Analytical methods of nonstationary processes modeling[⋆]

Sergei Zhmylev[0000−0001−8916−5474], Ilya Martynchuk[0000−0003−2409−9677], Valeriy Kireev[0000−0003−1619−0681], and Taufik Aliev[0000−0003−0278−688X]

ITMO University, Kronverkskiy pr., 49, lit. A, St. Petersburg, 197101, Russia
korg@cs.ifmo.ru, mt4.ilja@gmail.com, kk@cs.ifmo.ru, aliev@cs.ifmo.ru

Abstract. Due to complex structure and a huge scale, cloud systems design and development requires a preliminary workload estimation. A typical solution for this design step is a workload upper-boundary estimation based on a sum of maximal intensities per cloud application. This solution is not well suitable for nonstationary functioning cloud computing systems, due to such an estimation would result in lots of hardware resources mostly unused during the worktime. Any workload can be represented as a combination of a probability density function and certain parameters for modeling purposes. For any stationary processes, a shape of the probability density function and the parameters would have constant values. On the contrary, nonstationary processes are characterized by changes of the values in time. The objective of the work is to develop an analytical method for the nonstationary processes modeling and representation. Completing the objective, the authors proposed a linear-equation nonstationary processes representation form. Which allowed to carry out the estimation using standard mathematical transformations and operations. The proposed numerical method makes it possible to approximate periodic non-stationary distributions whose change in time is sinusoidal. The method is tested with different types of synthetic signals and in all cases demonstrates a high degree of compliance of the results with the original data.

Keywords: Nonstationarity · Mathematical modeling · Simulation · Cloud computing · Computational resources scaling.

1 Introduction

Currently, computing is widespread in cloud systems that solve a wide range of tasks with small user requests processing time [1]. Examples of such systems include various Internet services for converting image formats and media files, services for performing mathematical calculations and services for collaborative work with office documents. Due to the fact that cloud systems, as a rule, have a web interface, there is no need in specialized software to work with them,

⋆ Supported by ITMO University.

therefore the number of their users is constantly increasing as well as the total workload. Therefore, these systems require continuous operation and high performance to ensure the high quality of the services provided.

A cloud system is a set of computing systems, as a rule, located in one data center, serviced by one staff and under the jurisdiction of one managing organization [2]. The management organization provides the services of renting a part of the cloud system computing power using virtualization technologies automatically or in a manual manner. For instance, there could be a special software for company administrators whis is capable of provisioning new computing resources for any software. Thus, the tenant receives a certain number of virtual nodes with the same configuration on which the cloud application can be launched. The cloud application has a single entry point for user requests and it is horizontally scalable, so the structure of the cloud application is hidden from users [3]. Cloud applications in contradiction to desktop apllications are characterized with a high level of horizontal scalability. Therefore, if a tenant has insufficient computing capacity at some point in time, he can rent additional virtual nodes and launch additional instances of it's cloud application, increasing the overall performance of his cloud application.

2 Problem formulation

Due to complex structure and a huge scale, cloud systems design and development requires a preliminary workload estimation. A typical solution for this design step is a workload upper-boundary estimation based on a sum of maximal intensities per cloud application [4]. This solution is not well suitable for nonstationary functioning cloud computing systems, due to such an estimation would result in lots of hardware resources mostly unused during the worktime. Consider a cloud system with many cloud applications. An average workload of the applications in queries per second could be either constant or experience changes over the measurement time.

Any workload can be represented as a combination of a probability density function and certain parameters for modeling purposes [5]. For any stationary processes, a shape of the probability density function and the parameters would have constant values. On the contrary, nonstationary processes are characterized by changes of the values in time. Thus nonstationarity in any systems could be delimited in three classes: with constant probability distribution, with constant parameters (mean, variation, etc.), and total non-constant [6]. Each of those nonstationarity classes is typical for specific environmental conditions. Mostly, nonstationarity may occur due to natural workload behavior. There are number of nonstationarity reasons that could be converged in three classes: periodic, aperiodic (e.g. damped oscillations) and chaotic – without any visible structure or reproducibility. Figure 1 represents a workload of the biggest Internet data exchange point of the Russian Federation – MSK-IX. It's shown that a network traffic tends to be a periodic nonstationary process, due to number of Internet users at a specific point: much more active users during an evening.

