On-Line Change Detection for Resource Allocation in Service-Oriented Systems

Jakub M. Tomczak[1]

[1] Institute of Computer Science, Wrocław University of Technology,
Wybrzeże Wyspiańskiego 27,
50-370 Wrocław, Poland
Jakub.Tomczak@pwr.wroc.pl

Abstract. In this paper, an on-line change detection algorithm for resource allocation in service-oriented systems is presented. The change detection is made basing on a dissimilarity measure between two estimated probability distributions. In our approach we take advantage of the fact that streams of requests in service-oriented systems can be modeled by non-homogenous Poisson processes. Thus, for Bhattacharyya distance measure and Kullback-Leibler divergence analytical expressions can be given. At the end of the paper a simulation study is presented. The aim of the simulation is to demonstrate an effect of applying adaptive approach in resource allocation problem.

Keywords: change detection, Bhattacharyya distance, Kullback-Leibler divergence, Poisson process

1 Introduction

In the past few years, developing information and communication technologies (ICT) enable entrepreneurs to develop monolithic architectures into distributed ones. Hence Service Oriented Architecture (SOA) becomes crucial paradigm in designing service-oriented systems (SoS) [9].

In SoS the key element is a *service* which provides certain and well-defined functionalities and is characterized by parameters describing quality of required and delivered service [11]. Furthermore, services may be instantiated and assembled dynamically that leads to changing structure, behavior and location of software application at run-time [9]. However, to ensure high *quality of service* (QoS) a resource allocation problem needs to be solved [14]. The QoS depends on several factors but mainly on a stream of service requests which is time-varying. Many control algorithms, e.g., PID controller, can handle small fluctuations of the stream. However, problems arise in the presence of so called *abrupt* changes at unknown time instants [2], [12]. By abrupt changes, we mean changes in characteristics that occur very fast with respect to the sampling period of the measurements, if not instantaneously. The time immediacy refers to a moment at which properties of the stream suddenly change but before and after which properties are more or less

constant in some sense, e.g. stationary. Hence, to allow adaptive QoS management, a change detection method can be proposed.

Our approach builds on the change detection using a dissimilarity measure framework [7], [16], [17], [18]. The dissimilarity measure compares two probability distributions and a change is reported if the dissimilarity value is greater than a given threshold. The change detection using the dissimilarity measure provides an elegant framework for detection of abrupt changes. Furthermore, in our considerations we take advantage of characteristics of streams of service requests. We propose to model them as point processes, i.e., non-homogenous Poisson processes. According to the application of Poisson distributions, the dissimilarity measure, e.g., Bhattacharyya measure or Kullback-Leibler divergence can be expressed in the analytical way.

Together, the change detection with an assumption of Poisson streams of service requests and the dissimilarity measure expressed in the analytical way yield a fast and robust abrupt change detection algorithm.

The paper is organized as follows. In section 2 the contribution to the value creation is outlined. Next, related works are given. In section 4 the problem is stated and in section 5 the algorithm is described. In section 6 the simulation study is carried out. At the end conclusions are drawn and future works are proposed.

2 Contribution to the Value Creation

Generally speaking, service-oriented systems aim to increase satisfaction of usage for clients and service providers as well. Moreover, the quality of service becomes the crucial value in SoS. Therefore, there is constant need to improve methods and algorithms that sustain high QoS. This work contributes to improvement and creation of value, i.e., QoS, in service-oriented systems.

3 Related Work

In this paper, we do not focus on resource allocation methods and consider only change detection methods. In the literature, there are used two main approaches to solve the change detection problem [2], [5], [8], [12], namely, statistical methods, and machine learning-based methods. The first approach divides into the following:

- Parametric methods – a signal is split into time windows and each time window is described by a parameterized probability distribution (pd). Then parameters are estimated and two consecutive time windows are compared via a likelihood ratio [2], [12] or using dissimilarity measure between pds, such as Kullback-Leibler divergence [7] or entropy [18].
- Non-parametric methods – pd is estimated by using a non-parametric estimator, such as Parzen window estimator or histograms [16], [17]. Then typically pds are compared via a dissimilarity measure, e.g., Kullback-Leibler [17].

The second approach applies machine learning algorithms with supervised and unsupervised learning, such as clustering-based [5] and classifier-based methods [1], [5], [8]. In general the idea is to support a learning algorithm with observations that

are labeled and a change is reported if upcoming observations are classified to a new class. On the other hand, in the case of unsupervised learning, observations are grouped and a change occurs when observations belongs to other cluster than before.

Change detection algorithms are widely applicable in intrusion detection systems [5], quality change detection [2], anomaly detection in network traffic [7], change detection in streams of requests [16], and many others [5].

