On-line Bayesian Context Change Detection in Web Service Systems

Jakub M. Tomczak*
Institute of Computer Science
Wybrzeże Wyspiańskiego 27
50-370, Wrocław, Poland
jakub.tomczak@pwr.wroc.pl

Maciej Zięba
Institute of Computer Science
Wybrzeże Wyspiańskiego 27
50-370, Wrocław, Poland
maciej.zieba@pwr.wroc.pl

ABSTRACT

In real-life situations characteristics of Web service systems evolve in time. Therefore, change detection techniques become substantial elements of adaptive procedures for Web service systems management, such as resource allocation and anomaly detection methods. In this paper, we propose an on-line change detector which uses the Bayesian inference. We define two models which describe situations with one change and no change within data. Next we apply Bayesian model comparison for change detection. In order to obtain analytical expressions of *model evidences* used in the model comparison we provide a coherent framework of change detection which focuses on an approximation of the *Bayes factor*. The proposed solution, contrary to state-of-the-art methods, works in an on-line fashion and the algorithm's computational complexity is proportional to the constant size of the shifting window. Low computational complexity of the change detector enables its application in complex computer networks. At the end of the research paper, the quality of the proposed algorithm is examined using simulated Web service system.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Time series analysis; H.3.5 [**Online Information Services**]: Web-based services; I.5.4 [**Pattern Recognition**]: Applications

General Terms

Algorithms, Performance

Keywords

Change detection; Web service; BIC

*Corresponding author.

HotTopiCS'13, April 20–21, 2013, Prague, Czech Republic.

1. INTRODUCTION

In this paper, we explore the issue of context change detection in web service systems for maintaining quality of service execution. Web service systems consist of the following layers [41]: (i) an *execution layer* which controls the execution of composite Web services and manages dataflow between them, and (ii) an *application service layer* which delivers requested functionalities to clients. Web services are designed according to service oriented computing (SOC) paradigm [30] and represent encapsulated functionalities of applications.

Change detection is the identification of abrupt or gradual changes in the probability distribution of sequential data [1]. Generally, detection of changes in Web service systems is made due to two reasons: (i) to maintain high quality of service by controlling resource allocation process [36, 46], (ii) to detect anomalies [7, 13].

Change detection has proven to be a useful signal processing tool in applications such as sound processing [8], learning of driving behavior [26], video segmentation [35], environmental time series analysis [15, 33], biomedical signal analysis [19], anomaly detection [13, 42], outlier detection [18], adaptive resource allocation [36, 43, 46] and many others [4, 16]. There are two main groups of methods for change detection problem, namely, frequentist and Bayesian approaches.

The first approach focuses on hypothesis testing basing on *likelihood functions* [4, 16] or *dissimilarity measures* between probability distributions, e.g., Kullback-Leibler divergence, Bhattacharyya measure [22, 36, 43], and recently *Support Vector Machines* [10]. Probability distributions are estimated using data in two consecutive shifting windows. Such estimation technique enables frequentist detectors to work in an on-line fashion. Nevertheless, frequentist approach is very sensitive to the size of shifting windows. In case of an improper size of the windows the distribution estimation results in false alarms.

On the other hand, most of Bayesian change detection methods are retrospective, i.e., changes are detected after collecting all data. In general, these methods apply *Bayesian model comparison* via *Bayes factor* to select a model which represents possible changes. The idea of the Bayesian modelling is to treat all quantities occurring in the model, i.e., variables and parameters, as random variables. Hence, the number of changes as well as the length of periods between consecutive changes are random quantities. Eventually, the model with the highest value of *model evidence* (*marginal likelihood*) is selected to determine the number of changes

in the shifting window. Such approach was applied to exponential family of distributions [24], with a special concern on Poisson processes [12, 34]. To overcome the disadvantage of working in an off-line manner, several Bayesian on-line detectors have been proposed. In [28] a linear Gaussian model with conjugate priors is considered and the parameters are estimated according to recursive expressions based on Woodbury matrix inverse update formula. Recently, recursive estimation procedure for general *a posteriori* distributions have been proposed. One applies *particle filter* to sample from distributions which are analytically intractable [11]. Other method, called *Bayesian Online Change Detection* (BOCD) [1], focuses on modelling length of periods between two consecutive changes and the inference procedure is based on the *message-passing algorithm* and sampling methods like *Markov Chain Monte Carlo* (MCMC). Next, in order to extend the family of distributions that can be modelled using BOCD, *Gaussian processes* were used [37]. Nevertheless, all Bayesian on-line detectors use sampling methods to estimate a posteriori distributions and thus they are computationally insufficient in network applications.

In this paper, we present a Bayesian change detection method which is able to work in an on-line fashion and consumes a small amount of computational resources. We apply the Bayesian model comparison for two models, i.e., one with no change occurrence, and second with one change. We use one shifting window for which model evidences are calculated. Further, we make an assumption that the change in the second model occurs in the middle of the shifting window. Additionally, we assume that the *a priori* distributions of the parameters are non-informative. According to these assumptions we get an approximation of the Bayes factor which is in fact the *Bayesian Information Criterion* (BIC) [38, 40]. Hence, we obtain a closed form of Bayes factor which can be calculated easily and applying a sampling method is not needed to compute *a posteriori* distributions. Our main contribution is to use a Bayesian inference for change detection in computer networks so that all calculations are performed on-line. In addition, we use multinomial random variables as a flexible method of modelling network quantities which does not require pre-defined forms of the probability distributions. For the multinomial random variables we provide the BIC expression.