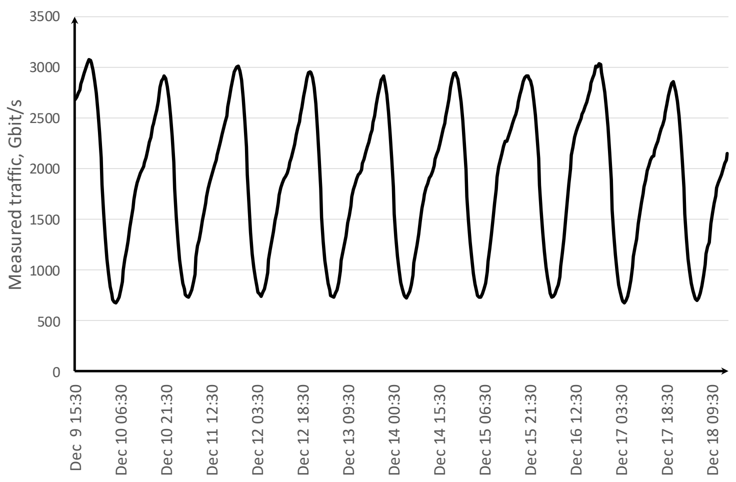

Fig. 1. MSK-IX weekly traffic

Absolutely the same workload class – periodic nonstationary process – was discovered in a course of numerous cloud systems studies. In spite this fact, typical approach of cloud systems design and real-time management consists of an estimation on maximum intensity values [7]. Such an approach leads to non-optimal workload distribution due to big amount of unused hardware resources. Therefore, the objective of the work is to develop an analytical method for the nonstationary processes modeling and representation.

3 Period estimation

For the non-stationary processes investigation during measurements on a real system, it is necessary to take not only the numerical characteristics of the distribution into account, which affect the absolute values of the measurands, but also the periodic component describing distribution characteristics changing with time. There is a need to automatically estimate the period length based on a sample of measured values in order to present the observed process in an analytical form. Below we propose a method for such processes period length estimating to ensure their analytical description and research.

Existing methods for such problems solving are usually associated with the sound signals analysis or have a different, narrow field of application, which makes it difficult to use them in network computing systems [8]. In addition, currently used methods are not suitable for the investigation of non-stationary processes occurring in cloud computing systems, since the function describing changes in such processes does not always have a sinusoidal form. Methods such as, for example, the fast Fourier transform (FFT) are not suitable for the period estimation of signals with a missing fundamental [9]. In particular, the fast

Fourier transform has a fairly good algorithmic complexity, but it requires a large number of floating point operations, including the results post-processing, which eliminates the algorithmic complexity advantage over the method described below. Also, despite the fact that the fast Fourier transform works well on signals having a natural nature, its applicability is not always relevant for the "artificial" signals observed in computer science [10]. In particular, the number of terms required for the FFT calculating in gap functions processing rushes to infinity, which makes the calculation impossible [11]. Therefore, it became necessary to develop a method for the period T length estimating, satisfying the following requirements.

1. The method should evaluate the period on an incomplete set of input data. Since the load created by users changes over time [12], to manage the cloud system, it is necessary to estimate the length of the requests arrival process period in "real time".
2. The method must have a polynomial or constant algorithmic complexity. It is clear that during long-term high frequency measurements, a sufficiently large number of measured values will be obtained, which will lead to a large processing time using existing algorithms with the exponential complexity [13].
3. The main memory usage should linearly dependent on the amount of input data, since it is assumed to use the developed method to manage the cloud system with a limited memory capacity [14].
4. The method should be resistant to measurement errors and obtain reliable results (in real environment as well) characterized by a non-ideal input signal form [15].
5. Calculation results should not depend on the quality of a periodic input data, due to real systems discreteness and therefore function holes presence.

Let there be some periodic value Y, depending on time $Y = y(t)$. Obviously, there exists a moment of time t_i for which the interval $(0; t_i)$ is an integer number of the $y(t)$ function periods. However, in practice, measurements of the Y values can be made with a variable time step, often skipping i-th samples. This makes it impossible to iteratively pass through all points in time at an equal step to analyze the $y(t)$ function behavior, so it becomes necessary to obtain the missing values of Y, that is, to move from the Y values to the Y' values pending at regular time intervals.

A linear interpolation could be used for $y'(t)$ missing values estimation. Having computational complexity of $O(1)$, it provides more precise period estimation [16].

