particular MM method, called the *interacting multiple-model (IMM)*. The necessary background for IMM is presented in Sect. 3. Practical issues related to IMM are analyzed. These includes model set design, models with different state dimensions, and performance model linearization. As far as we know, this is the first successful attempt to track system behavior modes *along with* parameters of performance models. Second, we illustrate the usage of IMM using case studies in both simulated and real systems (Sect. 5). These case studies show how IMM can be used to model changes in different system behaviors, including performance modeling, measurement noise, and process modeling. The experiments show that in the presence of system behavior change, IMM has better tracking and prediction results than the single-model method, in a wide range of design parameters, even if mode-model mismatch exists.

2. Related Work

Woodside et al. pioneered the use of tracking filters to estimate performance model parameters. Specifically, they used an extended Kalman filter to estimate the parameters of a closed queuing network [3]. They expanded upon their work in various ways: 1) They used the filter to track parameters in a layered queueing network model and devised a method of determining measurement noise covariance [4], [9]. 2) They used tracked performance models to implement feed-forward controllers [10]. 3) They integrated autoregressive trending in the filter to improve the quality of the predictions [6]. Kumar et al. investigated the convergence problem of the filter in tracking parameters of multi-class queueing model [5]. They addressed the problem by augmenting the measurement vector with a set of constraints based on past measurements. All these studies made the assumption that the filter inputs are fixed and known. However, previous studies have shown that the results of filter prediction can suffer if incorrect performance models are used [4], [6], [9]. And ad hoc compensation techniques [11] have been used to adapt measurement noise covariance but without success [9]. To the best of our knowledge, our study presents the first viable solution to the problem of filter input uncertainty.

There is a vast amount of literature covering the adaptation of tracking filters in domains other than performance modeling. MM methods have been shown to be superior to many popular methods [12], and are generally considered the mainstream approach, especially in the field of target tracking [13]. We refer readers to Li and Jilkov [13] for a comprehensive survey of MM methods. Among many MM methods, IMM, first proposed by Blom and Bar-Shalom [14], is considered to strike the best balance between tracking accuracy and computational complexity [15]. Many practical issues arise when applying IMM to the domain of performance model tracking, in particular the design of model set. We present our solutions to these issues and evaluate the tracking results under various setting. Although we focus on IMM, these solutions and results may be applied to other MM methods as well.

MM methods are essentially semi-parametric estimators [16], similar ideas have been successfully applied to related areas such as control [19] and time-series forecast [17]. In particular, multiple time-series models are often combined to improve the accuracy of performance prediction [18]. In contrast to our approach, the prediction of such methods is based directly on the observable performance metrics, thus cannot predict hidden parameters of performance models, with which better prediction can be made [6]. Moreover, MM methods provide a general method that can combine not only multiple process models (time-series models), but also multiple measurement models and error covariances.

3. Background

We first present a brief introduction of the extended Kalman filter and interacting multiple-model method. The extended Kalman filter forms the basis of multiple-model methods. It is assumed that the reader is aware of the general theory of Kalman filtering. The reader is referred to Simon [7] for more details.

3.1 Extended Kalman Filter (EKF)

The Kalman filter is used to recursively estimate discrete time system state space models. In discrete time cases, time is advanced in fixed intervals and indexed using a step counter k. The system states remain stationary within a step but may undergo changes across steps. The system state space model can be described as follows:

$$\boldsymbol{x}_k = \mathrm{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{w}_k, \quad \boldsymbol{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k) \tag{1}$$

$$\boldsymbol{z}_k = \mathrm{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k, \quad \boldsymbol{v}_k \sim \mathcal{N}(0, \mathbf{R}_k) \tag{2}$$

where $\boldsymbol{x}$ is the *hidden state* vector, $\boldsymbol{z}$ is the *measurement* vector, f is the *process model* of the evolution of the state vector, and h is the *measurement model* of the impact of $\boldsymbol{x}$ on $\boldsymbol{z}$. $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the process and measurement noises, respectively. These are assumed to be zero-mean Gaussian white noises with covariances $\mathbf{Q}$ and $\mathbf{R}$. For tracking the performance model, $\boldsymbol{x}$ corresponds to performance model parameters such as request service time, $\boldsymbol{z}$ corresponds to the performance metrics such as CPU utilization and response time, and h corresponds to the performance model.

Table 1 EKF notations.

Notation	Description
$\hat{\boldsymbol{x}}_k$	(a posteriori) state estimate at time k
$\boldsymbol{P}_k$	(a posteriori) state covariance estimate at time k
$\hat{\boldsymbol{x}}_k^-$	a prior state estimate at time k
$\boldsymbol{P}_k^-$	a prior state covariance estimate at time k
$\boldsymbol{z}$	measurement data at time k
f	process model
h	measurement model
$\mathbf{Q}$	process noise covariance
$\mathbf{R}$	measurement noise covariance
$\boldsymbol{F}_k$	sensitivity matrix of f at time k
$\boldsymbol{H}_k$	sensitivity matrix of h at time k
$\boldsymbol{v}_k$	measurement residual at time k
$\boldsymbol{S}_k$	measurement residual covariance at time k
$\boldsymbol{K}_k$	Kalman gain at time k

The Kalman filter is a recursive state estimator. This means that only the state estimate of the previous time step and current measurement data are required to estimate the current state. EKF is a variant of the Kalman filter that deals with measurement and process model nonlinearity [7]. This is the case here because h is derived from the performance model, a nonlinear function. A summary of the EKF notations is given in Table 1.

EKF has two steps. The *prediction step* predicts next state of the system $\hat{\boldsymbol{x}}_k^-$ using the previous state estimate $\hat{\boldsymbol{x}}_{k-1}$ and the *update step* combines the predicted state of the system $\hat{\boldsymbol{x}}_k^-$ with the new measurement data $\boldsymbol{z}_k$ in a weighted average manner, to produce the a posteriori state estimate $\hat{\boldsymbol{x}}_k$. The weight factor, called the Kalman gain ($\boldsymbol{K}_k$) is updated at each iteration to minimize the mean square errors of state estimation. These steps can be formulated as follows:

- Prediction

$$\hat{\boldsymbol{x}}_k^- = \mathrm{f}(\hat{\boldsymbol{x}}_{k-1}) \tag{3}$$

$$\boldsymbol{P}_k^- = \boldsymbol{F}_{k-1}\boldsymbol{P}_{k-1}\boldsymbol{F}_{k-1}^T + \mathbf{Q}_{k-1} \tag{4}$$

- Update

$$\boldsymbol{S}_k = \boldsymbol{H}_k\boldsymbol{P}_k^-\boldsymbol{H}_k^T + \mathbf{R}_k \tag{5}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}_k^T/\boldsymbol{S}_k \tag{6}$$

$$\boldsymbol{v}_k = \boldsymbol{z}_k - \mathrm{h}(\hat{\boldsymbol{x}}_k^-) \tag{7}$$

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\boldsymbol{v}_k \tag{8}$$

$$\boldsymbol{P}_k = (\mathbf{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k^- \tag{9}$$

where

$$\boldsymbol{F}_{k-1} = \left.\frac{\partial \mathrm{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1}}, \quad \boldsymbol{H}_k = \left.\frac{\partial \mathrm{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k^-} \tag{10}$$

$\boldsymbol{F}_{k-1}$ and $\boldsymbol{H}_k$ are the sensitivity matrices of f and h around state estimates $\hat{\boldsymbol{x}}_{k-1}$ and $\hat{\boldsymbol{x}}_k^-$. A more detailed description of the two steps can be found in Simon [7].

3.2 Interacting Multiple-Model (IMM)

To model the system with possible mode change, IMM prepares a set of M models $\{m^1, \ldots, m^M\}$ as possible candidates of the true model, and a bank of *elemental filters* (we use EKFs in this paper), each based on a unique model in the set. The system model at step k is one of M models. The model switching is described by a Markov chain with known prior probability $P\{m_0^j\} = \pi_0^j$ and transition probabilities $P\{m_k^i|m_{k-1}^j\} = \pi_{ji}$. A summary of the notations specific to IMM is given in Table 2. Not included in this table are the notations for the variables of each model, such as $\hat{\boldsymbol{x}}_k^j$, the $\hat{\boldsymbol{x}}_k$ of model-j-based filter. They are similar to those given in Table 1 except that they are given a superscript j to denote the index of the model.

Table 2 Notations specific to IMM.