The paper is organized as follows. We present the main contribution of this paper in Sect. 2. We give preliminary notations and definitions in Sect. 2.1. Next, we describe details regarding Bayesian inference for change detection. In Sect. 2.3, we calculate the BIC for the considered models. Thereafter, we outline the algorithm for change detection. In Sect. 3, experiments are presented. Section 4 concludes the research paper.

2. ON-LINE BAYESIAN CONTEXT CHANGE DETECTION

2.1 Preliminaries

The aim of a Web service system is to meet clients' requests in order to guarantee required level of *Quality-of-Service* (QoS) attributes such as latency, response time, price, availability, or reliability [14, 29, 30, 45]. Therefore, the execution system needs to react to changes in its working regime, e.g., to reduce increasing latency or keep high reliability by preventing intrusions. In other words, the execution systems should be self-adaptive, i.e., be able to reallocate resources and reorganize itself.

Let us assume that we observe a quantity $x \in \mathcal{X}$, e.g., $\mathcal{X} = \mathbb{R}_+$, which describes the QoS of the execution system[1] such as latency. Latency refers to the time delay between the initial input and the output of the execution system [29, 45]. Additionally, we make an assumption that the execution system is influenced by a hidden circumstance called *context* [17, 21, 23]. The context is denoted by $c_n \in \mathcal{C}$ and is not observable which means that neither the value of the variable c_n nor $\mathcal{C}$ are known. Further, we consider all variables as random quantities and thus the model can be seen as a *Hidden Markov Model* [5]. The system can be described using probability distributions as follows[2,3]:

$$p(c_n|c_{n-1}), \tag{1}$$

$$p(x|c_n) \tag{2}$$

with initial latent variable:

$$p(c_1). \tag{3}$$

However, the context is unobservable and no rational assumptions can be made because of unknown $\mathcal{C}$. Hence, we deal with *non-stationarity* [44] because (2) cannot be calculated without knowing c_n.

Introduction of a quantity responsible for the context allows us to treat all causes of changes as a *context change* [7]. Independently on the domain, a context change is a change in the probability distribution (2) due to evolving c_n. Therefore, we do not make any distinction between the reasons for the change. In case of changes caused by anomalies possible values of the context c_n may reflect a normal working regime of the execution system or abnormal situations caused by, e.g., reduction of computational resources or temporary server failure. However, in case of evolving streams of Web service requests the context values reflect clients' behavior [9].

2.2 Context change modelling

Let us consider a random variable $x \in \mathcal{X}$, $\mathcal{X} = \{1 \dots K\}$, i.e., x is a discrete random variable[4]. Next, let us introduce a K-dimensional binary random variable $\mathbf{x}$ having a 1-of-K representation [5], in which a particular element x_k is equal to 1 and all other elements are equal to 0. The values of x_k thus must satisfy $x_k \in \{0, 1\}$ and $\sum_k x_k = 1$. In other words, instead of $x = k$ we will use $x_k = 1$.

[1]For simplicity of further reasoning and to keep the notation uncluttered we consider a single variable only. However, an extension to a high-dimensional formulation of the presented approach is straightforward.

[2]In case of higher order dependency, the equation (1) should be replaced with $p(c_n|c_{n-m}, c_{n-m+1}, \dots, c_{n-1})$, for $1 < m < n$.

[3]Arguments of probability distributions differentiate distributions, e.g., $p_x(x) \equiv p(x)$ and $p_y(y) \equiv p(y)$.

[4]Presented approach can be easily applied to continuous variables. However, in order to avoid choosing specific form of a probability distribution we consider a discrete variable. Such approach is especially useful if we consider anomaly detection and anomaly is defined as, e.g., *if the value of x is greater than A, then report an anomaly*, or, in case of resource allocation, if we are interested in maintaining given level of QoS.

We further assume that there is given a sequence of N observations $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. This sequence may be divided into non-overlapping *product partitions* [3]. In other words, there are $M < N$ context changes within the data, that is,

$$\underbrace{\mathbf{x}_1 \ldots \mathbf{x}_{n_1}}_{\hat{c}_1} \underbrace{\mathbf{x}_{n_1+1} \ldots \mathbf{x}_{n_2}}_{\hat{c}_2} \ldots \underbrace{\mathbf{x}_{n_M+1} \ldots \mathbf{x}_N}_{\hat{c}_M} \tag{4}$$

where n_m denotes the end of m^{th} context. Context is constant for $n = n_{m-1}+1, \ldots, n_m$, $m = 1, \ldots, M$ and equals $\hat{c}_m$, where $\hat{c}_m \in \mathcal{C}$.

We further assume that the data within each m^{th} partition is independent and identically distributed from the probability distribution $p(\mathbf{x}|\hat{c}_m)$, that is,

$$\mathcal{D}_m = \{\mathbf{x}_n : \mathbf{x}_n \sim p(\mathbf{x}|\hat{c}_m), n = n_{m-1}+1, \ldots, n_m\} \tag{5}$$

where $\sim$ is a symbol denoting that observations are drawn from a given probability distribution.

The probability distribution of $\mathbf{x}$ for given $\hat{c}_m$ is as follows:

$$p(\mathbf{x}|\boldsymbol{\theta}_m) = \prod_{k=1}^{K} \left(\theta_{k,m}\right)^{x^k}, \tag{6}$$

where $\boldsymbol{\theta}_m = \left(\theta_{1,m} \ldots \theta_{K,m}\right)^\top$ and $\theta_{k,m}$ fulfills the following constraints: $\theta_{k,m} \geq 0$ and $\sum_k \theta_{k,m} = 1$. Hence, $\theta_{k,m}$ denotes probability $p(x_k = 1|\hat{c}_m)$. Further, to simplify the notation, we use $\boldsymbol{\theta}_m = \boldsymbol{\theta}$, and $\theta_{k,m} = \theta_k$.