Since the period T includes an integer number of intervals Δt (provided that $T > \Delta t$ and $T \, mod \, \Delta t = 0$), at time t_i the $y'(t_i) = y'(t_i - T)$ equality is true. Thus, two arbitrary adjacent equal intervals in which the $y'(t)$ function will have the same behavior can be found to search for a period in the sample.

In the proposed method, an iterative passage through the normalized sample of a given value is used, starting from the moment of t_0, that is, the moment of its first measurement. At every second moment of time t_i the interval $t = t_i - t_0$

is divided into two equal segments $t_0 \dots t_{i/2}$ and $t_{i/2} \dots t_i$ respectively. If the $y'(t)$ behaves identically on the obtained segments, then the segment $t_{i/2}$ is the desired $y'(t)$ function period.

Considered many ways to assess the functions characters similarity. In the proposed method, as a measure of the $y'(t)$ function behavior similarity degree on time intervals $t_0 \dots t_{i/2}$ and $t_{i/2} \dots t_i$ Pearson correlation coefficient r is used, which is calculated according to the formula (1).

$$r = \frac{cov(t_0 \dots t_{i/2}, t_{i/2} \dots t_i)}{\sigma[t_0 \dots t_{i/2}] * \sigma[t_{i/2} \dots t_i]} \tag{1}$$

Here, the correlation moment $cov(t_0 \dots t_{i/2}, t_{i/2} \dots t_i)$ is defined as $cov(X, Y) = M[(X - M[X]) * (Y - M[Y])]$, where M is the mean value defined for the series of values analyzing task as the arithmetic average. The standard deviation σ is defined as $\sigma = \sqrt{M[X^2] - M[X]^2}$.

Due to the limitation on the method computational complexity, for calculating M and σ values, required for the r coefficient determination, it is proposed to also store $\sum_{j=0}^{N} y'(t_j)$ and $\sum_{j=0}^{N} y'(t_j)^2$ sums for each value of the $y'(t_i)$ function. This makes it possible to reduce the mean value determination time of the sample to the time of calculating the difference between two values and the quotient, and the operation computational complexity is $O(1)$, while the r coefficient calculating computational complexity decreases to $O(N)$.

Thus, for each second sample of the $y'(t)$ function, the similarity degree of the two functions describing both halves of a known values series is calculated, respectively. If the correlation r exceeds a certain threshold value R, the $t' = t_{i/2}$ value is assumed to be a multiple of the function period T. In this case, the t' value s placed in the table of the $y'(t)$ function expected periods. This table is a correspondence of a supposed period T'_i and the number of times n, which this period was recorded among the $y'(t)$ function values $y'(t)$ with a high degree of similarity. Therefore, it is necessary not only to estimate the $y'(t)$ function half-segments similarity degree of the half-segments of the function when processing the next incoming samples of the $y'(t)$ function, but also to evaluate the added samples similarity of the function with all multiple expected periods.

To increase the accuracy of the obtained results, the n value can be considered not the number of periods occurrences in the set of the function known values, but the calculated correlation coefficients sum. In this case, for $r \geqslant R$ the n value will increase by the $\Delta n = R \leqslant \Delta n \leqslant 1$ value. Such an approach does not decrease the proposed method algorithmic complexity, however, it will require slightly more floating-point calculations, which may be undesirable, for example, in embedded systems.

In the course of numerous experiments it was found that for ideal functions (sinusoidal, meander, sawtooth and triangular) the suitable value of R is $0,999$. On data received from Russia's largest traffic exchange point [17], high estimation accuracy is achieved at $R = 0,9$. In the general case, it is proposed to estimate R using preparatory simulation experiments or use $R = 0,8$.

4 Composition

To solve the problem of designing cloud systems [18] with non-stationary processes it is important to identify the properties of the nonstationary distributions composition. Let two non-stationary processes are given by non-stationary distributions (2) and (4).

$$\begin{cases} f(x,t) = a(\lambda_f(t), x) \\ \lambda_f(t) = c(t) \end{cases} \quad (2) \qquad \begin{cases} sum(x,t) = a(\lambda_f(t), x)\otimes \\ \quad \otimes\, b(\lambda_g(t), x) \\ \lambda_h(t) = c(t) + d(t) \end{cases} \quad (3)$$

$$\begin{cases} g(x,t) = b(\lambda_g(t), x) \\ \lambda_g(t) = d(t) \end{cases} \quad (4) \qquad \begin{cases} comp(x,t) = (a(\lambda_f(t), x)+ \\ \quad +\, b(\lambda_g(t), x))/2 \\ \lambda_h = c(t) + d(t) \end{cases} \quad (5)$$

To calculate the sum of these processes (3), it is assumed that the probability density function can be represented as a convolution of the original probabilty density functions [19], and to calculate their composition (5), the probability density function can be calculated as the half-sum.