Notation	Description
M	size of the model set
m^j	jth model in the model set
m_k^j	jth model is in effect at time k
π_0^j	initial probability of m^j
π_{ji}	model transition probability from m^j to m^i
$\mu_k^{-,j\|i}$	probability of m_{k-1}^j given that m_k^i conditioned on $\boldsymbol{z}_{k-1}$
μ_k^j	(posterior) probability of m^i at time k
$\mu_k^{-,j}$	prior probability of m^i at time k
$\bar{\boldsymbol{x}}_{k-1}^i$	reinitialized state of m^i at time $k-1$
$\bar{\boldsymbol{P}}_{k-1}^i$	reinitialized state covariance of m^i at time $k-1$
$\hat{\boldsymbol{x}}_k$	combined state estimate at time k
$\boldsymbol{P}_k$	combined state covariance estimate at time k

IMM consists of three major steps: interaction (mixing), filtering, and combination. During each time step, IMM reinitializes each filters using a different combination of previous model-conditioned estimates (interaction). Then IMM performs standard EKF filtering for each model as if this particular model is the right model at current time step (filtering). Finally, IMM computes a weighted combination of updated state estimates produced by all filters, yielding a final estimate for the state and error covariance in that time step (combination). The weights are chosen according to the probabilities of the models, which are updated during the filtering step of the algorithm. The equations for each step are as follows:

- Step 1: Interaction (mixing)

The mixing weight is computed as follows:

$$\mu_k^{-,j|i} \triangleq P\{m_{k-1}^j|m_k^i, \boldsymbol{z}_{k-1}\} = \pi_{ji}\mu_{k-1}^j/\bar{u}_k^{-,i} \tag{11}$$

where $\triangleq$ means "equals by definition." μ_{k-1}^j is the probability of m^j at time $k-1$, $\bar{u}_k^{-,i} = \sum_j \pi_{ji}\mu_{k-1}^j$ is the predicted probability of m^i, which is the normalization constant for $\mu_k^{-,j|i}$.

Then each filter is reinitialized with mixed inputs, which are the state and covariance estimates:

$$\bar{\boldsymbol{x}}_{k-1}^i \triangleq E(x_{k-1}|m_k^i, \boldsymbol{z}_{k-1}) = \sum_j \hat{\boldsymbol{x}}_{k-1}^j\mu_k^{-,j|i} \tag{12}$$

$$\bar{\boldsymbol{P}}_{k-1}^i = \sum_j \left\{\boldsymbol{P}_{k-1}^j + [\bar{\boldsymbol{x}}_{k-1}^i - \hat{\boldsymbol{x}}_{k-1}^j][\bar{\boldsymbol{x}}_{k-1}^i - \hat{\boldsymbol{x}}_{k-1}^j]^T\right\}\mu_k^{-,j|i} \tag{13}$$

- Step 2: Filtering

For each model m^i

$$[\hat{\boldsymbol{x}}_k^{-,i}, \boldsymbol{P}_k^{-,i}] = \mathrm{EKF}_p(\bar{\boldsymbol{x}}_{k-1}^i, \bar{\boldsymbol{P}}_{k-1}^i, \mathrm{f}^i, \mathbf{Q}_{k-1}^i) \tag{14}$$

$$[\hat{\boldsymbol{x}}_k^i, \boldsymbol{P}_k^i] = \mathrm{EKF}_u(\hat{\boldsymbol{x}}_k^{-,i}, \boldsymbol{P}_k^{-,i}, \mathrm{h}^i, \mathbf{R}_k^i, z_k) \tag{15}$$

Here, we denote the prediction and update steps (Eq. (3)–(4) and Eq. (5)–(9)) of the extended Kalman filter with $\mathrm{EKF}_p(\cdot)$ and $\mathrm{EKF}_u(\cdot)$. We also compute the likelihood of the measurement for each filter as follows:

$$L_k^i = \mathcal{N}(\boldsymbol{v}_k^i; 0, \boldsymbol{S}_k^i) \tag{16}$$

Here $\boldsymbol{v}_k^i$ is the measurement residual and $\boldsymbol{S}_k^i$ is the measurement residual covariance for model m^i in the Eq. (5). $\mathcal{N}$ is the pdf of multivariate Gaussian distribution.

Then the probability of each model m^i at time k is updated as follows:

$$\mu_k^i = \bar{u}_k^{-,i} L_k^i / c \tag{17}$$

Here $c = \sum_i \bar{u}_k^{-,i} L_k^i$ is a normalization constant.

- Step 3: Combination

The combined estimates for the state and covariance are computed as follows:

$$\hat{\boldsymbol{x}}_k = \sum_i \hat{\boldsymbol{x}}_k^i \mu_k^i \tag{18}$$

$$\boldsymbol{P}_k = \sum_i \left\{ \boldsymbol{P}_k^i + [\hat{\boldsymbol{x}}_k - \hat{\boldsymbol{x}}_k^i][\hat{\boldsymbol{x}}_k - \hat{\boldsymbol{x}}_k^i]^T \right\} \mu_k^i \tag{19}$$

IMM is a Bayesian suboptimal estimator. The key aspect of the IMM algorithm is the interaction step. This mixing of the estimates allows the individual poor estimates caused by model mismatching to be replaced with estimates from better models. The whole filters bank then benefits from these better estimates. This is especially important when a large change in states is accompanied by a change in system mode.

Figure 1 shows the structure of IMM estimation algorithm. Note that the elemental filters run in parallel, and the only difference from the single-model Kalman filter is that each filter is reinitialized by the interaction component before the normal prediction-update recursion. Also note that the recursion loops between interaction component and the elemental filters are internal to the IMM and that the final tracked state is given by the output component.

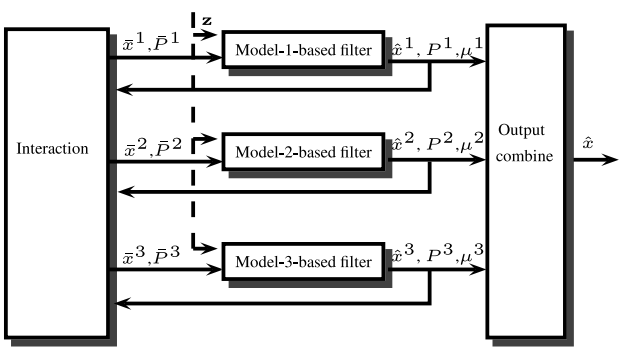


Fig. 1 Logic architecture of IMM. (three models, step counter k omitted)

4. Multiple-Model Method for Performance Model Tracking

We investigate various issues that occur during the use of IMM for the tracking of performance model.

Model set design

The first and most important job of IMM (and any MM method), is to design a set of different but complement behavior models, so that together they cover all system modes. The models can differ in performance model, process model, error covariance, or any combination of these.

For performance models, the queueing model and its variants are preferred because they have efficient solution methods, which are crucial in an online environment. IMM can be applied directly to a *homogeneous model set*. The performance models in such sets have the same model structure and share the same continuous-value parameters i.e. states, but they have different discrete-value attributes, whose derivatives ($\boldsymbol{H}$) are either undefined or difficult to calculate. The discrete-value attributes can involve service discipline (see Case 1), number of servers per queue, and load-dependent patterns [20]. In contrast, the models in the *heterogeneous model set* may have different structures, and the parameters and covariances in one model may need additional transformation in order to mix with those of other models during the interaction and combination steps. We will discuss the heterogeneous model set in future works.

For process models, we have two candidates: The drift model assumes the states drift independently around previous states (the least knowledge of state changes), i.e. $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{w}_k$ (where $\boldsymbol{w}_k$ is the Gaussian noise) [4], [9]. The autoregressive (AR) model assumes that the states follow local trend and the trend drifts randomly [6]. The simplest AR model is the first-order AR model, AR(1). It can be formulated as follows:

$$\hat{\boldsymbol{x}}_k = \boldsymbol{g}_{k-1} \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{w}_k, \quad \boldsymbol{g}_k = \boldsymbol{g}_{k-1} + \boldsymbol{\xi}_k \tag{20}$$

Here $\boldsymbol{g}_{k-1}$ is the first-order AR coefficient, and $\boldsymbol{w}_k$ and $\boldsymbol{\xi}_k$ are the Gaussian noises.

For error covariance, it's straight forward to form a set of models with different $\mathbf{Q}$ and $\mathbf{R}$. The values of $\mathbf{Q}$ and $\mathbf{R}$ shall match different levels of state disturbance and measurement noise in different time. Since different process models often assume different state disturbance, model sets with different process models often have different $\mathbf{Q}$ settings.

Setting the design parameters

IMM requires two sets of design parameters: The first is the per filter setting, which includes the initial state estimate $\boldsymbol{x}_0$,

its covariance $\boldsymbol{P}_0$, $\mathbf{Q}$, and $\mathbf{R}$. The other is the IMM-specific setting, which includes the initial model probability $\boldsymbol{\pi}_0^j$ and the transition matrix $\boldsymbol{\pi}$.