4 Problem Statement

4.1 Resource Allocation Problem

Let us assume that the total computational amount of resources equals U (for further simplicity $U = 1$) and there are S complex services within one computational node. A vector of resources allocation is denoted by $\mathbf{u}=[u^1\ u^2\ \dots\ u^S]^T$, $0 < u^s < 1$ for all $s = 1,2,\dots,S$. Furthermore, with each service a stream of requests is associated that at the n^{th} moment expresses the number of service requests to the s^{th} service, $r_n^s \in \{0,1,...\}$. In the literature about modeling network traffic it is said that a stream of data during user sessions is characterized with self-similarity [6], [15]. However, in the SoS, the stream of upcoming service requests can be successfully modeled with non-homogenous Poisson processes [13], [15] like in the typical telecommunication systems [10], $r_n^s \sim Poisson(\lambda_n^s, \tau, k) = \exp(-\lambda_n^s \tau) \cdot (\lambda_n^s \tau)/k!$, where λ_n^s is the intensity of the stream in the n^{th} moment, $\tau > 0$ is the length of the time interval, $k \geq 0$ is the number of requests in the time interval $[n, n+\tau]$.

For further simplicity of this paper it is assumed that only a *service time* for each service as a QoS is taken into account. Hence, the QoS is expressed as follows

$$q_n^s(r_n^s; a^s, u^s) = \frac{a^s}{u^s} \cdot r_n^s, \tag{1}$$

for all $s = 1,2,\dots,S$, where a^s denotes an average service time of 1 request over 1 unit of time. This simplistic model depends inversely on resource which means that if less resources are given to the service, then the total service execution time is longer.

Hence, the total quality of the system at n^{th} time step can be considered as a sum of all total service times, i.e. , $Q_n(\mathbf{u}) = \sum_{s=1}^{S} q_n^s(r_n^s; a^s, u^s)$.

Moreover, the expected value over streams of requests can be considered, i.e.

$$\mathbf{E}\left[Q_n(\mathbf{u})\right] \overset{def}{=} \overline{Q}_n(\mathbf{u}) = \sum_{s=1}^{S} q_n^s(\lambda_n^s; a^s, u^s). \tag{2}$$

The equation (2) is proper because of the linearity of (1). Furthermore, if we assume that the streams of requests are generated according to random processes with

means constant on some periods of time and which change abruptly (point processes), then the final resource allocation problem can be stated as follows:

$$\begin{aligned} &\text{minimize} && \overline{Q}_{m:n}(\mathbf{u}) \\ &\text{subject to} && 0 < u^s < 1,\ s = 1,2,\ldots,S, \\ & && \sum_{s=1}^{S} u^s \le 1. \end{aligned} \tag{3}$$

where lower index by mean total quality, m:n, expresses a period from the m^{th} time step to the n^{th} moment. After the change detection resources are re-allocated.

4.2 Change Detection Problem

Generally speaking, the problem of change detection points in determining moments of abrupt changes. Thus, we assume there are given S sequences of observations, $r_{1:N}^s = \{r_n^s\}_{n=1}^N$. Basing on observations the probability distributions are estimated and two following hypotheses are checked:

$$\begin{cases} H_0 : D(P_1, P_2) \le h & (No\ abrupt\ change) \\ H_1 : D(P_1, P_2) > h & (An\ abrupt\ change) \end{cases}, \tag{4}$$

where P_1 and P_2 are pds for two consecutive time windows, h is the threshold, $D : \wp \times \wp \to [0, +\infty)$ is the dissimilarity measure. Hence, a change is reported if hypothesis H_1 holds true so that we obtain the following sequence

$$d_n = \begin{cases} 0\ if\ H_0\ holds\ true\ in\ the\ n^{th}\ moment \\ 1\ if\ H_1\ holds\ true\ in\ the\ n^{th}\ moment \end{cases}. \tag{5}$$

In this work, three dissimilarity measures are considered, i.e., Bhattacharyya distance measure [4], Kullback-Leibler divergence [4], and an absolute mean difference [12] for which analytical formulae for Poisson distributions can be given:

$$D(Poisson_1, Poisson_2) = \begin{cases} \frac{1}{2}\tau\left(\sqrt{\lambda_1} - \sqrt{\lambda_2}\right)^2\ for\ Bhattacharyya\ measure \\ \tau\left(\lambda_2 - \lambda_1 + \lambda_1 \log\frac{\lambda_1}{\lambda_2}\right)\ for\ Kullback - Leibler \\ \tau|\lambda_1 - \lambda_2|\ for\ an\ absolute\ mean\ difference \end{cases} \tag{6}$$

Hence, the problem of the change detection is as follows: for estimated pds and given dissimilarity measure find the following set of moments

$$t = \{n : d_n = 1\}. \tag{7}$$

5 Change Detection Algorithm and Resource Re-Allocation

The proposed algorithm for change detection consists of three steps. First, for each service, means of Poisson processes of two consecutive time widows are estimated (lines 7-11 in Table 1). Second, for each service, the value of the dissimilarity measure between two processes is calculated (line 10 in Table 1). Finally the stopping criterion for each service is checked, i.e., if the value of dissimilarity measure is grater or not than given threshold *h* (line 13 in Table 1).