To estimate an average and maximum queries intensity one can just sum the relevant values: $\overline{\lambda(t)} = \overline{c(t)} + \overline{d(t)}$ and $\lambda_{max}(t) = c_{max}(t) + d_{max}(t)$. All of those hypothesis have been successfully tested with a following model in AnyLogic Professional 7.0.1 simulation environment:

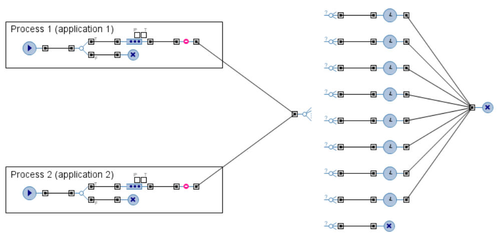

Fig. 2. AnyLogic model

The model (Figure 2) has the following parameters: $\lambda_i(t)$ – input streams intensity; b – application processing time; $A_1(\tau)$ and $A_2(\tau)$ – input stream probability density functions; $B(\tau)$ – application processing time probability distribution function; characteristic of the model: ρ – load average.

As a result of modeling the following property was identified: if the sum of average values of intensity does not exceed intensity of service [20], then, regardless of the sum of maximum values of intensities, the system will not pass to constantly overloaded state [21]. Thus, with sufficient storage capacities in such systems, mass losses do not occur. The revealed property allows to draw conclusions about the expediency of redistribution of non-stationary load in cloud systems to solve the problem of automatic scaling of the cloud [22]. However, the question of choosing suitable storage capacities remains open, so that the system functions correctly, without mass loss of applications [23].

Taking into account the confidence intervals with a confidence probability of 95% and the generally accepted error of simulation modeling of 5% [24], it can be concluded that at constant average intensities of incoming flows the total intensity is their sum.

5 Approximation

Let there be a series of non-stationary distributed data $X = (x_0, x_1, ..., x_N)$, the distribution intensity of which λ varies according to some time depinding periodic law: $\lambda(t)$, meanwhile the probability distribution function does not change.

Let $\lambda(t)$ be a periodic function, having the form of a sinusoid with a period T. Then the following function can be found (6), where A – the sin amplitude, ϕ_0 – the initial phase, C – the shift on the y-axis, which will most closely match the original $\lambda(t)$.

$$\lambda^{'}(t) = A * sin(\omega(t) + \phi_0) + C \tag{6}$$

At the same time, the Pearson correlation coefficient r can be chosen as a measure of compliance [24].

That is, the task of the X set approximation is reduced to the selection of such values A, ϕ_0, ω and C, for which $\lambda(t)$ and $\lambda^{'}(t)$ have the highest correlation coefficient r value.

At the first stage of the method, it is proposed to obtain a set of averaged values λ_i $(i \in (0..mT), m \in \mathbb{N})$ with a length of one or several $\lambda(t)$ function periods by averaging the measured X values at $i + kT$ points, where $k = (0, 1, 2, ..., \frac{N}{mT} - 1)$.

Since $\int_0^{2k\pi} (A * sin(t) + C)dt = 2k\pi C$, C is the $\lambda(t)$ function's average [25]. Thus, C can be obtained as $C = \frac{\sum_{i=0}^{N} \lambda_i}{N}$, given number of λ values N is multiple of the period [26].

The sine amplitude A can be calculated using the mean square of λ_i values: $A = \sqrt{\frac{2 * \sum_{i=0}^{N} \lambda_i^2}{N}}$. Or, given the value of C, $A = \sqrt{\frac{2 * \sum_{i=0}^{N} (\lambda_i - C)^2}{N}}$.

Having A and C values, the initial phase ϕ_0 could be estimated as follows, resulting in $\lambda^{'}(t)$ estimation. Consider ϕ_0 at point $t = 0$: $\lambda(0) = \lambda_0 = A * sin(\phi_0) + C$. There are only two potential values of ϕ_0: $\phi_{0(1)} = \arcsin(\frac{\lambda_0 - C}{A})$ and $\phi_{0(2)} = \pi - \arcsin(\frac{\lambda_0 - C}{A})$. To choose the correct one, Pearson correlation coefficient [27] could be used again, and the function with proper ϕ_0 value would have higher coefficient value.