$\boldsymbol{x}_0$ can be set to match the average state values or the current estimate based on offline estimation. $\boldsymbol{P}_0$ is often set to a diagonal matrix with diagonal terms equal to the square of $\boldsymbol{x}_0$. Generally, $\boldsymbol{x}_0$ and $\boldsymbol{P}_0$ have little impact on the estimate.

$\mathbf{Q}$ and $\mathbf{R}$ are often set to diagonal matrices. Ideally, the diagonal terms should match the expected state disturbance and measurement noise, which are often expressed relative to the expected state and measurement values, as follows: $\mathbf{Q}_{ii} = (\text{Qfac} * \text{E}(\boldsymbol{x}(i)))^2$ and $\mathbf{R}_{ii} = (\text{Rfac} * \text{E}(\boldsymbol{z}(i)))^2$. Here i is the state or measure index, Qfac and Rfac are constants that are used as inputs to the EKF instead of $\mathbf{Q}$ and $\mathbf{R}$. $\mathbf{R}$ can also be set to measured variances from experiments under constant state values [4].

$\boldsymbol{\pi}_0$ represents the prior knowledge of model probability. If none is available it can be set to give identical probability to each model. For $\boldsymbol{\pi}$, if the expected sojourn time in model i is τ_i, then its diagonal terms can be set to $\pi_{ii} = 1 - \tau_i^{-1}$, and the *transition probability* $\pi_{ij,\, i\neq j}$ can be set to $\pi_{ij,\, i\neq j} = (1 - \pi_{ii})/(M - 1)$.

States with different dimensions

A problem develops during the IMM interaction and combination steps when the drift model is used with the AR model. The state vector of the AR model contains extra states that represent the previous states and AR coefficients that do not exist in the drift model. We overcome this problem by augmenting the state vector of the drift model $\hat{\boldsymbol{x}}_{k-1}^j$ in Eq. (12) and $\hat{\boldsymbol{x}}_k^j$ in Eq. (18) with suitable states to render it compatible with that of the AR model. These extra states are set as if the drift model were a first order AR model with a constant first order AR coefficient of 1. Similarly the corresponding drift model covariances $\boldsymbol{P}_{k-1}^j$ in Eq. (13) and $\boldsymbol{P}_k^j$ in Eq. (19) are augmented with columns and rows of zeros.

In general, if the state vectors of models differ in dimension, the lower dimension state vector can be augmented with suitable states to render them compatible. If such states are undefined, the compatible states can be mixed during the interaction and combination steps. This may degrade the tracking results because errors in incompatible states cannot be corrected during the interaction step and the overall state estimate may lose information from the models with shorter state vectors.

Linearization and its errors

EKF must calculate the derivatives of f and h in order to linearize process and measurement models around the current estimate. The derivatives of the drift and AR models can be calculated in closed form. For performance models, derivatives of common separable queueing models are also available in Zheng et al. [9]. In general, the derivatives of a performance model can be calculated using numerical differentiation:

$$\partial \text{h}/\partial x_i = (\text{h}(x_i + \delta) - \text{h}(x_i))/\delta \tag{21}$$

Here x_i is the ith state and δ is a small change.

The linearization errors can be quite large for h when large changes in states occur. IMM requires an EKF variant called the iterated extended Kalman filter (IEKF). This variant iteratively linearizes h to a new state estimate [7]. In an EKF iteration, more IEKF iterations can produce better linearization refinement. In this work, we use 5 IEKF iterations, which are sufficient to speedup the convergence of EKF estimation.

Performance prediction

There are two ways to use a tracked performance model for resource management: the *checkout* method and the *integrated* method. In the checkout method, one treats the tracked performance model at time k as the up-to-date model, then uses a separate load predictor for performance prediction after time k [10]. In this method, one assumes that the performance model parameters remain stationary in the near future, so a good tracking filter should produce accurate estimate of the states. In the integrated method, the load predictor is integrated into the tracking filter. One uses predicted performance from the tracking filter for management decisions [6]. In this method, the performance model parameters may vary substantially in the near future, and the tracking filter must focus more on the prediction accuracy. In IMM, the combined 1-step-ahead prediction of the states and measures are computed as follows:

$$\hat{\boldsymbol{x}}_{k+1|k}^- \triangleq E(x_{k+1}|\boldsymbol{z}_k) = \sum_i \hat{\boldsymbol{x}}_{k+1}^{-,i}\, \bar{u}_{k+1}^{-,i} \tag{22}$$

$$\hat{\boldsymbol{z}}_{k+1|k}^- \triangleq E(z_{k+1}|\boldsymbol{z}_k) = \sum_i \text{h}^i(\hat{\boldsymbol{x}}_{k+1}^{-,i})\bar{u}_{k+1}^{-,i} \tag{23}$$

1-step-ahead prediction may not provide enough foresight for tasks such as server provision which take substantial time to apply. One need multi-step-ahead prediction, i.e. $\hat{\boldsymbol{x}}_{k+\eta|k}^- \triangleq E(x_{k+\eta}|\boldsymbol{z}_k)$ and $\hat{\boldsymbol{z}}_{k+\eta|k}^- \triangleq E(z_{k+\eta}|\boldsymbol{z}_k)$. They can be obtained at time k by running the IMM recursion forward η times. During the forward recursion, no measurement is available, so the update steps of each elemental filters can be skipped and the predicted state can be treated as the a posteriori state estimate, i.e. $\hat{\boldsymbol{x}}_k^i \overset{assume}{=} \hat{\boldsymbol{x}}_k^{-,i}$.

Convergence conditions

IMM must satisfy a set of conditions in order to converge [21]. Here the convergence refers to that the estimation algorithm uniquely identifies states and modes in finite time.

First, IMM must ensure that the system is *observable* for *each* elemental Kalman filter. This requires that the Kalman filter meets the following rank condition [7]:

$$\text{rank}[\boldsymbol{H}^T, \ldots, (\boldsymbol{H}\boldsymbol{F}^i)^T, \ldots, (\boldsymbol{H}\boldsymbol{F}^{n-1})^T] = \dim(\boldsymbol{x}) \quad (24)$$

Here n is the number of time steps of the observable period, and $\dim(\boldsymbol{x})$ is the number of states. When the drift model is used as the process model, this implies that the number of measurements is greater than or equal to the number of states and that there is no state to which all measurements are insensitive. When the AR model is used, there is no need to collect extra measurements for the AR coefficients because they are estimated across time, through the trend. The rank condition is only approximate if either the measurement model or process model is nonlinear. If the rank condition is not met, as in the case of a multi-class performance model, then the technique introduced by Kumar et al. can be used to increase the number of measurements [22].

Second, the true model (or one that closely resembles to the true model) must be in the model set and unique. This implies that the models should cover all modes of the system and be very separate from each other. This separation should exhibit itself in the measurement residuals, especially between the filters based on the true model and mismatched models. Otherwise, the IMM will not be selective in terms of choosing the correct model. This may or may not be a problem depending on whether the real system model is explicitly desired. If it is, then the model set should be redesigned or more measurements collected. However, in most cases it is not, so IMM can be used to predict system performance regardless of whether the most probable model is the true model or not.

Time complexity

The time complexity of IMM is linear with the number of models, and the time complexity of EKF (the elemental filter) is dominated by the calculation of the performance model and its derivatives. If solving a performance model takes T, the number of states is n, and the number of IEKF iterations is J, then one iteration of IMM takes approximately $M(n+1)JT$, where the $n+1$ evaluations of performance models represent numerical differentiation. Because there are efficient analytical algorithms for queueing models and their extensions, the solution can be reached quickly. In the studies presented in the next section, one iteration of IMM takes less than 1 second, which is fast enough for common sampling intervals, such as 30 seconds [10]. IMM can be rendered even faster by running elemental filters parallelly in multiple CPU or cores during the filtering step.

In theory, the performance models are not limited to queueing models. Any model can be used provided that it has a well-defined sensitive matrix ($\boldsymbol{H}$). However, stochastic Petri nets and other state-based formalisms are not suitable for IMM because they do not tend to scale up well and because many of them require expensive computational simulation for solution [23].

5. Experiments

This section evaluates IMM in both simulated and real case studies. The simulation case studies demonstrate 3 typical usage of IMM. The real case study used a benchmark application to investigate the effectiveness of IMM under real system.

5.1 Simulation Enviroment

To simplify the presentation, we consider a web application as shown in Fig. 2, where the application server (AS) and the database server (DB) reside in the same node. A single class of 50 users alternates between *visiting* the site and *sleeping*. The system can be modeled using a closed queueing network with a delay center representing the user think time. Both the think time and the service time in the queueing model are exponentially distributed. The queueing model has two parameters which are the system states to be tracked:

$\boldsymbol{x}(1)$ = mean user think time between requests.

$\boldsymbol{x}(2)$ = mean service time of requests in the server CPU.

