DOI: 10.18721/JCSTCS.13101
УДК 004.75

AN APPROACH FOR AUTOMATED DEPLOYMENT OF CLOUD APPLICATIONS IN THE EDGE-TO-CLOUD COMPUTING CONTINUUM SATISFYING HIGH QUALITY OF SERVICE REQUIREMENTS

P. Kochovski, P.D. Drobintsev

Peter the Great St. Petersburg Polytechnic University,
St. Petersburg, Russian Federation

Modern component-based software engineering environments allow deployment of cloud applications on various computing infrastructures, such as Edge-to-Cloud infrastructures. The heterogeneous nature of such computing resources results in variable Quality of Service (QoS). Therefore, the deployment decision can seriously affect the application's overall performance. This study presents an approach for automated deployment of cloud applications in the Edge-to-Cloud computing continuum that considers non-functional requirements (NFRs). In addition, the authors explore multiple methods for selection of optimal cloud infrastructure, such as IaaS. The paper presents an experimental evaluation performed using a cloud application for storing data under different workloads. For the purposes of the experimental evaluation, a Kubernetes cluster composed of 44 computing nodes was used. The cluster nodes were geographically distributed computing infrastructures hosted by several service providers. The proposed approach allows a reliable selection of infrastructures, which satisfy high QoS requirements for cloud applications, from heterogeneous Edge-to-Cloud computing environments.

Keywords: cloud computing, cloud application deployment, Quality of Service, Infrastructure as a Service, Edge-to-Cloud.

Citation: Kochovski P., Drobintsev P.D. An approach for automated deployment of cloud applications in the Edge-to-Cloud computing continuum satisfying high Quality of Service requirements. Computing, Telecommunications and Control, 2020, Vol. 13, No. 1, Pp. 8-18. DOI: 10.18721/JCSTCS.13101

ПОДХОД К АВТОМАТИЗИРОВАННОМУ РАЗВЕРТЫВАНИЮ ОБЛАЧНЫХ ПРИЛОЖЕНИЙ В ВЫЧИСЛИТЕЛЬНОМ КОНТИНУУМЕ EDGE-TO-CLOUD, УДОВЛЕТВОРЯЮЩИХ ВЫСОКИМ ТРЕБОВАНИЯМ К КАЧЕСТВУ ОБСЛУЖИВАНИЯ

П. Кочовски, П.Д. Дробинцев

Санкт-Петербургский политехнический университет Петра Великого,
Санкт-Петербург, Российская Федерация

Современные среды разработки программного обеспечения на основе компонентно-ориентированного программирования позволяют беспрепятственно развертывать облачные приложения в различных вычислительных инфраструктурах, таких как Edge-to-Cloud. Неоднородная природа таких вычислительных ресурсов приводит к непостоянному качеству обслуживания (QoS). Поэтому решение о развертывании приложения может серьезно повлиять на его общую производительность. В статье рассмотрен подход к автоматизированному развертыванию облачных приложений в вычислительном континууме Edge-to-Cloud, учитывающий нефункциональные требования (NFR). Исследованы способы выбора оптимальной услуги с точки зрения ожидаемого качества обслуживания. Экспериментальная оценка проведена с помощью облачного приложения для хранения данных в трех случаях с разной нагрузкой. Проведены эксперименты на кластере Kubernetes, состоящем из 44 вычислительных узлов (облачных инфраструктур). Узлы кластера были географически распределены в нескольких местах и размещались несколькими поставщиками услуг. Подход позволит надежно выбирать инфраструктуры из гетерогенных Edge-to-Cloud сред, удовлетворяющих требованиям к качеству обслуживания облачных приложений.

Ключевые слова: облачные вычисления, развертывание облачных приложений, качество обслуживания, инфраструктура как услуга, Edge-to-Cloud.

Ссылка при цитировании: Кочовски П., Дробинцев П.Д. Подход к автоматизированному развертыванию облачных приложений в вычислительном континууме Edge-to-Cloud, удовлетворяющих высоким требованиям к качеству обслуживания // Информатика, телекоммуникации и управление. 2020. Т. 13. № 1. С. 8-18. DOI: 10.18721/JCSTCS.13101

Introduction

Intensive development of the Internet-of-Things (IoT), has led towards the development of smart applications in different domains (e.g. smart cities, smart environments, industry 4.0). In order to assure high Quality of Service (QoS), a plethora of non-functional requirements (NFRs), such as computing and network performance must be addressed throughout application's life-cycle.