The proposed numerical method makes it possible to approximate periodic non-stationary distributions whose change in time is sinusoidal [28] and represent it in a form of (6). The method is tested with different types of synthetic signals and in all cases demonstrates a high degree of compliance of the results with the original data. The results of experiments with confidence intervals are presented in the table below.

Table 1. Experiment results

Signal type	Correlation coefficient with exponential distribution	Correlation coefficient with Erlang distribution, k=5
Sine	0.890 ± 0.002	0.973 ± 0.002
Triangle	0.671 ± 0.006	0.915 ± 0.001
Saw	0.712 ± 0.002	0.765 ± 0.001
Meander (duty cycle 2)	0.68 ± 0.02	0.75 ± 0.02
Meander (duty cycle 7/10)	0.58 ± 0.03	0.71 ± 0.04

We can conclude that for the case of low coefficient of variation [28] ($\nu = \frac{1}{\sqrt{5}}$, the Erlang distribution with $k = 5$), the method as a result sets the function $\lambda^{'}(t)$, strongly correlating with the initial $\lambda(t)$ for the sine and triangular wave and, in general, acceptable results in all other cases.

This fact confirms the possibility of using the method in practice for systems having a similar workload.

6 Putting results into practice

The results obtained in the work can be applied in algorithms for automatic scaling of cloud computing systems. For example, let the cloud system have two nodes that run instances of the cloud application, shown on Figure 3.

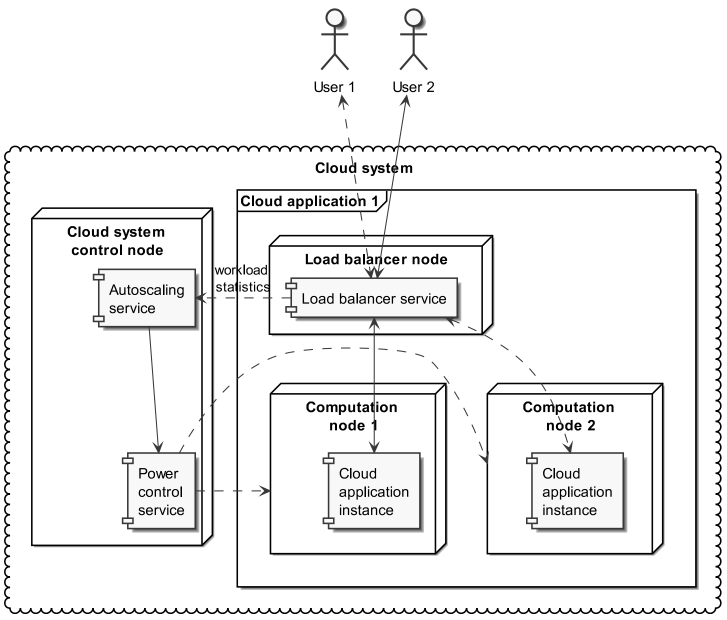

Fig. 3. Cloud system components and deployment

A workload from two users (User 1 and User 2) is distributed between the nodes, which is shown by solid and intermittent lines, respectively. Suppose each user creates a sinusoidal antiphase workload on the compute nodes. If we sum up the maximum values of the users workload intensities, we can make the erroneous conclusion that at least two nodes are needed to work simultaneously to ensure the required performance. At the same time, if we take into account the information about the nonstationarity of the load created by users and take advantage of the results obtained in the work, there will be no overload state with just one computing node. In this situation, the load balancer can send all user requests to one node, and the second node can be turned off to increase the energy efficiency of the cloud system.

Such an approach can significantly decrease energy consumption of large datacenters, in which cloud systems could use thousands and thousands of physical servers. Which is resulting in a huge economic cost reduce for the datacenter owners, who can farther reduce rent costs for end-users and increase their commerce income.

7 Conclusion

In this paper we proposed an analytical method for the nonstationary processes modeling and representation. This method includes the following: numerical parameterized method that allows to estimate a period length of non-stationary processes; a linear-equation nonstationary processes representation form, which allowed to carry out the estimation using standard mathematical transformations and operations; a numerical approximation method of periodic nonstationary distributions whose change in time is sinusoidal. This method helps to solve the problem of designing cloud systems with non-stationary processes.