In the presented approach we consider a shifting window which contains L observations starting from $(n-L+1)^{\text{th}}$ moment to the current n^{th} moment, i.e.,

$$\mathcal{D}_n^L = \{\mathbf{x}_{n-L+1}, \ldots, \mathbf{x}_n\}. \tag{7}$$

Further, we assume two possible situations: (i) the shifting window contains data from one partition (no context change), (ii) the shifting window contains data from two partitions (one context change). Hence, there are two models reflecting these circumstances:

1. If there is no context change in $\mathcal{D}_n^L$, we conjecture that data is generated from a model $\mathcal{M}_0$ and its likelihood function is as follows:

$$\begin{aligned} p(\mathcal{D}_n^L|\mathcal{M}_0, \boldsymbol{\theta}_0) &= p(\mathcal{D}_n^L|\boldsymbol{\theta}_0) \\ &= \prod_{k=1}^{K} \left(\theta_{0,k}\right)^{j_k}, \end{aligned} \tag{8}$$

   where $\boldsymbol{\theta}_0$ – parameters of $\mathcal{M}_0$, j_k – number of occurrences of $x_k = 1$ in $\mathcal{D}_n^L$.

2. If there is one context change in $\mathcal{D}_n^L$ at $t < n$, we conjecture that data is generated from a model $\mathcal{M}_1$ and its likelihood function is as follows:

$$\begin{aligned} p(\mathcal{D}_n^L|\mathcal{M}_1, \boldsymbol{\theta}_1, t) &= p(\mathcal{D}_t^{L-n+t-1}|\boldsymbol{\theta}_1^1)\ p(\mathcal{D}_n^{n-t+1}|\boldsymbol{\theta}_1^2) \\ &= \Big\{\prod_{k=1}^{K} \left(\theta_{1,k}^1\right)^{j_k^1}\Big\}\Big\{\prod_{k=1}^{K} \left(\theta_{1,k}^2\right)^{j_k^2}\Big\}, \end{aligned} \tag{9}$$

   where $\boldsymbol{\theta}_1 = (\boldsymbol{\theta}_1^1\ \boldsymbol{\theta}_1^2)^\top$ – parameters of $\mathcal{M}_1$, $\boldsymbol{\theta}_1^1$ are parameters for partition before context change, and $\boldsymbol{\theta}_1^2$ – parameters after context change, j_k^i – number of occurrences of $x_k = 1$, $i = 1$ – before context change, $i = 2$ – after context change. We assume that $\boldsymbol{\theta}_1^1, \boldsymbol{\theta}_1^2, t$ are independent random variables.

In order to select one model which is *more probable* to generate observed data we need to calculate *model evidences* [20]. The model evidence of $\mathcal{M}_0$ can be calculated as follows:

$$p(\mathcal{D}_n^L|\mathcal{M}_0) = \int p(\mathcal{D}_n^L|\mathcal{M}_0, \boldsymbol{\theta}_0)\ p(\boldsymbol{\theta}_0|\mathcal{M}_0)\ \mathrm{d}\boldsymbol{\theta}_0, \tag{10}$$

where $p(\boldsymbol{\theta}_0|\mathcal{M}_0)$ – *a priori* probability distribution of parameters. The model evidence of $\mathcal{M}_1$ is the following (assuming the independence of $\boldsymbol{\theta}_1^1, \boldsymbol{\theta}_1^2, t$)

$$\begin{aligned} p(\mathcal{D}_n^L|\mathcal{M}_1) = \iint p(\mathcal{D}_n^L|\mathcal{M}_1, \boldsymbol{\theta}_1, t)\ p(\boldsymbol{\theta}_1^1|\mathcal{M}_1)\ p(\boldsymbol{\theta}_1^2|\mathcal{M}_1) \times \\ p(t|\mathcal{M}_1)\ \mathrm{d}\boldsymbol{\theta}_1\ \mathrm{d}t, \end{aligned} \tag{11}$$

where $p(\boldsymbol{\theta}_1^1|\mathcal{M}_1)$, $p(\boldsymbol{\theta}_1^2|\mathcal{M}_1)$, $p(t|\mathcal{M}_1)$ – *a priori* probability distributions of parameters.

To calculate model evidences, we need to determine the *a priori* distributions of parameters and next to integrate out the parameters. For the considered likelihood functions (8) and (9) the calculation of integrals is troublesome because of choosing the form of the distribution for t, $p(t|\mathcal{M}_1)$. In the next section we discuss this problem.

Let us assume for a moment that the model evidences are known and the *a priori* probabilities of models are equal, $p(\mathcal{M}_0) = p(\mathcal{M}_1)$. Then, to compare both models, we can calculate the Bayes factor [20]:

$$B_{10} = \frac{p(\mathcal{D}_n^L|\mathcal{M}_1)}{p(\mathcal{D}_n^L|\mathcal{M}_0)}. \tag{12}$$

The Bayes factor denotes a summary of the evidence provided by the data in favor of $\mathcal{M}_1$ as opposed to $\mathcal{M}_0$ [20]. For value of the Bayes factor greater than 1 we get more evidence in favor of $\mathcal{M}_1$. Jeffreys has given an interpretation of the Bayes factor (see Table 1).

Table 1: Jeffrey's interpretation of the Bayes factor [20].