The user think time can change with the variation of request arrival rates. Higher arrival rates correspond to lower think time. The service time can change because of system reconfiguration, changes in the type of requests, and load-dependent behavior [20].

The performance measures include the mean system response time, CPU utilization of the server node, and system throughput, all of which are widely available. In the simulation, we obtained these data from the steady-state outputs of queueing models, which are solved analytically using approximate MVA. We simulated the sampling errors by adding zero-mean white Gaussian noise to the output of queueing models. This is reasonable because the measurement noise is mainly from statistical sampling errors, which are normally distributed [9]. The covariance matrix of the simulated noise is similar to those of assumed Gaussian noise in elemental filters ($\mathbf{Q}$ and $\mathbf{R}$).

Each simulation runs for 200 time steps, and both the states and modes of the system change. The system modes in steps (51–150) are different from those in other steps. The modes of the system are designed to change in the performance model, measurement noise, and process model. The use of IMM to track these three mode changes is illustrated in three case studies.

In each case study, we set up three tracking filters. Two are single-model-based EKFs designed to accurately model individual modes of the system. The third filter is a multiple-model Kalman filter based on IMM whose elemental fil-

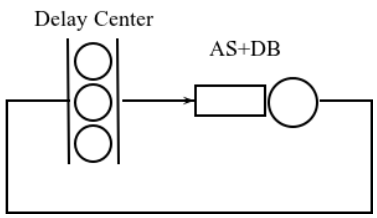


Fig. 2 A simple queueing model.

ters use the same filter setting as previous two single-model filters.

The initial model probability $\boldsymbol{\pi}_0$ is set to give identical probability to all models. The model transition matrix in IMM is as follows:

$$\boldsymbol{\pi} = \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix}$$

This amounts to expected mode duration of 100 steps. $\boldsymbol{x}_0$ is set to the real states and $\boldsymbol{P}_0$ is set to a diagonal matrix with diagonal terms equal to the square of $\boldsymbol{x}_0$. Qfac and Rfac default to 0.01, which serves as a baseline for further adjustment. The default Qfac is close to the default in [6]. It describes a standard deviation of state disturbance which is about 0.01 times the average state values. It prepares the filter for substantial state changes, but significant less than the average state values. The default Rfac describes a small measurement error, which is the statistical sampling error using the recommended sampling interval in [9].

The derivatives of process model are calculated analytically in closed form. These are identity matrices for the drift model; the derivatives of the performance models are calculated using numerical differentiation (Eq. (21)).

The effectiveness of the IMM is evaluated against single-model methods using the following tracking and prediction errors:

- **MARE(x):** mean absolute relative error between estimated and real states.
- **fMARE(x):** forward mean absolute relative error between predicted and real states.
- **fMARE(z):** forward mean absolute relative error between predicted and real measures.

fMARE defaults to 1-step-ahead prediction error, fMARE(z, η) is used to denote η-step-ahead prediction error. For a trace of K steps, the estimation and prediction error of ith state or measure is defined as:

$$\text{MARE}(\boldsymbol{x}_i) = \frac{1}{K}\sum_{k=1}^{K} |\hat{\boldsymbol{x}}_k(i) - \boldsymbol{x}_k(i)|/\boldsymbol{x}_k(i)$$
$$\text{fMARE}(\boldsymbol{x}_i, \eta) = \frac{1}{K-\eta}\sum_{k=1}^{K-\eta} |\hat{\boldsymbol{x}}^-_{k+\eta|k}(i) - \boldsymbol{x}_{k+\eta}(i)|/\boldsymbol{x}_{k+\eta}(i)$$
$$\text{fMARE}(z_i, \eta) = \frac{1}{K-\eta}\sum_{k=1}^{K-\eta} |\hat{z}^-_{k+\eta|k}(i) - z_{k+\eta}(i)|/z_{k+\eta}(i)$$

We report the average error over all states/measures. Note that fMARE(z) can evaluate the effectiveness of a measurement model more effectively than fMARE(x), but when the measurements are very noisy, fMARE(x) is more effective than fMARE(z).

Four questions about IMM are addressed in the following case studies:

1. Can the method track changes in the mode of the system?
2. How does the method compare to single-model methods in terms of tracking and prediction?
3. What is the influence of design parameters ($\mathbf{Q}$, $\mathbf{R}$, π)?
4. What if a mismatched model set is used? A model set is mismatched if none of models in the set is sufficiently accurate?

5.2 Case 1: Performance Model Changes

We consider the changes in the performance model, specifically the changes in the approximate service discipline for modeling server CPU. Two widely used service disciplines, PS and FCFS, are considered. The choice of discipline within a performance model has been shown to depend on the relative size of request service time and the OS scheduling time-slice [8]. As the request service time changes, the close-to-truth service discipline may also change. This motivates us to simulate a system that runs in one of the following two modes at different times:

1. PS mode: The service discipline of the server is PS. The service time is 10 ms and sleep time is 1 s.
2. FCFS mode: The service discipline of the server is FCFS. The service time is highly variable, and the squared coefficient of variation (SCV) is 10. The service time is 5 ms and sleep time is 0.5 s.

Rfac of simulated noise in both modes is set to 0.01, which matches the default Rfac setting for filters and describes an enviroment with low noise level. Note that the queueing model of FCFS mode does not have a product form solution. We solve it using the MAP-AMVA method [24].

We assume at first the modes are known to the modeler, so we model these two modes using exactly the same performance models as those used to generate simulation data. We also used the drift process model. Table 3 summarizes the IMM configuration.

Mode tracking

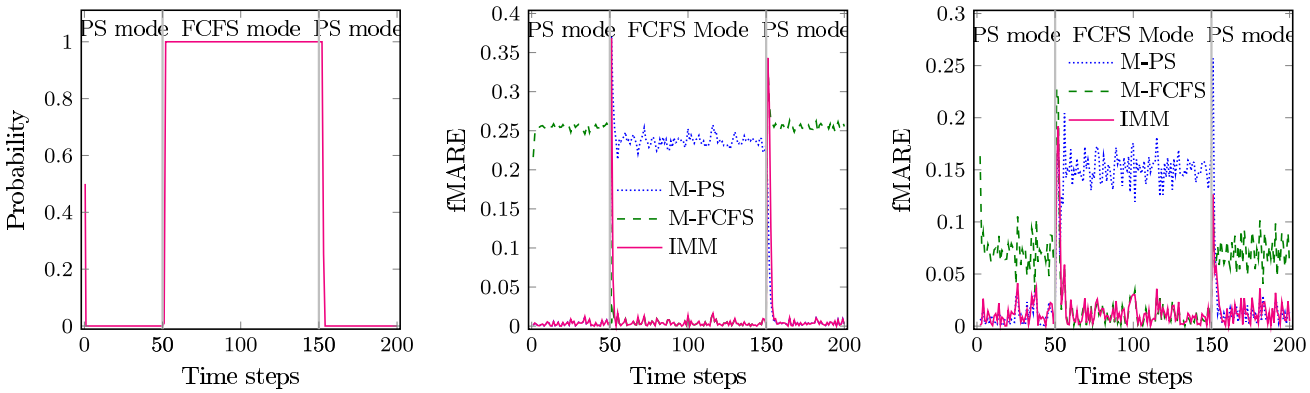
IMM's ability to track system mode changes can be evaluated using the posterior probabilities of its elemental filters. Figure 3 (a) shows that the estimated M-FCFS probabilities match expected FCFS mode probabilities closely. Even though the initial M-FCFS probability is 50%, it drops to nearly zero at the end of first step. Then IMM identifies the changes in modes after a delay of only a few steps (one step to switch to M-FCFS, and three steps to switch back to M-PS).

State tracking

While the system mode is being tracked, IMM can track the changes in states adaptively using different performance models. We show the service time tracking error of various filters in Fig. 3 (b). IMM tracks the real states of the system significantly better than the single-model filters. Further

Table 3 Settings of IMM elemental filters and their corresponding single-model filters in Case 1.

Filter	f	Qfac	h	Rfac
M-PS	Drift	0.01	PS queue	0.01
M-FCFS			FCFS queue (SCV = 10)	



(a) M-FCFS probabilities of IMM (b) Tracked service time error (c) Predicted response time error

Fig. 3 The system with performance model changes (Case 1).

Table 4 IMM state tracking errors in MARE(x)(%) of Case 1 when Qfac and Rfac vary.