References

1. Bogatyrev, Vladimir A., and A. V. Bogatyrev. "Functional reliability of a real-time redundant computational process in cluster architecture systems." Automatic Control and Computer Sciences 49.1 (2015): 46-56.
2. Park, K., Willinger, W. (2000). Self-similar network traffic and performance evaluation. John Wiley and Sons, Inc..
3. Bogatyrev, Vladimir A. "Fault tolerance of clusters configurations with direct connection of storage devices." Automatic Control and Computer Sciences 45.6 (2011): 330-337.
4. Aliev, T. I., M. I. Rebezova, and A. A. Russ. "Statistical methods for monitoring travel agencies in the settlement system." Automatic Control and Computer Sciences 49.6 (2015): 321-327.
5. Chhibber, Anju, and Sunil Batra. "Security analysis of cloud computing." International Journal of Advanced Research in Engineering and Applied Sciences 2.3 (2013): 2278-6252.
6. Mahon, Edward. Transitioning the Enterprise to the Cloud: A Business Approach. Cloudworks Publishing Company, 2015.

7. Kavis, Michael J. Architecting the cloud: design decisions for cloud computing service models (SaaS, PaaS, and IaaS). John Wiley and Sons, 2014.
8. Rafaels, Ray J. Cloud Computing: From Beginning to End. CreateSpace Independent Publishing Platform, 2015.
9. Buzzetto-More, Nicole A., ed. Advanced principles of effective e-learning. Informing Science, 2007.
10. Matthew Portnoy. Virtualization Essentials, 2nd Edition. New York: Wiley / Sybex, 2016
11. Jiang, Qiqi, Jianjun Qin, and Lele Kang. ”A literature review for open source software studies.” International Conference on HCI in Business. Springer, Cham, 2015.
12. Bogatyrev, Vladimir A., S. V. Bogatyrev, and I. Yu Golubev. ”Optimization and the process of task distribution between computer system clusters.” Automatic Control and Computer Sciences 46.3 (2012): 103-111.
13. V.A. Bogatyrev M.S. Vinokurova. ”Control and Safety of Operation of Duplicated Computer Systems” Communications in Computer and Information Science, vol. 700, pp. 331-342 2017.
14. Grossman, Robert L. ”The case for cloud computing.” IT professional 11.2 (2009): 23-27.
15. Lee, Gillam. ”Cloud Computing: Principles, Systems and Applications/Nick Antonopoulos, Lee Gillam.” L.: Springer (2010).
16. Park, Kihong, and Walter Willinger. Self-similar network traffic and performance evaluation. John Wiley and Sons, Inc., 2000.
17. Marston, Sean, et al. ”Cloud computing—The business perspective.” Decision support systems 51.1 (2011): 176-189.
18. Mao, Ming, and Marty Humphrey. ”A performance study on the vm startup time in the cloud.” 2012 IEEE Fifth International Conference on Cloud Computing. IEEE, 2012.
19. Boniface, Michael, et al. ”Platform-as-a-service architecture for real-time quality of service management in clouds.” Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on. IEEE, 2010.
20. Cover, Thomas M., and Joy A. Thomas. ”Elements of information theory 2nd edition.” Willey-Interscience: NJ (2006).
21. He, Sijin, et al. ”Elastic application container: A lightweight approach for cloud resource provisioning.” Advanced information networking and applications (aina), 2012 ieee 26th international conference on. IEEE, 2012.
22. Rashvand, Habib F., ed. Using cross-layer techniques for communication systems. IGI Global, 2012.
23. Gogouvitis, Spyridon, et al. ”Workflow management for soft real-time interactive applications in virtualized environments.” Future Generation Computer Systems 28.1 (2012): 193-209.
24. Gogouvitis, Spyridon, et al. ”Workflow management for soft real-time interactive applications in virtualized environments.” Future Generation Computer Systems 28.1 (2012): 193-209.
25. Gogouvitis, Spyridon, et al. ”Workflow management for soft real-time interactive applications in virtualized environments.” Future Generation Computer Systems 28.1 (2012): 193-209.
26. Gogouvitis, Spyridon, et al. ”Workflow management for soft real-time interactive applications in virtualized environments.” Future Generation Computer Systems 28.1 (2012): 193-209.

27. Gogouvitis, Spyridon, et al. "Workflow management for soft real-time interactive applications in virtualized environments." Future Generation Computer Systems 28.1 (2012): 193-209.
28. Bogatyrev, V. A. "Protocols for dynamic distribution of requests through a bus with variable logic ring for reception authority transfer." Automatic control and computer sciences 33.1 (1999): 57-63.