B_{10}	$\ln(B_{10})$	Evidence in favor of $\mathcal{M}_1$
$1-3$	$0-1.1$	Weak
$3-10$	$1.1-2.3$	Substantial
$10-100$	$2.3-4.6$	Strong
>100	>4.6	Decisive

As we will see shortly, it is more convenient to calculate logarithm of the Bayes factor, i.e.,

$$\ln(B_{10}) = \ln p(\mathcal{D}_n^L|\mathcal{M}_1) - \ln p(\mathcal{D}_n^L|\mathcal{M}_0). \tag{13}$$

2.3 Model evidence approximation

In order to obtain analytical expressions for model evidences of $\mathcal{M}_0$ and $\mathcal{M}_1$ we make the following assumptions:

- the *a priori* probability distributions of $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ are considered to be non-informative;
- the context change occurs in the middle of the shifting window, i.e., $\lceil n - \frac{1}{2}L \rceil$, [5] hence the *a priori* probability distribution of t is a Dirac delta function in the point $\lceil n - \frac{1}{2}L \rceil$.

[5] $\lceil \cdot \rceil$ denotes the ceil function.

For the assumptions described above we can approximate the model evidence for $\mathcal{M}$ by the Bayesian Information Criterion (BIC) [38, 40]

$$\ln p(\mathcal{D}_n^L|\mathcal{M}) \approx \ln p(\mathcal{D}_n^L|\hat{\boldsymbol{\theta}}) - \frac{|\mathcal{M}|}{2}\ln|\mathcal{D}_n^L|, \tag{14}$$

where $\hat{\boldsymbol{\theta}}$ is the maximum likelihood estimator of $\boldsymbol{\theta}$ calculated using $\mathcal{D}_n^L$, $|\mathcal{M}|$ denotes the number of parameters of the model $\mathcal{M}$, and $|\mathcal{D}_n^L|$ is the number of data in the shifting window.

The log likelihood function for the model $\mathcal{M}_0$ is given by

$$\begin{aligned}\ln p(\mathcal{D}_n^L|\boldsymbol{\theta}_0) &= \ln\prod_{k=1}^{K}\Big(\theta_{0,k}\Big)^{j_k}\\ &= \sum_{k=1}^{K} j_k\theta_{0,k}.\end{aligned} \tag{15}$$

For the model $\mathcal{M}_0$ the number of parameters equals $|\mathcal{M}_0| = K$ and the number of observations is $|\mathcal{D}_n^L| = L$. Thus, we get the following approximation of the logarithm of the model evidence for $\mathcal{M}_0$:

$$\ln p(\mathcal{D}_n^L|\mathcal{M}) \approx \sum_{k=1}^{K} j_k\hat{\theta}_{0,k} - \frac{K}{2}\ln L, \tag{16}$$

where the maximum likelihood estimators are as follows $\hat{\theta}_{0,k} = \frac{j_k}{L}$, for $k = 1, \ldots, K$.

Analogously, the log likelihood function for the model $\mathcal{M}_1$ is given by

$$\ln p(\mathcal{D}_n^L|\boldsymbol{\theta}_1) = \ln\Big\{\prod_{k=1}^{K}\Big(\theta_{1,k}^1\Big)^{j_k^1}\Big\}\Big\{\prod_{k=1}^{K}\Big(\theta_{1,k}^2\Big)^{j_k^2}\Big\} \tag{17}$$

$$= \sum_{k=1}^{K}\Big(j_k^1\ln\theta_{1,k}^1 + j_k^2\ln\theta_{1,k}^2\Big). \tag{18}$$

For the model $\mathcal{M}_0$ the number of parameters equals $|\mathcal{M}_1| = 2K$ and the number of observations is $|\mathcal{D}_n^L| = L$. Hence, we get the approximation of the logarithm of the model evidence for $\mathcal{M}_1$ in the following form:

$$\ln p(\mathcal{D}_n^L|\mathcal{M}) \approx \sum_{k=1}^{K}\Big(j_k^1\ln\hat{\theta}_{1,k}^1 + j_k^2\ln\hat{\theta}_{1,k}^2\Big) - K\ln L, \tag{19}$$

where the maximum likelihood estimators are as follows: for the first part of the window $\hat{\theta}_{1,k}^1 = \frac{j_k^1}{L/2}$, $k = 1, \ldots, K$, and for the second part of the window $\hat{\theta}_{1,k}^2 = \frac{j_k^2}{L/2}$, $k = 1, \ldots, K$.

Finally, we can approximate the Bayes factor (13) using (16) and (19), that is,

$$\ln B_{10} \approx \sum_{k=1}^{K}(j_k^1\ln\hat{\theta}_{1,k}^1 + j_k^2\ln\hat{\theta}_{1,k}^2) - \sum_{k=1}^{K} j_k\ln\hat{\theta}_{0,k} - \frac{K}{2}\ln L. \tag{20}$$

2.4 Change detection algorithm

Having an analytic form of the approximated Bayes factor allows us to select one of the two considered models. The selected model indicates whether the context change occurred in the shifting window $\mathcal{D}_n^L$ or not. We can propose the following on-line algorithm for change detection which use the Bayesian model comparison. In the first step move the shifting window and determine the model evidences (16) and (19) using $\mathcal{D}_n^L$. Then, calculate the approximated Bayes factor using (20). The final step is to report the context change if the Bayes factor is greater than a given value $\sigma \in \mathbb{R}_+$ called *a sensitivity parameter*. Determination of the σ is crucial in the proposed approach and may be seen as a compromise between detecting true changes and avoiding false alarms [16]. Nevertheless, we can use the Jeffrey's interpretation to determine the value of the sensitivity parameter (Table 1).

Let us denote a sequence of context change detections by τ. The final procedure of the change detection is presented in Algorithm 1.

