Context Change Detection for Resource Allocation in Service-Oriented Systems

Piotr Rygielski and Jakub M. Tomczak

Institute of Computer Science, Wrocław University of Technology,
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
{piotr.rygielski,jakub.tomczak}@pwr.wroc.pl

Abstract. In this paper, the problem of detecting the major changes in the stream of service requests is formulated. The change of stream component varies over time and depends on, e.g., a time of a day. The underlying cause of the change is called a context. Hence, at each moment there exists a probability distribution determining the probability of requesting the system service conditioned by the context. The aim is to find such a moment in which the distributions change. To solve that problem dissimilarity measures between two probability distributions are given. Nevertheless, detecting every change is not interesting but only long-lasting changes because of the costs of the service system resources reallocation. Therefore, in the proposed algorithm an issue of sensitivity to temporary changes detection is considered.

Keywords: datastream, dissimilarity measure, sliding window estimation.

1 Introduction

Recently, an increasing interest of service-oriented systems (SOA) can be noticed [6]. SOA consists of elements called *services* that are network-distributed on different computational nodes [8,9]. However, service-oriented architectures cause new problems connected with resource sharing and efficient resource management [10]. In the real-life situations in which streams of requests are time-varying and dependent on circumstances connected with e.g. time of a day, or changing number of users of the system, resources given to services have to be managed in an efficient way. It means that resources should be allocated in such a way that all operations and consequently – services are accomplished successfully in an appropriate time. However, due to the changes of the requests' streams the resources have to be reallocated [12] during the system life. Nevertheless, reallocation process consumes an additional time and operational resources and that is why it cannot be carried out at any change of the stream of requests. Thus, we deal with a tradeoff between rare reallocations but worst resource usage and often reallocation but higher costs associated with the resource management.

In this work we address the problem of the context change detection in service-oriented systems. We state the problem and present an algorithm for change detection in streams of requests. The main idea is to apply a dissimilarity measure

A. König et al. (Eds.): KES 2011, Part II, LNAI 6882, pp. 591–600, 2011.

between two probability distributions. However, the key challenge is to detect only long-lasting changes and no temporary changes because we would like to avoid too often resource reallocation operations. Therefore, we provide a detailed description of the change detection algorithm and present results of the simulation study.

2 Related Works

In the literature there are different methods for detecting changes. They could be divided into heuristics and theoretical approaches. The first group includes methods that are based on classifiers [15,21]. If the classification accuracy of the classifier drops significantly a context change is reported. It was used e.g. in the FLORA-family of algorithms [21]. Moreover, the weighted forgetting and window size management methods [11,15] could be included to this group as well.

The second group uses the theoretical aspects, especially taken from statistics. These approaches compare two probability distributions to detect a change. However, they differ in assumptions. For example in computational theory-based approaches [4,15] it is assumed that changes are rather permanent and gradual. Then an upper bound of the shift could be given and a number of recent observations for the processing may be calculated. Unfortunately, in practice sometimes we deal with abrupt changes and such methodology fails (e.g. in the anomaly detection [3,16]). Then a dissimilarity measure of two probability distributions to detect a change could be applied. In many applications the Kullback-Leibler divergence is used [3,16,19], or entropy-based measures [19,20]. Sometimes more theoretical measures are proposed [14].

3 Problem Statement

3.1 Problem Background

The service-oriented systems are systems that consist of network distributed components called *services* [8]. The single service is assumed to deliver atomic functionality and has its inputs and outputs precisely defined. We distinguish an atomic and complex service which delivers single functionality and the functionality composed of more than one functionality, respectively. Moreover, the service-oriented system is able to compose the complex service in order to deliver the service demanded by the user [8].

An atomic service is assumed to consume computational resources. Moreover, we assume that each atomic service is located within a computational node which is able to manage its resources using widely known virtualization techniques [18]. In this paper we consider the resource management task on a single computational node with atomic services installed on it.

Furthermore, there is a stream of service requests arriving to the service-oriented system. A request demands certain functionality. In order to compose

required service a composition procedure is executed which has been described in [8]. The most important result of composition is fact that proper atomic services are executed according to an execution scenario.