Additionally, resource allocation is initialized (lines 1-3 in Table 1) and in case of detected change the resources are re-allocated due to a given optimization algorithm (line 15 in Table 1), e.g., interior-point algorithm [3].

Table 1. Adaptive algorithm for resource re-allocation with change detection mechanism

```
   program Resource Re-Allocation

   Inputs:
   lambda - matrix (nxS) of means, r - matrix (nxS) of
requests,
   L - length of time window, h - threshold, S - number of
   services, D - vector (1xS) of dissimilarity, n - time step,
   lambda1 - matrix (nxS) of means for 1st shifting window,
   lambda2 - matrix (nxS) of means for 2nd shiftinf window,
   u - vector of allocations, t is empty

   Outputs:
   t - vector of moments of change,

   Procedure:
   Initialization of allocation:
   1. for s from 1 to S do
   2.  u(s):=1/S;
   3. end for
   4. n := 0;
   5. repeat
   6.   n:=n+1;
   7.   for s from 1 to S do
   8.     lambda1(s,n):=mean(r(s,max(1,n-2*L):max(1,n-L)));
   9.     lambda2(s,n):=mean(r(s,max(1,n-L+1):max(1,n)));
   10.    D(s):=Dissimilarity(lambda1(s,n),lambda2(s,n));
   11.  end for
   12.  for s from 1 to S
   13.    if D(s) > h
   14.      extand t by adding n at the end position
   15.      re-calculate u by using optimization algorithm
   16.      break;
   17.    end if
   18.  end for
   19. until new requests arrive
```

6 Simulation Study

6.1 Simulation Details

In order to verify the efficiency of the change detection algorithm a simulation environment implemented in Matlab® has been developed. We consider a scenario within one computational node (the procedure is the same for each computational node). To keep the example uncluttered only 3 services are examined. The simulation time was set to 1000 units. Moreover, to check if the proposed algorithm can handle abrupt changes the following changes in streams of service requests (modeled as Poisson processed) are considered: (1) for **service 1**: λ=4 from 1 to 200, λ=8 from 201 to 400, λ=12 from 401 to 800, λ=11 from 801 to 100; (2) **service 2**: λ=2 from 1 to 500, λ=4 from 501 to 100; (3) **service 3**: λ=2 from 1 to 300, λ=6 from 301 to 600, λ=3 from 601 to 1000. Hence, the first stream changes three times at 200, 400 and 800, the second one only once at 500, and the third one – two times at 300 and 600. There are 6 abrupt changes at moments 200, 300, 400, 500, 600, and 800.

After detecting a change, the resources are re-allocated due to *interior-point algorithm* [3]. In order to assess the adaptive approach, an approach with uniform resource allocation was applied ($u^s = 1/S$). The value quality criterion (2) for the uniform allocation is referred to as *reference mean quality*.

Thus, the change detection algorithm with different dissimilarity measure is compared via three indicators:

1. *Good* – the ratio of detected real changes.
2. *Bad* – the ratio of the number of reported wrong changes and the number of all detected changes.
3. *Difference* – the difference between mean quality after applying re-allocation and reference mean quality.

The length of time windows were L=50, 75, 100. The threshold values were chosen after several trials and were: h=0.02 for D_B, h=0.075 for D_{KL}, h=1.1 for D_M.

The simulation was run 1000 times, all results were averaged.

6.1 Results and Discussion

The results for considered dissimilarity measures with varying length of time windows L and for three considered indicators are gathered in Table 2.

Generally it can be said that applying dissimilarity measure for change detection is sufficient (from 0.87 to 0.96 of good detections, see Table 1). Nevertheless, for too small window length L there can be a lot of wrong detections (from 0.37 to 0.57, see Table 1). It is especially important because in real-life situations each application of resource re-allocation is associated with additional costs that affect the quality of the system.

However, the most important issue, which should be considered, is the difference in the mean quality between the mean quality after applying the re-allocation procedure (an adaptive approach) and a uniform resource allocation (so called *reference quality*). For assumed values of parameters **a**, the difference was from 20 to

30 units of time in favor of the adaptive approach. Of course, for other values of **a** this difference is smaller or bigger but the magnitude is not interesting. The important result is that the adaptive approach performed better than the one with the uniform resource allocation.

Table 2. Results for three chosen indicators and three dissimilarity measures with varying *L*.