Algorithm 1: Change detection using approximated Bayes factor

Input : $\mathcal{D}$, L, $\mathcal{M}_0$, $\mathcal{M}_1$, σ
Output: Moments of context change $\tau_1, \ldots, \tau_M$
$n \longleftarrow L$, $m \longleftarrow 0$, $\tau_0 \longleftarrow 0$;

while $n < \text{card}\{\mathcal{D}\}$ **do**
  Calculate $\ln p(\mathcal{D}_n^L|\mathcal{M}_0)$ using (16);
  Calculate $\ln p(\mathcal{D}_n^L|\mathcal{M}_1)$ using (19);
  Calculate $\ln B_{10}$ using (20);
  if $\ln B_{10} > \sigma$ **then**
    $m \longleftarrow m + 1$;
    $\tau_m \longleftarrow n - \lceil L/2 \rceil$;
    $n \longleftarrow n + \lceil L/2 \rceil$;
  else
    $n \longleftarrow n + 1$;
  end
end

The computational complexity of Algorithm 1 for each new observation is proportional to the size of the shifting window, i.e., $O(L)$. In order to calculate the model evidences we need to have the number of occurrences of $x_k = 1$ which requires to read all values of L observations in the shifting window only once.

3. APPLICATION TO WEB SERVICE SYSTEMS

The purpose of the experiment is to examine the change detection quality of the proposed algorithm. We take under consideration a Web service execution environment and average latency of services' responses in the system as a QoS attribute. To reflect the nature of real Web service execution system we propose a simulation model designed in a discrete events simulation environment *Arena* [2]. The simulation model is presented in Fig. 1. The model consists of the following components: (i) *request generator*, which imitates clients' behaviour by generating Web service requests, (ii) *scheduler*, which distributes service requests to proper computational nodes, (iii) *computational nodes*, where services are executed. Each computational node consists of virtual machines and each virtual machine contains instances of Web services.

It is worth noting that the proposed detector works independently on the web service network model complexity. Therefore, in the simulation process we use two computational nodes with eight processors each, two virtual machines

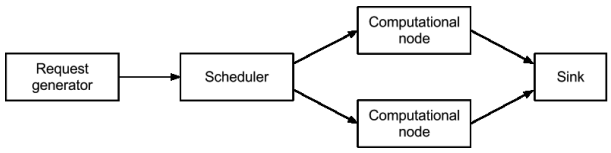

Figure 1: Simulation model of a Web service execution system.

on each node and four services. In addition, the presented method can be used to detect changes in characteristics of single computational node as well as of the entire execution system.

3.1 Modelling request generator

Typically, in the telecommunication theory, streams of requests are modelled by Poisson process. However, research conducted by Paxson and Floyd on traffic traces for Wide Area Networks (WANs) shows that user-initiated "session" arrivals are well-modelled by Poisson processes [31]. Nevertheless, packet-level traffic streams may deviate considerably from Poisson processes, and may exhibit non-stationarity and self-similarity over different time scales.

In our simulation model we generate Web service requests from non-stationary Poisson process. This way of generating requests is methodologically correct in service oriented systems only if we assume that clients invoke Web services independently and requests are delivered to the execution system as as single message, e.g., as SOAP messages. Basing on the presented assumptions we can treat Web service requests in the same manner as user-initiated "session" arrivals.

3.2 Modelling computational nodes

The computational nodes are modelled basing on architectural paradigms for modelling complex Web server networks presented in [27]. We assume that computational nodes are represented by Web servers with processors as computational resources. We estimate the values of processing parameters for Web servers basing on benchmark research described in [25]. The *Small Static File (KeepAlive)* test[6] is used to compare performance of servers such as *Apache* or *LiteSpeed*. We set the processing delays for Web servers equal 0.4 milliseconds and for virtual machines – 0.8 milliseconds, by averaging the results gained in testing procedure described in [25]. Additionally, we assume that each of two Web servers in the model uses eight processors which are assigned to virtual machines in the following manner:

- on the first server six processors are used by the first virtual machine and the remaining two by the second one;
- on the second server both virtual machines use four processors.

3.3 Modelling Web services

In the simulation model we consider four Web services located on virtual machines. We model performance of real data processing services implemented in service oriented data mining system described in [32]. Selected services represent the functionality of building the following classification models: *Naive Bayes*, *Logistic Regression*, *J48* and *Multilayer Perceptron*. We assume that the processing time for each of selected services is modelled with triangular distribution. We estimate the parameters of distributions for each service using soapUI tool [39]. For constant size of input message (16 kB) we gain the following processing times:

- *Multilayer Perceptron*: minimum processing time = 51 [ms], maximum processing time = 672 [ms], average processing time = 88 [ms];
- *Logistic Regression*: minimum processing time = 28 [ms], maximum processing time = 214 [ms], average processing time = 45 [ms];
- *J48*: minimum processing time = 5 [ms], maximum processing time = 183 [ms], average processing time = 11 [ms];
- *Naive Bayes*: minimum processing time = 6 [ms], maximum processing time = 53 [ms], average processing time = 10 [ms].

In addition, we assume that the processing time is divided by the number of processors assigned to virtual machine on which considered Web service is executed. Basing on the estimated distributions for Web services we arbitrarily allocate resources in the following manner:

- *Multilayer Perceptron* service allocation: first virtual machine on the first Web server and first virtual machine on the second Web server (total number of ten processors).
- *Logistic Regression* service allocation: first virtual machine on the first Web server and first virtual machine on the second Web server (total number of ten processors).
- *J48* service allocation: second virtual machine on the first Web server and second virtual machine on the second Web server (total number of six processors).
- *Naive Bayes* service allocation: second virtual machine on the second Web server (total number of four processors).