In order to optimize the performance of each atomic service the computational resources should be managed properly. The ratio of the available amount of the resource should be assigned to the proper virtual machine depending on the intensity of the requests stream and the complexity of the atomic service itself. The way how the complexity of the service is measured is not a concern of this paper and will be considered in future works. Denote the intensity of request stream arriving to the computational node at the moment t as $\mathbf{x}_t$ where $\mathbf{x}_t$ consists of substreams $\mathbf{x}_t = (x_t^{(1)}\ x_t^{(2)} \dots x_t^{(N)})^T$, each $x_t^{(n)}$ is dedicated to the proper atomic service installed on the considered computational node, and n denotes a number of the service, $n = 1, 2, \dots, N$.

We assume that intensities of the request stream components $(x_t^{(1)} \dots x_t^{(N)})^T$ can vary over time e.g. depending on the time of a day. Therefore, the resources should be periodically reallocated in order to maximize the quality of service delivered to the system users.

In order to determine allocation different algorithms could be applied e.g. well-known optimization algorithm called *interior-point* method [2]. Unfortunately, each operation of resource reallocation introduces a slight delay in computational node response time [12]. Thus, in order not to deteriorate the quality resource reallocation should not be performed too often.

3.2 Context Change Detection Problem Statement

In this work we assume that the stream of requests could vary in time because of a changing *context* [11,21]. By the context we understand external, unobservable circumstances e.g. a time of a day, or an increasing number of users, that affect the intensity of requested services. Formally, let us introduce a random variable that is time-depndent and that describes the requested service, r_t. It takes values in $\{1, 2, \dots, n \dots, N\}$ where n denotes the n^{th} service that was requested.

Moreover, the context is also a stochastic process, c_t, but we assume that it is a latent variable that is constant for some periods of time (see Fig. 1.b). However, the context affects the probability of requested services and thus the probability of the n^{th} requested service is denoted by $p(r_t|c_t)$. In real-life situations context changes in a *non-stationary* way [7,11]. Therefore, we deal with the non-stationary stochastic processes r_t, and c_t.

Besides, it could be said that we consider a *Hidden Markov Model* [1] (see Fig. 1.a) in which the context evolves in an abrupt way (see Fig. 1.b). We assume that we are able to observe only the service that was requested and there is no information about the context. We can only try to determine moments of change of the probability distribution, $\tau = \{t : p(r_t|c_t) \neq p(r_{t+1}|c_{t+1})\}$.

However, for our purpose, we are not interested in finding *any* change. For example detecting temporary changes leads to frequent reallocation of resources which implies high costs connected with that process. Therefore, the change should be reported only if estimated probability distributions differ significantly.

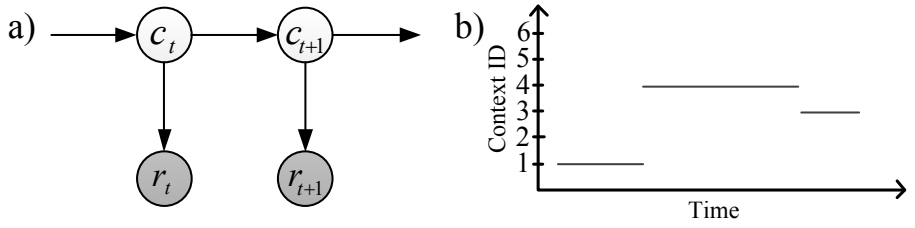


Fig. 1. (a) A probabilistic graphical model for the requesting n^{th} atomic service conditioned by a context. (b) An abrupt change of a context.

The change is measured by a certain dissimilarity measure $\mathcal{D}$. Then the change is reported if the dissimilarity value is higher than a *sensitivity factor* $\sigma \in [0, 1]$. Hence, the problem can be stated as follows:

Given:

- probability distribution at any time t, $p(r_t|c_t)$;
- the dissimilarity measure $\mathcal{D}$;
- the value of the sensitivity factor σ.

Find:

- The moments such that:

$$\tau = \Big\{ t : \mathcal{D}\big(p(r_{t-1}|c_{t-1}), p(r_t|c_t)\big) \geq \sigma \Big\}.$$

In the next section we present an algorithm for context change detection. However, to propose the algorithm two issues have to be solved. First, a dissimilarity measure has to be defined. Second, at any time we do not have probability distributions. Thus, a proper estimation technique adequate for stream of data has to be proposed.