Qfac	Rfac						
	0.01	0.02	0.05	0.1	0.2	0.5	1
0.01	0.82	4.79	6.31	8.68	12.70	19.64	21.68
0.02	0.81	0.87	5.15	6.20	7.74	14.24	18.55
0.05	1.22	0.99	0.81	1.48	2.94	8.54	13.07
0.1	1.28	1.22	0.86	0.87	2.00	6.87	10.03
0.2	1.30	1.28	1.16	0.83	1.90	6.41	8.33
0.5	1.30	1.30	1.27	1.22	2.10	6.27	7.26
1	1.30	1.30	1.29	1.28	2.35	6.36	6.90

investigation has shown that the states of the single-model filters converge to the wrong states when there is a model mismatch. When system mode and states changes together (step 51 and 151), the tracking error spikes. This is expected because IMM need to adjust the state estimates first before it can switch to a better model.

While both the states and modes are being closely tracked, IMM can give better performance prediction than the single-model filters can, as shown in Fig. 3 (c).

Influence of design parameters

In a single-model filter, smaller **Q** can make the filter more conservative thus less affected by noise, while smaller **R** can make the filter more aggressive thus react to changes faster. The behavior of a filter is determined by the relative ratio of **R** to **Q**, rather than their absolute values [4]. However, in the case of multiple-model filter, from the Eq. (16) we can see that **Q** and **R** have additional effect on the model likelihood. Larger values of **Q** and **R** can make the pdf of Gaussian noise more spread out thus make a mismatched model more probable.

To investigate the actual influence of **Q** and **R** on tracking results, Qfac and Rfac are varied over two orders of magnitude. Results are shown in Table 4. Above the diagonal of the table, the error increase steadily as Rfac/Qfac increases. The error can become large when Rfac/Qfac is 100. Below the diagonal, the error is much smaller and does not varies much. Along the diagonal, where ratio of **Q** to **R** is identical, the tracking error is nearly the same with Rfac from 0.01 to 0.2, but when Rfac > 0.2 the tracking error triples. After investigating the posterior probability of M-FCFS in FCFS mode, we find that when Qfac = Rfac = 0.5, the probability converges to 27% instead of one. The large tracking error caused by model mismatch is treated as measurement noise, which prevents IMM from switching to the better M-FCFS.

We also investigate the influence of the transition matrix $\boldsymbol{\pi}$ on state tracking. We use identical diagonal terms in $\boldsymbol{\pi}$, and repeat the tracking experiments with the transition probability ($\pi_{ij,i\neq j}$) range from 0.5 to 0.0001. The results show the tracking accuracy is not sensitive to the transition probability setting when the modes can be well approximated by the models in the model set. When a model mismatch exists, the large transition probability causes IMM to settle into a new model mixture more quickly at the cost of slightly higher tracking error in the mode in which no model mismatch exists.

Mismatched model sets

Often the accurate performance model of the system is unknown, thus here we consider the impact of mismatch between mode and model. The elemental filter M-FCFS is modified to have different SCV from the actual one. The impact of incorrect SCV settings on tracking and prediction is summarized in Table 5. We also list the results of a single-model filter (EKF) in the first two rows. Note that the performance of a PS queue is similar to that of a FCFS queue with SCV = 1. When SCV = 10, M-FCFS can match FCFS mode exactly, so the tracking and prediction error is minimal. When SCV = 5 or 15, in FCFS mode, IMM still switches to M-FCFS with a probability of one. This is the model that most closely resembles the real mode. The tracking and prediction results suffer, but tend to remain better than those of the single-model filters. Because the performance of real FCFS mode is somewhere between that of M-PS and M-FCFS when SCV = 15, a better estimate can be obtained by mixing their results more evenly. When IMM is configured with Qfac = Rfac = 0.2, M-FCFS probability converges to 73.74%, and the tracking and prediction results are good, close to the case of SCV = 10. However,

when Rfac > 0.2, IMM begins to treat the model mismatch as measurement noise and uses mismatched models in the output combination. This causes the accuracy of tracking and prediction to deteriorate.

5.3 Case 2: Changes in Measurement Noise

We here consider the case of changes in measurement noise caused by variations in the load. Here the measurement noise is mainly statistical sampling error, and it is higher when system throughput is low [9]. The simulated system runs in one of the following two modes at different times:

1. Noisy mode: The service time is 5 ms and sleep time is 1.5 s. System throughput is low, so level of measurement noise is high, and Rfac of simulated noise is set to 0.1.
2. Noise-free mode: The service time jumps to 15 ms and sleep time to 0.5 s. System throughput is rather high, so level of measurement noise is low, and Rfac of simulated noise is set to 0.01.

In IMM, we use two filters with different **Q** and **R** settings. The one with high level of measurement noise responds conservatively to changes in states and is used to model the noisy mode. The one with low level of measurement noise reacts aggressively to changes in states and is used to model the noise-free mode. Their levels of process noise also have different values. These two models are therefore well separated. Various **Q** and **R** settings are discussed elsewhere in this section. Table 6 summarizes the IMM configuration.

Table 5 The impact of model mismatch on state tracking and performance prediction of Case 1. M-FCFS in IMM uses various SCV and the FCFS mode uses SCV = 10. Qfac and Rfac default to 0.01.

Type	(M-FCFS) SCV	M-FCFS prob(%)	MARE(x)(%) PS mode	MARE(x)(%) FCFS mode	fMARE(z)(%) PS mode	fMARE(z)(%) FCFS mode
EKF	1	-	0.73	13.64	1.58	12.82
	10	-	16.59	0.46	11.82	1.36
IMM	10	99.01	1.03	0.60	1.63	1.30
	5	99.01	1.02	6.48	1.98	10.21
	15	99.50	1.09	4.56	1.81	7.69
	15[a]	73.74	1.09	1.44	1.90	2.29
	15[b]	56.60	2.64	3.36	4.04	2.66

[a] Qfac = Rfac = 0.2
[b] Qfac = Rfac = 0.5

Mode tracking

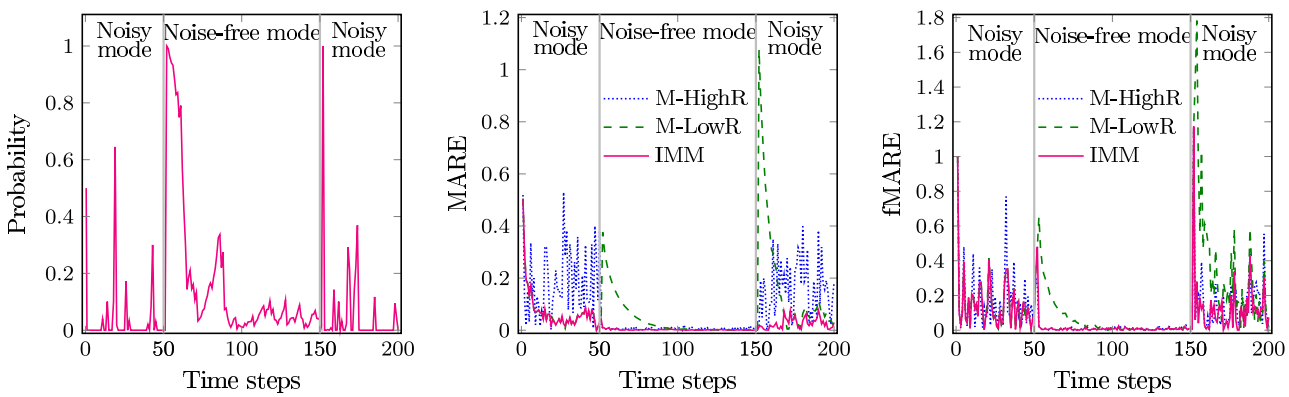
We investigate the estimated M-LowR probabilities (Fig. 4 (a)). The probabilities do not match the expected noise-free mode probabilities exactly. However, during the steps when changes in states occur (steps 51 and 151), IMM quickly switches to M-LowR, which allows IMM to track large changes in states in a timely manner. After step 152, IMM quickly switches back to M-HighR and remains mostly in M-HighR afterwards. This enables IMM to filter more noise in the noisy mode. We note that the spikes of M-LowR probability in the noisy mode occurs when the measurement noise happens to be small for several steps; the low probability of M-LowR after step 52 occurs because the actual states remain constant during the noise-free mode. As shown later, in both scenarios, the expected model fit the data well (has small tracking error), but the other model fits equally well or even better.

State tracking

Figure 4 (b) shows that IMM has optimal tracking results despite the discrepancy between estimated and expected model probabilities. During the noisy mode, the level of measurement noise is high, and M-LowR overreacts to the changes in performance measures and fails to reduce the noise. The tracked states fluctuate greatly with the noise (large MARE oscillates around 20%). In contrast, IMM continues to reduce noise by switching to the more conservative M-HighR. The spikes in M-LowR probability do not increase MARE

Table 6 Settings of IMM elemental filters and their corresponding single-model filters for Case 2.