3.4 Simulation scenarios

The following three simulation scenarios are considered:
1. *Slight context change.* The context changes periodically (5 times per simulation). For each change the intensity parameters of Poisson process are increased thrice. After fixed period of time the intensities are decreased thrice (Fig. 2).
2. *Significant context change.* The context changes periodically (5 times per simulation). For each change the intensities of Poisson process are increased sixfold. After fixed period of time the intensities are decreased sixfold (Fig. 3).
3. *Processors failure (anomaly).* Anomaly is gained by failure of 4 processors on the first virtual machine (Fig. 4).

In the first two cases changes in latency characteristics are caused by non-stationary clients' requests arrival rate to the execution system. The last case represents a typical anomaly detection problem in which changes are caused by temporary resource unavailability.

[6]The main idea of the test is to send continuously requests of constant size (100B) to the Web server and to monitor connection status.

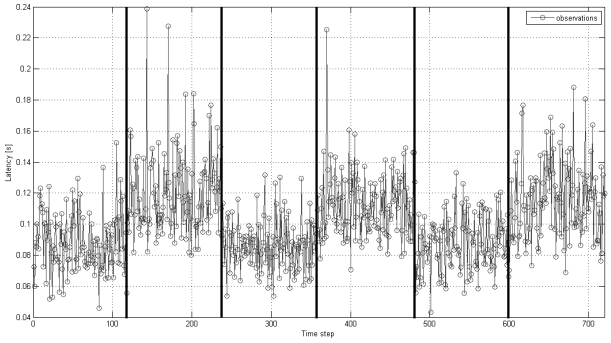

Figure 2: Exemplary characteristic of average latency in modelled execution system for *Slight context change* scenario. Moments of changes in the system are represented by vertical lines.

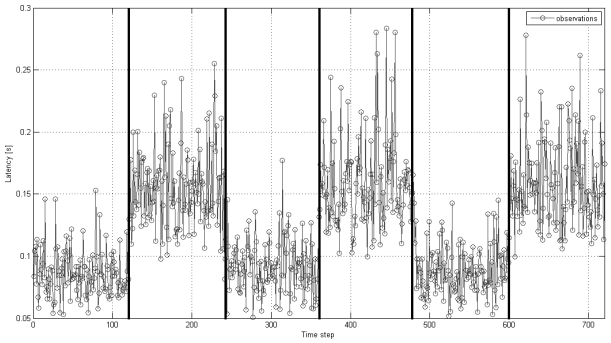

Figure 3: Exemplary characteristic of average latency in modelled execution system for *Significant context change* scenario. Moments of changes in the system are represented by vertical lines.

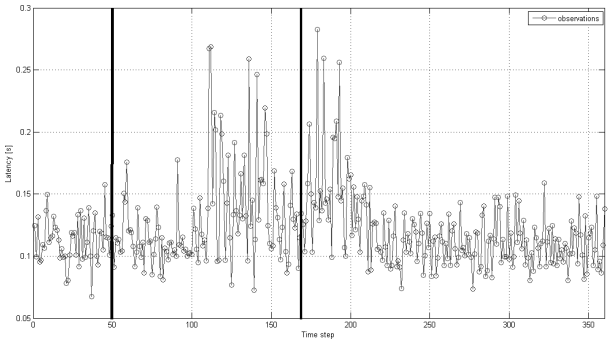

Figure 4: Exemplary characteristic of average latency in modelled execution system for *Processors failure* scenario. Moments of changes in the system are represented by vertical lines.

3.5 Results and Discussion

The exemplary characteristics of average latency in execution system for considered simulation scenarios are presented in Figures 2–4. For each case the moments of changes are known before the simulation, thus it is easy to asses if the change is correctly detected. To evaluate the change detection approaches we use two quality criteria, namely, the number of correctly and incorrectly detected changes. We assume that a change is correctly detected if it is reported with a delay less than ten time units (which are seconds in the simulation). Tables 2, 3, 4 contain the experimental results for simulation scenarios *Slight context change*, *Significant context change* and *Processors failure*, respectively. For each scenario simulation is repeated ten times and the obtained results are averaged. We compare the performance of the Bayesian approach for change detection with the following frequentist-based dissimilarity measures [6]: Bhattacharyya, Kullback-Leibler, Lin-Wong and modified Lin-Wong. The values of parameters for the frequentist approaches, i.e., size of shifting window (denoted by L) and sensitivity parameter (denoted by σ), are obtained empirically.

In general, the results of the experiment show that most of the changes were successfully detected using all of considered approaches. The lowest number of detected changes was gained for *Slight context change* and *Processors failure* simulation scenarios. In *Slight context change* case a slight increase of intensity of Web service requests provokes latency changes which are negligible by detectors (see Figure 2). In *Processors failure* simulation scenario, after gradual increase of average latency caused by system collapse, the

Table 2: Average number of correctly and incorrectly detected changes for *Slight context change* simulation.

Change detection method (L,σ)	Correctly detected (max. 5)	Incorrectly detected
Bhattacharyya (25, 0.2)	3.2	0.2
Kullback-Leibler (25, 1)	3.8	0.8
Lin-Wong (25, 0.15)	2.8	0.7
mod. Lin-Wong (25, 0.02)	2.9	0.9
Bayesian approach (25, 0)	3	0.2

Table 3: Average number of correctly and incorrectly detected changes for *Significant context change* simulation.