	Bad		*Good*		*Difference*
	Mean	**Std**	**Mean**	**Std**	
D_B, L=50	0.55	0.08	0.93	0.09	28.45
D_B, L=75	0.28	0.14	0.88	0.10	29.33
D_B, L=100	0.12	0.14	0.87	0.11	30.43
D_{KL}, L=50	0.57	0.08	0.94	0.09	28.50
D_{KL}, L=75	0.32	0.14	0.89	0.11	29.33
D_{KL}, L=100	0.14	0.15	0.87	0.11	30.44
D_M, L=50	0.37	0.10	0.96	0.07	28.50
D_M, L=75	0.16	0.12	0.94	0.09	29.55
D_M, L=100	0.08	0.10	0.92	0.09	30.70

At the end it is worth noting that the length of the shifting windows has a huge influence on the performance of the change detection. If the length is chosen incorrectly (too small or too high), then the pd is underestimated or overestimated. Thus, some changes remain undetected or can be reported even if there are no changes. However, in the real-life situations when each re-allocation costs extra it may be better to miss some changes then to report them too often.

Last but not least, to maintain the robustness with respect to too often change detection, both the length of the window and the threshold should be tuned appropriately.

7 Conclusions and Further Work

In this paper, the on-line change detection algorithm based on the dissimilarity measure between two parameterized pds was outlined. Its performance was checked in the simulation environment with linear quality function. It was assumed that the streams of requests were modeled by non-homogenous Poisson processes which is acceptable assumption for streams in service-oriented systems [13], [15].

In the future following aspects should be considered:

- more sophisticated quality functions to assess QoS [11], [14];
- conducting real-life experiment;
- including noise in the streams of requests (omitted in this paper because of lack of space);
- proposition of Bayesian framework for change detection.

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

References

1. Baena-Garcia, M., del Campo-Avila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno, R.: Early drift detection method, In Proceedings of ECML PKDD 2006 Workshop on Knowledge Discovery from Data Streams, Berlin, Germany (2006)
2. Basseville, M., Nikiforov, I.: Detection of Abrupt Changes: Theory and Application, Prentice-Hall (1993)
3. Boyd, S., Vandenberghe, L.: Convex Optimization, Cambridge University Press, New York (2009)
4. Cha, S.-H.: Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions, Int. J. of Math. Models and Methods in Applied Sciences, 1:4 (2007)
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly Detection: A Survey. ACM Computing Survey, 41:15:1–15:58, 2009
6. Crovella, M.E., Bestavros, A.: Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes, IEEE Trans. Netw., 5:6, pp. 835-846 (1997)
7. D'Alconzo, A., Coluccia, A., Ricciato, F., Romirer-Maierhofer P.: A Distribution-based Approach to Anomaly Detection and Application to 3G Mobile Traffic, IEEE Global Telecommunications Conference, pages 1–8 (2009)
8. Desobry, F., Davy, M., Doncarli, C.: An Online Kernel Change Detection Algorithm, IEEE Trans. Signal Processing, 53:8, pp. 2961-2974 (2005)
9. European Commission: From Grids to Service-Oriented Knowledge Utilities. A critical infrastructure for business and the citizen in the knowledge society, ftp://ftp.cordis.europa.eu/pub/ist/docs/grids/soku-brochure en.pdf (2006)
10. Gnedenko, B.V., Kovalenko, I.N.: Introduction to Queueing Theory, Birkhauser, Cambridge (1989)
11. Grzech A., Rygielski P., Świątek, P.: Translations of Service Level Agreement in Systems Based on Service-Oriented Architectures. Cybernetics and Systems, 41:8, pp. 610-627 (2010)
12. Gustafsson, F.: Adaptive Filtering and Change Detection, John Wiley & Sons, Chichester, UK (2001)
13. van der Mei, R.D., Hariharan, R., Reeser, P.K.: Web Server Performance Modelling, Telecommunication Systems, 16:3-4, pp. 316-378 (2001)
14. O'Brien, L., Merson, P., Bass, L.: Quality Attributes for Service-Oriented Architecture, Proc. of IEEE SDSOA'07, pp. 3-9 (2007)
15. Paxson, V., Floyd, S.: Wide Area Traffic: The Failure of Poisson, IEEE Trans. Netw., 3, pp. 226-244 (1995)
16. Rygielski, P., Tomczak, J.M.: Context Change Detection for Resource Allocation in Service-oriented Systems LNAI, (2011)
17. Sebastiao, R., Gama, J., Pereira Rodrigues, P., Bernardes, J.: Monitoring Incremental Histogram Distribution for Change Detection in Data Streams, LNCS, 5840, pp. 24-42 (2010)
18. Vorburger, P., Bernstein, A.: Entropy-based concept shift detection. In Proc. of the Sixth Int. Conf. on Data Mining, pp. 1113–1118 (2006)