4 Context Change Detection Algorithm

4.1 Dissimilarity Measures

In the literature a variety of dissimilarity measures are known e.g. Kullback-Leibler divergence [16], Lin-Wong divergence (modified Kullback-Leibler) [17], Bhattacharrya measure [13], Kolmogorov measure [5], entropy-based measure [20], cosine distance measure [19], and others (more measures could be found in [5]). Obviously, they differ in forms and mathematical properties.

In further considerations let assume that there is a discrete random variable $x \in \mathcal{X}$, $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$, and two probability distributions: $p(x)$, called *reference* distribution, and $q(x)$, called *model* distribution.

In this paper, based on our preliminary research and because of the discrete stochastic process r_t, we decided on four measures: *Bhattacharyya* measure, *Lin-Wong* divergence, *modified Lin-Wong* divergence, and *Kullback-Leibler* divergence. That is:

The Bhattacharyya measure is defined as follows [13]:

$$\mathcal{D}_B(p,q) = -\ln\Big(\sum_{n=1}^{N}\sqrt{p(x_n)\cdot q(x_n)}\Big). \tag{1}$$

The Lin-Wong divergence is defined as follows [17]:

$$\mathcal{D}_{LW}(p,q) = \sum_{n=1}^{N} p(x_n)\cdot\ln\Big(\frac{p(x_n)}{0.5\cdot p(x_n)+0.5\cdot q(x_n)}\Big). \tag{2}$$

The Kullback-Leibler divergence is defined as follows [16]:

$$\mathcal{D}_{KL}(p,q) = \sum_{n=1}^{N} p(x_n)\cdot\ln\Big(\frac{p(x_n)}{q(x_n)}\Big). \tag{3}$$

The modified Lin-Wong divergence uses the fact that for discrete probability distributions $\mathcal{D}_{LW}(p,q)\in[0,1]$. Then the dissimilarity measure could be defined as follows:

$$\mathcal{D}_{LW2}(p,q) = \Big(\mathcal{D}_{LW}(p,q)\Big)^2. \tag{4}$$

Remark. In the preliminary research it turned out that for the estimation technique used in this paper (see next section) the measure (4) is less sensitive to underestimations. It is easy to explain and it follows from the property of quadratic function and that $\mathcal{D}_{LW}(p,q)\in[0,1]$. The original function $\mathcal{D}_{LW}(p,q)$ is *flattened*, especially near zero, which protects from underestimations.

4.2 Probabilities Estimation Technique

To estimate the reference and model distributions the sliding window technique was applied [14,20]. The sliding window technique uses two windows, one representing L_1 of older and the other representing L_2 of recent observations (light and dark gray areas in Fig. 2) in the stream. The older instances (the older window) are chosen for estimating the reference distribution and the current data (the current window) – for the model distribution.

For the further simplicity, we set $L_1 = L_2 \equiv L$.

4.3 Detection Algorithm

The idea of the algorithm is very similar to the meta-algorithm presented in [14] and the method in [3]. First, data from datastream is taken and divided into

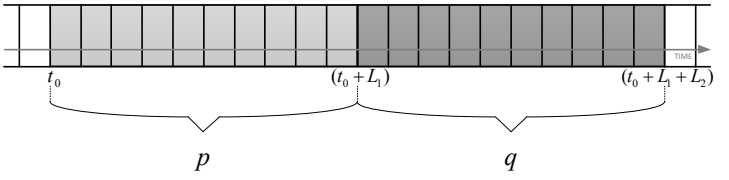


Fig. 2. An illustration of sliding windows for probabilities estimation

two sliding windows. Then, the reference and model distributions are estimated. Later, a value of the dissimilarity measure is calculated. If the value is lower than the given sensitive factor, sliding windows are moved so that all new observations are included. Otherwise the change is reported and sliding windows are set at the same moment and only the window with new observations is moved as long as it does not overlap instances from the older window. After that both windows are moved next to each other. The flow chart of the algorithm is presented in the Fig. 3.