Change detection method (L,σ)	Correctly detected (max. 5)	Incorrectly detected
Bhattacharyya (25, 0.2)	4.6	0.1
Kullback-Leibler (25, 1)	4.8	0.2
Lin-Wong (25, 0.15)	4.6	0.3
mod. Lin-Wong (25, 0.02)	4.6	0.2
Bayesian approach (25, 0)	5	0

Table 4: Average number of correctly and incorrectly detected changes for *Processors failure* simulation.

Change detection method (L,σ)	Correctly detected (max. 2)	Incorrectly detected
Bhattacharyya (25, 0.2)	1	0.3
Kullback-Leibler (25, 1)	0.7	0.3
Lin-Wong (25, 0.15)	1.1	0.1
mod. Lin-Wong (25, 0.02)	1	0.1
Bayesian approach (25, 0)	1.1	0.1

execution system returned automatically to the level before the failure due to smart scheduling Web requests among execution units. Therefore, it is difficult to detect reparation moment of system failure (see second vertical line in Fig. 4).

The comparison of performance of considered change detectors shows that the Bayesian approach performed slightly better for *Significant context change* and *Processors failure (anomaly)* scenarios in comparison to the frequentist methods. Moreover, the number of incorrectly detected changes using the Bayesian model was the lowest for all considered scenarios. This feature of the Bayesian approach is particularly important in Web service execution systems in which each notification of change detection implies computationally demanding resource allocation. Last but not least it is worth mentioning that the Bayesian approach, in contrast to frequentist-based methods, does not require defining additional parameters except shifting windows size. The value of the sensitivity parameter can be set according to the Jeffrey's interpretation (see Table 1) without any additional empirical tuning.

4. CONCLUSIONS

In this paper, we presented the change detection method using the approximated Bayes factor. The main advantages of the proposed algorithm is that it works in the on-line fashion and the algorithm's computational complexity is $O(L)$ because of the closed forms of the model evidences. Moreover, the usage of a discrete random variable unifies the approach for different applications, e.g., in resource allocation or anomaly detection.

The proposed algorithm was applied in the Web service system in order to detect changes in the QoS parameter, i.e., latency. Three different situations were considered, namely, recurrent slight change in the mean of the random process, recurrent significant change in the mean of the random process, and the system failure. The obtained results (see Tables 2, 3, and 4) indicate high efficiency of the proposed change detector.

It is worth noting that the proposed approach can be easily applied to continuous random variables. For example, considering normal distributions leads to very similar method to that presented in [8] but with the on-line working regime. Moreover, our method can be successfully applied not only in the Web service systems but in other domains as well. Especially in domains in which the computational complexity of the change detector needs to be small, e.g., like in computer vision (background change detection) or on-line signal segmentation.

5. ACKNOWLEDGMENTS

The presented research is partially supported by the fellowship co-financed by European Union within European Social Fund.

6. REFERENCES

[1] Adams R.P., MacKay D.J.C., Bayesian Online Changepoint Detection, Technical Report, University of Cambridge, Cambridge, UK, arXiv:0710.3742v1 [stat.ML] (2007)

[2] Arena® Sim. Soft., http://www.arenasimulation.com/.

[3] Barry D., Hartigan J.A., Product partition models for change point problems, Annals of Statistics, Vol. 20, No. 1, pp. 260–279 (1992)

[4] Basseville M., Nikiforov I.V., Detection of Abrupt Changes: Theory and Application, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1993)

[5] Bishop C.M., Pattern Recognition and Machine Learning, Springer-Verlag, Singapore (2006)

[6] Cha, S.-H., Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions, Int J Math Models Methods Appl Sci Vol. 1, No. 4, pp. 300–307 (2007)

[7] Chandola V., Banerjee A., Kumar V., Anomaly detection: A survey, ACM Comput Surv, Vol. 41, Issue 3, pp. 1–58 (2009)

[8] Chen S.S., Gopalakrishnam P.S., Speaker, environment and channel change detection and clustering via the Bayesian Information Criterion, in: Proc. 1998 DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne, VA, pp. 127–132 (1998)

[9] Cox L.A. jr, Data Mining and Causal Modeling of Customer Behaviors, Telecommun Syst, Vol. 21, No. 2–4, pp. 349–381 (2002)

[10] Desobry, F., Davy, M., Doncarli, C., An Online Kernel Change Detection Algorithm, IEEE Trans Signal Process, Vol. 53, No. 8, pp. 2961–2974 (2005)

[11] Fearnhead P., Exact and efficient Bayesian inference for multiple changepoint problems, Stat Comput, Vol. 16, pp. 203–213 (2006)

[12] Green P.J., Reversible jump Markov Chain Monte Carlo computation and Bayesian model determination, Biometrika, Vol. 82, No. 4, pp. 711–732 (1995)

[13] Grzech A., Anomaly detection in distributed computer communication systems, Cybern Syst: Int J, Vol. 37, No. 6, pp. 635–652 (2006)

[14] Grzech A., Świątek P., Complex Services Availability in Service Oriented Systems, IEEE Int. Conf. on Systems Engineering (ICSEng), pp. 227–232 (2011)

[15] Gyarmati-Szabó J., Bogachev L.V., Chen H., Modelling threshold exceedences of air pollution concentrations via non-homogeneous Poisson process with multiple change-points, Atmospheric Environ, Vol. 45, pp. 5493–5503 (2011)

[16] Gustafsson F., Adaptive Filtering and Change Detection, John Wiley & Sons (2000)

[17] Harries M.B., Sammut C., Horn K., Extracting hidden context, Mach Learning, Vol. 32, pp. 101–126 (1998)

[18] Hido S., Tsuboi Y., Kashima H., Sygiyama M., Kanamori T., Statistical outlier detection using direct density ratio estimation, Knowl Inf Syst, Vol. 26, pp. 309–336 (2011)