Filter	f	Qfac	h	Rfac
M-HighR	Drift	0.01	PS queue	0.1
M-LowR		0.1		0.01



(a) M-LowR probabilities of IMM (b) Tracked service time error (c) Predicted throughput error

Fig. 4 The system with changes in measurement noise (Case 2).

because at those moments M-LowR also has low MARE. In steps 51 and 151, large changes in states occur. However, M-HighR is too sluggish to keep up with rapid changes (MARE remains large for many steps). In contrast, IMM can track the large changes in states by switching to the more aggressive M-LowR. We note here that the interaction step is crucial after step 151. A separate experiment (not reported here) showed that MM methods without this step, such as the autonomous MM [13], also switch to M-HighR but the states of M-HighR remain uncorrected, which causes inaccurate state estimation. From step 52 to 150, IMM unexpectedly switches to M-HighR, but this do not increase MARE because it is beneficial to trust the process model when the states remain constant. Figure 4 (b) suggests that as long as the model set covers the system modes well, IMM can have optimal tracking results.

We further investigate the performance prediction results in Fig. 4 (c). After the state changes in steps 51 and 151, the predicted throughput of M-HighR deviates from measurements greatly (fMARE remains large for many steps). This is because M-HighR fails to change the service time to the real one in a timely manner. In the noisy mode, M-LowR cannot follow the random changes in performance measures due to noise, its prediction always lags one step behind. This causes an inaccurate and unstable prediction (fMARE peaks at nearly 80%). In contrast, the prediction of IMM is both timely and stable.

We summarize the tracking and prediction results quantitatively in Table 7. In addition to the single-model filters and IMM, we included an optimal single-model filter (M*). This filter was tuned through experiments with different combinations of Qfac and Rfac from 0.01 to 1. As shown in the table, by combining two inaccurate elemental filters, M-HighR and M-LowR, IMM becomes even more accurate with respect to state tracking and prediction than the optimal single-model filter whose **Q** and **R** settings are not known ahead of time.

Influence of design parameters

Through repeated experiments similar to those in Case 1, we find that the transition probability ($\pi_{ij,i\neq j}$) has little influence on tracking. The tracking errors in terms of MARE(x) remain below 3% when transition probability ranges from 0.0001 to 0.1. However, starting at 0.2, errors begin to grow but remain smaller than those of the optimal single-model filter (4.64%). When the transition probability is large, IMM frequently alternates between models in the noisy mode, which can slightly interfere with the noise reduction. In the noise-free mode, the large transition probability causes the M-LowR probability to fluctuate around a constant of about 50% instead of 100%. This does not degrade the tracking results by much.

In Case 2, the only difference between two elemental filters is **Q** and **R**, so their impact on tracking can be said to be related to the problem of mismatched model sets, which is discussed as follows.

Mismatched model sets

The default R values of the two elemental filters in Case 2 are the same as those of corresponding simulated noise. In practice, the process noise and (to a lesser extent) the measurement noise can only be approximated. It is important to understand the impact that setting these noise covariances incorrectly in the elemental filters can have on tracking.

We fix the Qfac of M-HighR and Rfac of M-LowR to a low value (here 0.01), and vary other covariance factors from 0.01 to 1. The tracking experiments were performed and MARE(x) is given in Table 8. We find that IMM provides more accurate state tracking than the optimal single-model filter (whose MARE(x) is 4.64%) in 25 out of a total of 49 settings. The optimal tracking results (in bold) are produced when the Rfac of M-HighR is within 50–200% of the real Rfac and Qfac of M-LowR is not too small. The worst case occurs in the upper right corner, where M-HighR blindly trusts the process model (too conservative) and M-LowR does not trust measurement enough (not aggressive enough).

This experiment suggests that the model set design should avoid including a filter with a large Rfac/Qfac ratio and always include a filter with a small Rfac/Qfac ratio. The filter with large Rfac/Qfac tends to overtrust the process model, so the inclusion of such filter can cause IMM to respond slowly to the sudden change in states. The filter with small Rfac/Qfac is very sensitive to the state changes, so the inclusion of such filter can serve as a defensive strategy against rapid changes.

Table 7 State tracking and prediction errors of IMM comparing with its elemental filters and an optimal single-model filter in Case 2.

Filter	MARE(x)(%)			fMARE(x)(%)		
	Noisy mode	Noise-free mode	Overall	Noisy mode	Noise-free mode	Overall
M-HighR	11.65	5.07	8.36	12.81	6.65	9.72
M-LowR	14.00	0.77	7.39	15.12	2.03	8.54
M*[a]	8.49	0.79	4.64	15.52	1.63	8.54
IMM	4.26	0.25	2.26	5.34	1.79	3.57

[a]Optimal single-model filter, Qfac = Rfac = 0.01

Table 8 The state tracking errors in MARE(x)(%) of Case 2 when IMM uses a M-HighR with varying Rfac and a M-LowR with varying Qfac. Rfac of M-LowR and Qfac of M-HighR fixed at 0.01. Optimal regions are shown in **bold**.

Qfac of M-LowR	Rfac of M-HighR						
	0.01	0.02	0.05	0.1	0.2	0.5	1
0.01	4.64	3.70	3.77	6.69	14.78	18.73	32.01
0.02	5.64	3.70	3.16	5.40	14.69	18.77	32.23
0.05	6.76	4.63	**2.38**	**2.36**	**2.60**	19.55	26.93
0.1	7.24	5.91	**2.92**	**2.26**	**2.32**	6.55	27.48
0.2	8.10	7.04	**2.88**	**2.63**	**2.43**	4.15	4.58
0.5	8.44	7.33	**2.79**	**2.70**	**2.76**	6.00	4.19
1	8.34	7.02	**2.89**	**2.72**	**2.83**	4.21	7.52

5.4 Case 3: Process Model Changes

We consider two process models: Drift and AR. AR models excel in the prediction of short-term state, but they have a significant limitation. The state trends must continue for one the order of steps in order to enable stable tracking [6]. In contrast, the drift model can track large state changes separated by just a few steps, but it cannot predict trends [3]. We show that IMM can combine the strength of these two models for tracking time-varying system. We simulate a system that has the following two modes:

1. Smooth mode. The system states undergo a slow sinusoid variation. The service time is 10 ms and sleep time is 2 s.
2. Chaotic mode. The system states drift randomly, but keep system utilization below 80%.

Rfac of simulated noise in both modes is set to 0.02, which describes a slightly more noisy enviroment. Such noisy enviroment needs a better process model to filter noise. The state changes are illustrated in Fig. 5.

We use two filters with different process models in IMM: one filter had a first-order AR model and a low process noise level for the state changes in the smooth varying region, the other filter had a drift model and a high process noise level for the chaotic region. Table 9 summarizes the IMM configuration.

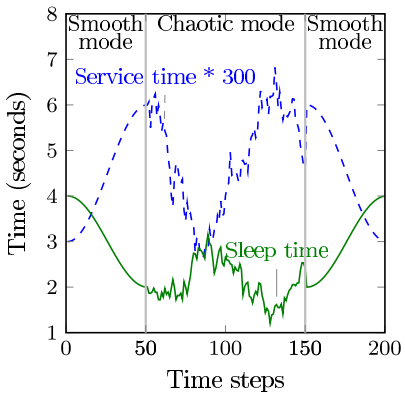


Fig. 5 Real states.

Mode tracking

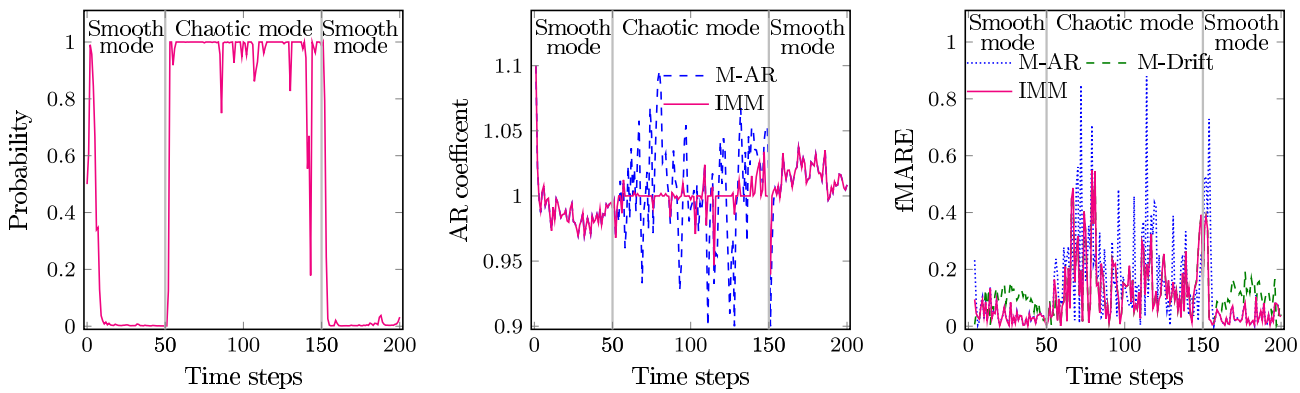
As in Case 1, the M-Drift probabilities match the expected chaotic mode probabilities closely (Fig. 6 (a)). In particular, after the AR model learns the trend in the first few steps of the smooth mode, IMM quickly identifies the AR model as the more probable model. After the system goes into the chaotic mode and the trend oscillates, IMM quickly switches to the drift model, which fits the state changes better. During the chaotic mode, IMM occasionally tries to switch to the AR model (M-Drift probabilities drop) when the state changes trend happens to be consistent in several steps, which means IMM is agile enough to utilize AR model for better state prediction in a short stable period.