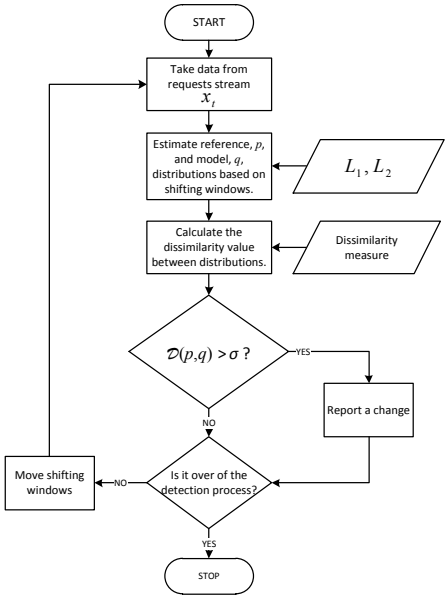


Fig. 3. The flow chart of the change detection algorithm

5 Experimental Study

5.1 Experiment Description

In order to verify the efficiency of the context change detection algorithm the simulation environment has been developed. The OMNeT++ as the simulation

platform was used. We modeled the physical machine with proper amount of virtual machines — each machine was responsible for responding to user's requests. The request stream was generated in such a way that there were three substreams present — each generated by a varying number of users. Each user is modelled as a process that repeatedly formulates a request sized from 1 to 4 kilobytes and after formulating the request the process is idle for a random period of time. One substream was assumed to be a background traffic and had almost constant intensity with small fluctuations during the simulation. The aggregated stream of requests was analyzed and then decomposed and passed to proper virtual machines for execution. Each virtual machine was assigned with initial amount of computational resources. When the context had changed, the resource allocation algorithm was executed.

In the experiment the measures presented in the Section 4.1 were used with following values of sensitivity factor (fixed after several trials): Bhattacharyya measure – $\sigma = 0.01$, Lin-Wong measure – $\sigma = 0.01$, modified Lin-Wong measure – $\sigma = 0.0001$, Kullback-Leibler measure – $\sigma = 0.015$. The sensitivity of modified Lin-Wong measure has to be much more smaller than for Lin-Wong because its value is squared. Besides, because of high sensitivity to even small changes, the value of the sensitivity factor for Kullback-Leibler has to be higher.

The size of the window was set to $L = 1second$ so that the estimation of probabilities was ensured. Moreover, a change is assumed to be correctly detected if the detection took place no later than 1 second from the actual change (no longer than the size of the current window). Otherwise it was not counted as a proper detection. To compare the algorithm with the different dissimilarity measures, following criteria were defined:

– A percentage of properly detected changes:

$$Q_g = \frac{\#\text{ of properly detected changes}}{\#\text{ of real changes}}$$

– A percentage of wrongly detected changes among all detected changes:

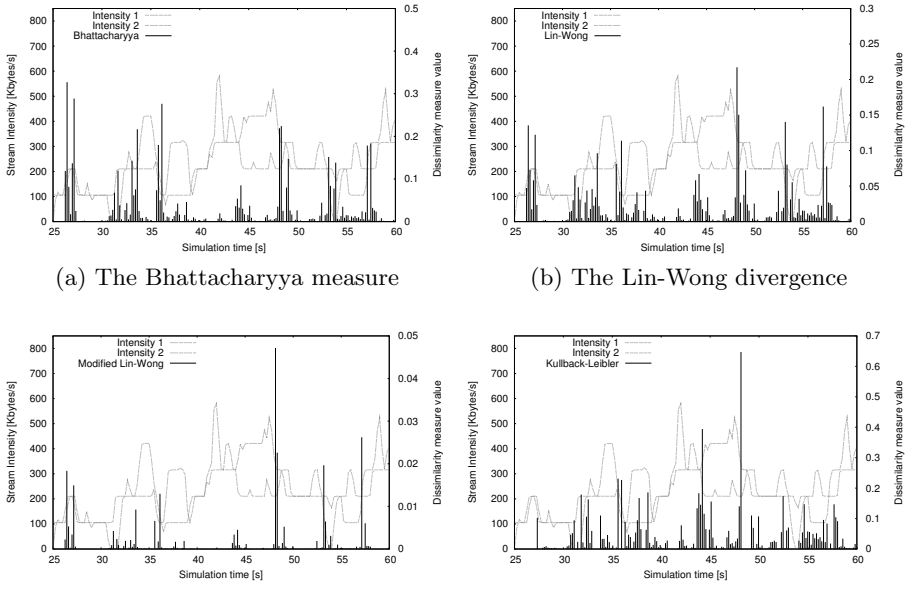
$$Q_b = \frac{\#\text{ of wrongly detected changes}}{\#\text{ of all detected changes}}$$