[19] Kaplan A.Y., Shishkin S.L., Application of the change-point analysis to the investigation of the brain's electrical activity, in: Non-Parametric Statistical Diagnosis: Problems and Methods, eds. Brodsky B.E, Darkhovsky B.S., Kluwer Academic Publishers, Dordrecht, pp. 333–388 (2000)

[20] Kass R.E., Raftery A.E., Bayes factors, J Am Statistical Association, Vol. 90, No.430, pp. 773–795 (1995)

[21] Katakis I., Tsoumakas G., Vlahavas I., Tracking recurring contexts using ensemble classifiers: an application to email filtering, Knowl Inf Syst, Vol. 22, No. 3, pp. 371–391 (2010)

[22] Kifer D., Ben-David S., Gehrke J., Detecting change in data streams, in Proc. of the 30th Int Conf on Very Large Data Bases Toronto, Canada (2004)

[23] Kubat M., Induction in time-varying domains: Motivation, origins, and encouragements, 11th IEEE Int Conf and Workshop on the Engineering of Computer-Based Systems, pp. 316–321 (2004)

[24] Lee C.-B., Bayesian analysis of a change-point in exponential families with applications, Computational Statistics & Data Analysis, Vol. 27, pp. 195–208 (1998)

[25] Lite Technologies. Web server performance comparison: Litespeed 2.0 vs., http://litespeedtech.com/web-server-performance-comparison-litespeed-2.0-vs.html

[26] Maye J., Triebe R., Spinello L., Siegwart R., Bayesian on-line learning of driving behaviors, Int Conf on Robotics and Automation, pp. 4341–4346 (2011)

[27] van der Mei R.D., Hariharan R., Reeser P.K., Web server performance modelling, Telecommun Syst, Vol. 16, pp. 316–378 (2001)

[28] Ó Ruanaidh J.J., Fitzgerald W.J., Pope K.J., Recursive Bayesian location of a discontinuity in time series, IEEE Int Conf on Acoustics, Speech, and Signal Processing, pp. 513–516 (1994)

[29] de la Ossa B.A., Sahuquillo J., Pont A., Gil J.A., Key factors in web latency savings in an experimental prefetching system, J Intell Inf Syst, DOI 10.1007/s10844-011-0188-x (2011)

[30] Papazoglou, M.P., Georgakopoulos, D., Service-Oriented Computing, Communications of the ACM, Vol. 46, No. 10, pp. 25–28 (2003)

[31] Paxson V., Floyd S., Wide-area traffic: The failure of Poisson modeling, IEEE/ACM Trans on Networking, Vol. 3, No. 3, pp. 226–244 (1995).

[32] Prusiewicz A., Zięba M., The Proposal of Service Oriented Data Mining System for Solving Real-Life Classification and Regression Problems, IFIP Advances in Information and Communication Technology, Vol. 349, pp. 83–90 (2011)

[33] Raftery A.E., Extreme Value Analysis of Environmental Time Series: An Application to Trend Detection in Ground-Level Ozone: Comment: Are Ozone Exceedance Rates Decreasing?, Statistical Sci, Vol. 4, No. 4, pp. 378–381 (1989)

[34] Raftery A.E., Akman, V.E., Bayesian Analysis of a Poisson Process with a Change-Point, Biometrika, Vol. 73, No. 1, pp. 85–89 (1986)

[35] Ranganathan A., PLISS: Detecting and Labeling Places using Online Change-point Detection, Proceedings of Robotics: Science and Systems (2010)

[36] Rygielski P., Tomczak J.M., Context change detection for resource allocation in service-oriented systems, Lect Notes Computer Sci, Lect Notes Artificial Intell, Vol. 6882, pp. 591–600 (2011)

[37] Saatçi Y., Turner R., Rasmussen C.E., Gaussian Process Change Point Models, Proc. of the 27th International Conference on Machine Learning, pp. 927–934 (2010)

[38] Schwarz G., Estimating the dimension of a model, Annals Statistics, Vol. 6, No. 2, pp. 461–464 (1978)

[39] soapUI, http://www.soapui.org/

[40] Stoica P., Selen Y., Model-order selection: A review of information criterion rules, IEEE Signal Process Magazine, Vol. 21, Issue 4, pp. 36–47 (2004)

[41] Tari A., Elgedawy I., Dahmani A., A dual-layered model for web services representation and composition, J Intell Inf Syst, Vol. 32, pp. 237–265 (2009)

[42] Thottan M., Ji C., Anomaly Detection in IP Networks, IEEE Transactions On Signal Process, Vol. 51, No. 8, pp. 2191–2204 (2003)

[43] Tomczak J.M., On-line change detection for resource allocation in service-oriented systems, in: Technological innovation for value creation, eds. L.M. Camarinha-Matos, E. Shahamatnia, G. Nunes, IFIP AICT Vol. 372, Springer-Verlag, Heilderberg, pp. 51–58 (2012)

[44] Tomczak J.M., Świątek J., Brzostowski K., Bayesian classifiers with incremental learning for nonstationary datastreams, in: Advances in Systems Science, pp. 251–260 (2010)

[45] Tomczak J.M., Świątek J., Latawiec K., Gaussian process regression as a predictive model for Quality-of-Service in Web service systems, arXiv preprint arXiv:1207.6910 (2012)

[46] Welsh, M., Culler, D., Brewer, E., SEDA : An Architecture for Well-Conditioned , Scalable Internet Services, Proceedings of the eighteenth ACM symposium on Operating systems principles, pp. 230–243 (2001)