State tracking

Because the measurement noise is quite small, the state tracking error is also quite small for all models, MARE is around 1%. The AR model has extra states that correspond to the state changes trend. We investigate the estimated trend of single-model M-AR filter and that of elemental filter M-AR in IMM (Fig. 6 (b)). During the smooth mode, because IMM switches to the AR model, the trend estimated is the same as that estimated using the single-model AR filter (around 0.98 when the sleep time is decreasing and 1.02 when it is increasing). During the chaotic mode, however, M-AR in IMM differs significantly with the single-model AR filter: The single-model AR filter cannot follow the transient trend of state changes, and it fails to find a stable value

Table 9 Settings of IMM elemental filters and their corresponding single-model filters in Case 3.

Filter	f	Qfac	h	Rfac
M-AR	AR(1)	0.01	PS queue	0.02
M-Drift	Drift	0.1		



(a) M-Drift probabilities of IMM (b) Tracked sleep time AR coefficient (c) 3-step-ahead utilization prediction error

Fig. 6 The system with process model changes (Case 3).

Table 10 Prediction errors of IMM relative to their elemental filters and an optimal single-model filter in Case 3.

Filter	fMARE(z,1)(%)			fMARE(z,3)(%)		
	Smooth mode	Chaotic mode	Overall[a]	Smooth mode	Chaotic mode	Overall
M-AR	2.85	10.95	7.03	5.29	20.95	14.42
M-Drift	3.23	9.10	6.28	7.28	14.74	12.19
M-AR*[b]	2.67	9.39	6.17	4.46	16.51	11.34
IMM	2.42	9.15	5.91	3.38	14.91	10.42

[a]This is not solely an average value. It contains predictions across the mode boundary.

[b]The optimal M-AR. Qfac for real states are 0.1. Qfac for AR coeff. are 0.01

Table 11 3-step-ahead prediction errors in fMARE(z,3)(%) of Case 3 when IMM uses M-AR and M-drift with varying Qfac. Rfac of both models fixed at 0.02. Optimal regions are shown in bold.

Qfac of M-AR	Qfac of M-Drift						
	0.01	0.02	0.05	0.1[a]	0.2	0.5	1
0.001	13.45	12.61	12.24	**10.43**	**10.39**	**10.41**	**10.42**
0.002	12.22	12.61	**10.28**	**10.18**	**10.20**	**10.22**	**10.24**
0.005	**10.86**	**10.42**	**10.21**	**10.20**	**10.22**	**10.27**	**10.30**
0.01[b]	13.58	**10.79**	**10.43**	**10.42**	**10.48**	**10.52**	**10.60**
0.02	15.30	15.30	**10.99**	**10.68**	**10.75**	**10.99**	11.44
0.05	16.42	17.11	16.93	11.97	14.12	15.52	15.81
0.1	17.03	17.87	16.72	12.83	12.74	16.76	16.82

[a]Real Qfac of Chaotic mode

[b]Real Qfac of Smooth mode

(AR oscillates around 1). In contrast, M-AR in IMM is almost stable at a value of 1 as a result of the interaction (IMM interaction step) with the dominating M-Drift model, which always assumes trend coefficient of value 1. The difference among filters in trend estimation leads to different levels of prediction accuracy, especially with respect to multi-step-ahead predictions.

The 3-step-ahead utilization prediction error is shown in Fig. 6 (c). In the smooth mode, the trend remains stable, and M-AR can utilize estimated trends to produce better state predictions. (Its fMARE is smaller than that of M-Drift.) In the chaotic mode, the state changes randomly, and the estimated trend of M-AR becomes unreliable. As a result, the estimated states of M-AR often overshoot the real state, which causes large, frequent prediction errors of more than 60%. These inaccurate predictions may trigger false alarms and cause unnecessary resource management activities. Instead of following transient trend, M-Drift projects the current state estimate constantly into the future. This has been shown to be more accurate in the chaotic mode. (Its fMARE is smaller than that of M-AR.) By dynamically switching between M-AR and M-Drift, IMM can produce accurate predictions in both modes.

We summarize prediction results quantitatively in Table 10. In addition to M-AR, M-Drift, and IMM, we also included an optimal M-AR (M-AR*), which was obtained through experiments with different Qfac from 0.001 to 0.1. IMM produces the best predictions of any the filters studied here.

Influence of design parameters and mismatched model sets

Because the AR(1) and drift model are already approximation of true state evolution process, the only design parameters are **Q**. We vary Qfac of from 0.001 to 1 and find that as long as the Qfac of M-Drift is 10 times larger than that of M-AR, optimal prediction accuracy can be obtained (Table 11). The two exceptions are shown in the upper left and bottom right corners of Table 11. In the upper left corner, the Qfac of M-Drift is close to the real Qfac in the smooth mode, therefore M-Drift is more probable than the expected model, M-AR, whose Qfac is too small. In the bottom right corner, the Qfac of M-AR is close to the real Qfac of the chaotic mode, therefore M-AR is more probable than the expected model, M-Drift, whose Qfac is too large. When designing a model set with different process models, the suggestion is to ensure that **Q** of a process model matches the magnitude of changes in states of the target operating region or at least matches it better than any other process model in the set.

As in Case 2, transition probability was found to have little influence on the tracking and prediction results.

5.5 Case 4: Validation on a Practical System

TPC-W is a standard web-based benchmark. It implements the basic functions of online bookstores. We deployed TPC-W on a platform consisting of Tomcat 6.0 and MySQL 5.1. Tomcat and MySQL run on two virtual machines using Ubuntu 10.04 as the OS. The workload was generated using 100 emulated browsers in a separate server. The performance model of the system is a closed product-form queueing network consisting of 2 PS queues representing web and DB CPU, and 1 delay center representing network delay. The disk operation is not considered in this case. All servers in our test-bed has one 2.60 GHz Pentium dual-core processor, and are interconnected using switched ethernet.

The benchmark contains 14 different requests. We used the most common shopping mix as the workload. To reflect the real workload variation, we used the ClarkNet access log [26], collected from a web server of a commercial Internet provider, to guide the load generation. The think time of the emulated browsers was changed every minute so that the arrival rates, scaled down with a constant factor, matched the one of web page requests in the ClarkNet log. The portion of ClarkNet log used was from Aug. 28 to Aug. 29. The traffic of ClarkNet shows typical diurnal pattern, which increases gradually to a peak in the early afternoon, then drops gradually to a lowest level in the early morning.

Besides the gradually changes in traffic volume, practical system can also undergo sudden large changes in states as a result of system reconfiguration and software upgrade. We injected large changes in states by adding a controlled CPU-intensive loop in the code of web server servlets. As a result, the service time of requests in the web server doubled after 6 pm and changed back to the original one after 6 am.

We used the model set of Case 3. The AR(1) model

Table 12 3-step ahead prediction errors in fMARE(z,3)(%) of IMM relative to their elemental filters and an optimal single-model filter in Case 4.

	Smooth mode	Chaotic mode	Overall
M-AR	4.75	14.86	6.77
M-Drift	7.54	8.89	7.81
M-AR*[a]	5.20	11.82	6.52
IMM	4.41	9.23	5.37

[a]The optimal M-AR. Qfac = 0.04

with a small Qfac is used to track the gradual changes in think time and service time due to load dependent behavior. The drift model with a larger Qfac is used to track the sudden changes in service time. The CPU utilization was collected using Sysstat tools, and the end-to-end response time and throughput was collected in the load generator. As in [6], we chose a relative large sampling interval of 15 min. Since the real traffic as in the ClarkNet log is quite bursty, we smoothed the data by calculating a moving average with a window of 3 samples before feeding it into the tracking filters.

Unlike the simulation experiments, the real states are unknown, therefore we evaluate the predictive ability of different filters using only fMARE(z, 3). There are 2 potential modes in the system. Normally, the system is in a smooth mode where states change gradually with the variation of traffic. There is also a transient chaotic mode where system states undergo sudden large changes. We use "chaotic mode" to refer to the subsequent 10 time steps after the injected state changes. As shown in Table 12, in overall, IMM has significant smaller prediction error than its elemental filters and the optimal single-model filter. In particular, in the chaotic mode, by temporally switching the process model to the drift model, the prediction error of IMM decreases by 38% (14.86 vs 9.23). In other settings of Qfac and Rfac for elemental filters, IMM shows similar accuracy improvement. The findings are consistent with those found through simulation.