5.2 Results and Discussion

The simulator was run 10 times with about 25 changes of the context. Each simulation run lasted 100 seconds what was enough to process about 10 Mbytes of requests data. Results are shown in the Table 1. The best results are bold font.

Moreover, single exemplary run of the simulation environment for different dissimilarity measures are presented in Fig.s 4a, 4b, 4c, and 4d. Grey lines represent varying intensities of two classes (third class was fixed to be constant), and black bars – values of the measures.

The results show that Bhattacharyya, Lin-Wong, and modified Lin-Wong measures performed very similarly. However, they outperformed Kullback-Leibler



(a) The Bhattacharyya measure

(b) The Lin-Wong divergence

(c) Modified Lin-Wong divergence

(d) Kullback-Leibler divergence

Fig. 4. Exemplary results for four measures in a chosen simulation period

Table 1. Results for the different dissimilarity measures and considered criteria Q_g and Q_b

	Q_g				Q_b			
Batch No.	$\mathcal{D}_B$	$\mathcal{D}_{LW}$	$\mathcal{D}_{LW2}$	$\mathcal{D}_{KL}$	$\mathcal{D}_B$	$\mathcal{D}_{LW}$	$\mathcal{D}_{LW2}$	$\mathcal{D}_{KL}$
#1 (25 changes)	1	1	0.96	0.64	0.14	0.17	0.17	0.44
#2 (24 changes)	0.96	0.92	0.96	0.75	0.18	0.29	0.26	0.5
#3 (25 changes)	0.96	0.92	0.92	0.68	0.2	0.23	0.18	0.54
#4 (25 changes)	1	0.97	1	0.71	0.06	0.12	0.09	0.41
#5 (27 changes)	0.89	0.93	0.93	0.78	0.2	0.22	0.19	0.46
#6 (24 changes)	0.96	1	0.96	0.67	0.21	0.2	0.21	0.53
#7 (27 changes)	1	0.96	0.96	0.67	0.13	0.13	0.13	0.54
#8 (29 changes)	0.89	0.93	0.97	0.69	0.04	0.04	0.03	0.43
#9 (26 changes)	0.96	0.96	1	0.65	0.04	0.07	0.07	0.54
#10 (29 changes)	0.93	0.93	0.93	0.66	0.04	0.07	0.04	0.42
mean:	**0.96**	0.95	**0.96**	0.69	**0.12**	0.15	0.14	0.48
std dev:	0.04	0.03	0.03	0.045	0.07	0.08	0.08	0.05
worst case:	0.89	**0.92**	**0.92**	0.64	**0.21**	0.29	0.26	0.54

divergence. Moreover, it could be said that Bhattacharyya measure and modified Lin-Wong measure are slightly better than Lin-Wong measure. Because of lack of space no statistical proof is given here. However, after applying t-Student test it turned out that mentioned remark holds true.

Besides, the results of the KL measure seem to be quite odd and puzzling. After analyzing the raw results it turned out that KL divergence detected almost all of real changes but very often too late. Nevertheless, very weak performance according to the *bad* criterion follows from the definition of that measure. Even very small changes in probability distributions affect in quite big values of the KL divergence. Therefore, it is very sensitive to underestimations.

Furthermore, it can be stated that the proposed modification of the Lin-Wong measure is very interesting and promising. Because of the *flattening* less bad detections, in comparison to Lin-Wong measure, were made. The effect of the *flattening* could be easily seen in Fig.s 4b and 4c. However, more theoretical and experimental proofs are needed.