6. Discussion

This work presents the first successful attempt to adapt Kalman filter models when tracking parameters of performance models. Our case studies show that the proposed IMM based method can improve tracking and prediction accuracy significantly. Setting the parameters for IMM is not difficult, and a wide range of parameters are adequate to produce optimal tracking results. However, the setting of **Q** and **R** is more restrictive than in single-model filters. As shown in Case 1, the absolute value of **Q** and **R** can influence the probabilities of models, so **Q** and **R** should not deviate greatly from the real change magnitude observed in the data. IMM also renders the parameterization of filters easier because one can include multiple models with different **Q** and **R** values. In particular, one can include a filter that has small **Q** which is considered dangerous in single-model filter [9]. This remains true as long as one also includes a filter that has a large **Q** value. The extra design parameter introduced by IMM, the model transition matrix π, is easy to set. The transition probabilities ranging from 0.0001 to 0.1 can consistently give good tracking results.

IMM is not limited to two models. Rather, it can easily scale to a model set with 10 models as long as the computation load is acceptable [15]. However, because large model sets are difficult to design, smaller ones are often preferable.

The biggest problem in MM methods is the problem of designing a set of models that complement each other and cover various system modes. Our three case studies offer a good start, but more development is needed. For performance models, heterogeneous model sets are possible. These include queueing networks with different numbers of queues and different structures, even the combination of queueing models and blackbox models [27]. For process models, the drift and AR models are the only serious options. Models that consider the correlation between different states could be useful for tracking multi-class performance models. For error covariance, experimental methods such as those mentioned previously may be used to determine multiple **Q** and **R** settings [4].

The model set design problem is more difficult for performance models with complex structure and large numbers of parameters. Each queue may have different behavior mode (such as service discipline). Each parameter may vary in manner and magnitude. Each measurement may have different noise levels. To model all the behavioral uncertainties, the model set must include all possible combinations of modes. The number of combinations can be quite large which is not desirable. In practice, one should identify and model the behavior uncertainties that have the largest impact on system performance.

IMM assumes that the system mode can be represented sufficiently accurately at any time by one of the models in the set. Problems may arise if none of the models is accurate enough. We investigate one such case in Case 1. IMM can give estimates as accurate as the best single-model filter. Mixing the models more evenly by properly increasing measurement covariance **R** can give even better results. Further work is required to determine the conditions under which mixing models evenly are optimal. Another possible solution is to use the so-called *variable structure MM* methods [13]. These methods can generate models that fit measurements better than existing models in the set and remove unlikely models from the set dynamically.

References

[1] J. Guitart, J. Torres, and E. Ayguade, "A survey on performance management for internet applications," Concurrency and Computation: Practice and Experience, vol.22, no.1, pp.68–106, Jan. 2010.
[2] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi, "CPU demand for web serving: Measurement analysis and dynamic estimation," Performance Evaluation, vol.65, no.6-7, pp.531–553, June 2008.
[3] M. Woodside and M. Litoiu, "The use of optimal filters to track parameters of performance models," 2nd Int'l Conf. on the Quantitative Evaluation of Systems (QEST'05), pp.74–83, 2005.

[4] T. Zheng, J. Yang, M. Woodside, and M. Litoiu, "Tracking time-varying parameters in software systems with extended Kalman filters," Proc. 2005 Conf. of the Centre for Advanced Studies on Collaborative research, pp.334–345, 2005.
[5] D. Kumar, A. Tantawi, and L. Zhang, "Real-time performance modeling for adaptive software systems," Proc. 4th Int'l ICST Conf. on Performance Evaluation Methodologies and Tools, pp.11:1–11:10, Sept. 2009.
[6] T. Zheng, M. Litoiu, and M. Woodside, "Integrated estimation and tracking of performance model parameters with autoregressive trends," Proc. 2nd ACM Int'l Conf. on Perf. Engineering, p.157, New York, USA, 2011.
[7] D. Simon, Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches, Wiley-Interscience, 2006.
[8] J. Happe, H. Groenda, and R. Reussner, "Performance evaluation of scheduling policies in symmetric multiprocessing environments," 2009 IEEE Int'l Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, pp.1–10, Sept. 2009.
[9] T. Zheng, M. Woodside, and M. Litoiu, "Performance model estimation and tracking using optimal filters," IEEE Trans. Softw. Eng., vol.34, no.3, pp.391–406, May 2008.
[10] M. Woodside and M. Litoiu, "Service system resource management based on a tracked layered performance model," 2006 IEEE Int'l Conf. on Autonomic Computing, pp.175–184, 2006.
[11] A. Moghaddamjoo and R. Kirlin, "Robust adaptive Kalman filtering with unknown inputs," IEEE Trans. Acoust. Speech Signal Process., vol.37, no.8, pp.1166–1175, 1989.
[12] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan, Estimation with Applications to Tracking and Navigation, Wiley-Interscience, 2001.
[13] V. Jilkov, "Survey of maneuvering target tracking. part v: multiple-model methods," IEEE Trans. Aerosp. Electron. Syst., vol.41, no.4, pp.1255–1321, Oct. 2005.
[14] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients," IEEE Trans. Autom. Control, vol.33, no.8, pp.780–783, 1988.
[15] R.R. Pitre, "A comparative study of multiple-model algorithms for maneuvering target tracking," Proc. SPIE, pp.549–560, 2005.
[16] P.J. Bickel, C.A. Klaassen, Y. Ritov, and J.A. Wellner, Efficient and Adaptive Estimation for Semiparametric Models, Springer, 1998.
[17] H. Zou and Y. Yang, "Combining time series models for forecasting," Int. J. Forecasting, vol.20, no.1, pp.69–84, Jan. 2004.
[18] R. Wolski, N.T. Spring, and J. Hayes, "The network weather service: a distributed resource performance forecasting service for metacomputing," Future Generation Computer Systems, vol.15, no.5-6, pp.757–768, Oct. 1999.
[19] S. Fekri, M. Athans, and A. Pascoal, "Issues, progress and new results in robust adaptive control," Int. J. Adaptive Control and Signal Processing, vol.20, no.10, pp.519–579, Dec. 2006.
[20] D. Kumar, L. Zhang, and A. Tantawi, "Enhanced inferencing: estimation of a workload dependent performance model," Proc. 4th Int'l ICST Conf. on Performance Evaluation Methodologies and Tools, Pisa, Italy, pp.47:1–47:10, 2009.
[21] I. Hwang and H. Balakrishnan, "Observability criteria and estimator design for stochastic linear hybrid systems," Proc. IEE Euro. Control Conf., pp.1–11, Cambridge, UK, 2003.
[22] D. Kumar, A. Tantawi, and L. Zhang, "Estimating model parameters of adaptive software systems in real-time," Autonomic Systems, pp.45–71, 2010.
[23] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni, "Model-based performance prediction in software development: a survey," IEEE Trans. Softw. Eng., vol.30, no.5, pp.295–310, May 2004.
[24] G. Casale and E. Smirni, "MAP-AMVA: Approximate mean value analysis of bursty systems," 2009 IEEE/IFIP Int'l Conf. on Dependable Systems & Networks, pp.409–418, June 2009.
[25] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," ACM SIGMETRICS Perf. Eval. Review, vol.33, no.1, pp.291–302, June 2005.
[26] M.F. Arlitt and C.L. Williamson, "Web server workload characterization: The search for invariants," Proc. 1996 ACM SIGMETRICS Int'l Conf. on Measurement and modeling of computer systems, New York, USA, pp.126–137, 1996.
[27] E. Thereska and G.R. Ganger, "IRONModel: Robust performance models in the wild," ACM SIGMETRICS Perf. Eval. Review, vol.36, no.1, pp.253–264, 2008.

Zhen Zhang received B.S in computer science and technology from Zhejiang University, Hangzhou, China in 2006. He is currently a Ph.D. candidate under supervision of Prof. Shanping Li. His research interests include performance modeling, resource management and distributed applications.

Shanping Li received the M.S. and Ph.D. degrees in the computer science and technology from Zhejiang University, Hangzhou, China in 1987 and 1993, respectively. He is currently a professor in the College of Computer Science and Technology at Zhejiang University. His research interests include the distributed computing, very large information system and software engineering.

Junzan Zhou received B.S in computer science and technology from Zhejiang University, Hangzhou, China in 2008. He is currently a Ph.D. candidate under supervision of Prof. Shanping Li. His research interests include performance testing, anomaly detection and cloud computing.