6 Final Remarks

In this paper a problem of the context change detection was stated and the algorithm for detection was presented. Moreover, four dissimilarity measures were given including one proposed for the first time. At the end the experimental study was carried out. The simulation environment implemented in OMNeT++ was briefly described and conclusions were drawn.

The outcomes of this work are planned to be used in a complex algorithm for decision making about resources management in the service-oriented systems. However, the problem of the context change detection is broader and the algorithm given here could be easily implemented in other domains e.g. learning algorithms. Nevertheless, our goal i.e. implementation of presented approach in the real-life service-oriented system is the priority of our future works.

Acknowledgments. The research presented in this paper has been partially supported by the European Union within the European Regional Development Fund program no. POIG.01.03.01-00-008/08.

The research has been also partially co-financed by European Union within European Social Fund.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Singapore (2006)
2. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York (2009)
3. D'Alconzo, A., Coluccia, A., Ricciato, F., Romirer-Maierhofer, P.: A Distribution-Based Approach to Anomaly Detection and Application to 3G Mobile Traffic. In: IEEE Global Telecommunications Conference, Honolulu, pp. 1–8 (2009)
4. Dries, A., Rückert, U.: Adaptive Concept Drift Detection. Statistical Analy Data Mining 2(5-6), 311–327 (2009)
5. Dragomir, S.S., Sunde, J., Buse, C.: New Inequalities for Jeffreys Divergence Measure. Journal of Mathematical Sciences 16(2), 295–309 (2000)

6. European Commission: From Grids to Service-Oriented Knowledge Utilities. A critical infrastructure for business and the citizen in the knowledge society (2006), ftp://ftp.cordis.europa.eu/pub/ist/docs/grids/soku-brochure_en.pdf
7. Grzech, A.: Teletraffic Control in Teleinformatics Networks. Oficyna Wydawnicza PWr, Wrocław (2002) (in Polish)
8. Grzech, A., Świątek, P., Rygielski, P.: Translations of Service Level Agreement in Systems Based on Service Oriented Architectures. Cybernetics and Systems 41(8), 610–627 (2010)
9. Grzech, A., Świątek, P.: Modeling and optimization of complex services in service-based systems. Cybernetics and Systems 40(8), 706–723 (2009)
10. Grzech, A., Świątek, P.: Parallel processing of connection streams in nodes of packet-switched computer communication networks. Cybernetics and Systems 39(2), 155–170 (2008)
11. Harries, M.B., Sammut, C., Horn, K.: Extracting Hidden Context. Machine Learning 32, 101–126 (1998)
12. Huu, T.T., Montagnat, J.: Virtual Resources Allocation for Workflow-Based Applications Distribution on a Cloud Infrastructure. In: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 612–617 (2010)
13. Kailath, T.: The divergence and Bhattacharyya distance measures in signal selection. IEEE Trans. on Comm. Tech. 15(1), 52–60 (1967)
14. Kifer, D., Ben-David, S., Gehrke, J.: Detecting Change in Data Streams. In: Proceedings of the 30th VLDB Conference, Toronto, Canada, pp. 180–191 (2004)
15. Klinkenberg, R.: Predicting Phases in Business Cycles Under Concept Drift. In: Proc. of Tagungsband der GI-Workshop-Woche, pp. 3–10 (2003)
16. Lee, W., Xiang, D.: Information-theoretic measures for anomaly detection. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 130–143 (2001)
17. Lin, J.: Divergence Measures Based on the Shannon Entropy. IEEE Transactions on Information Theory 37(1), 145–151 (1991)
18. Rose, R.: Survey of system virtualization techniques, Technical report (2004), http://citeseer.ist.psu.edu/720518.html
19. Sebastião, R., Gama, J.: Change Detection in Learning Histograms from Data Streams. In: Neves, J., Santos, M.F., Machado, J.M. (eds.) EPIA 2007. LNCS (LNAI), vol. 4874, pp. 112–123. Springer, Heidelberg (2007)
20. Vorburger, P., Bernstein, A.: Entropy-based Concept Shift Detection. In: Proceedings of the Sixth International Conference on Data Mining, pp. 1113–1118 (2006)
21. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23(1), 69–101 (1996)