Novel software development environments support software development based on component-based development (CBD). In other words, they allow to compose software from existing microservices, discovery and selection of computing resources, deployment and resource orchestration in heterogeneous computing environments. The deployment process of microservices in such heterogeneous environments, covering the complete computing Edge-Fog-Cloud continuum, is a difficult problem. At this stage the microservice needs to be placed on optimal or near to optimal infrastructure from a large number of options, whilst considering multiple quality constraints.

The goal of this study is to describe an approach for resource balancing that increases application's performance by ranking and automatically deploying applications on optimal Edge-to-Cloud computing infrastructure.

Related work

The selection of optimal computing infrastructure (i.e. Infrastructure as a Service – IaaS) has been a point of interest in various studies related to load balancing [1–4], resource management and allocation [5–8], resource provisioning [9–11] or service placement and management systems [12, 13]. Since in such cases is necessary to consider large number of NFRs, the reviewed studies recommend implementing various multi-criteria approaches for different placement scenarios in Edge-to-Cloud computing environments.

The authors [14–16] describe the implementation of a multi-criteria decision-making method called the Analytic Hierarchy Process (AHP) for ranking various computing infrastructures. In order to perform the ranking, the AHP executes pairwise comparison of infrastructures instead of considering the software engineer's QoS requirements.

Zheng et al. [17] proposed a framework that ranks the infrastructures according to QoS requirements. In order to perform the ranking, the framework implements two forecasting algorithms, which calculate the ranking results based on the software engineer's QoS requirements. However, the framework's results are only based on network-level measurements data from prior usage experience.

Another commonly used method that is used to compute optimal cloud infrastructure is the Pareto method, which is used to find the optimal set of solutions by performing a trade-off between the conflicting objectives. Guerrero et al. [18] described an approach for resource allocation that is based on the Pareto optimization and implements Non-dominated sorting genetic algorithm (NSGA-II). In addition, another study [19] also utilizes the Pareto optimization for a trade-off of non-functional requirements at the earliest stages of the software development process and thus place the software on an optimal cloud infrastructure. However, the reviewed Pareto-based solutions suggest their results are based on no more than three criteria. Although some studies proposed more than three criteria, they all combined them into two or three main criteria upon which a trade-off was performed and a decision was derived. Using Pareto optimization for more than three criteria is also computationally expensive [20]. Moreover, it is impossible to visualize the Pareto curve on one figure for more than three criteria.

In comparison to the studies considered above, where cloud computing is considered as deterministic, multiple studies consider cloud computing as stochastic [21, 22]. In particular, they investigate the infrastructure's dependability on uncertainty. Moreover, numerous studies in various domains utilize Markov decision-process (MDP) to make decisions in random cases where unforeseen situations may arise. In this context, MDP is also suitable for applications in the field of cloud computing due to its stochastic nature. Yang et al. [23] present an MDP-based method to select a deployment infrastructure that provides optimal performance for applications. Su et al. [24] also proposed an MDP-based planning mechanism that maintains a compromise between the three attributes (accuracy, data usage, and computational cost) by implementing an iterative approach for decision making. In addition, the studies of Tsoumakos et al. [27] and Naskos et al. [28] applied MDP to the problem of horizontal scaling of virtual machines. However, their computational complexity hinders the integration of such methods in the main software practices, thus this challenge has not been addressed in existing studies. MDP also allows to formally verify the correctness of the deployment decision placement. Llerena et al. [25] developed a methodology for analyzing the influence of probability perturbations by checking the reachability properties of MDP models with applications for cloud computing.

Nevertheless, as far as we know, the use of MDP to ensure high quality of service when deploying a software component in the context of containers within the Edge-to-Cloud computing environments has not yet been considered.

Approach for automated deployment of microservices in Edge-to-Cloud computing environments

The foundation of this work is the hypothesis of implementing MDP as an effective decision-making mechanism for deploying microservices on an optimal cloud infrastructure by taking into account specific quality requirements, relevant infrastructure and network measurements.

Fig. 1 illustrates the process of ranking all available computing infrastructures and automated deployment of microservices by considering NFRs and their utilization context. The process is composed of five consecutive steps that are described as follows.

At the first step, the software engineer composes an application from containerized microservices that need to be deployed. The engineer selects the important NFRs, such as: location, cost, network performance, infrastructure performance and etc. In addition, at this step the engineer also defines threshold values for the chosen NFRs. However, this choice may vary significantly between different types of microservices. In other words, the engineer determines which requirements must be continuously met at run time (i.e. hard constraints), and which requirements are desirable but not mandatory (i.e. soft constraints). Once hard and soft constraints are defined, they are used in the two final stages of the automated decision-making process. Hard constraints are used as input parameters for the equivalence classification (second step), while soft constraints are used at the stage of generation and verification of the probabilistic model (third step).

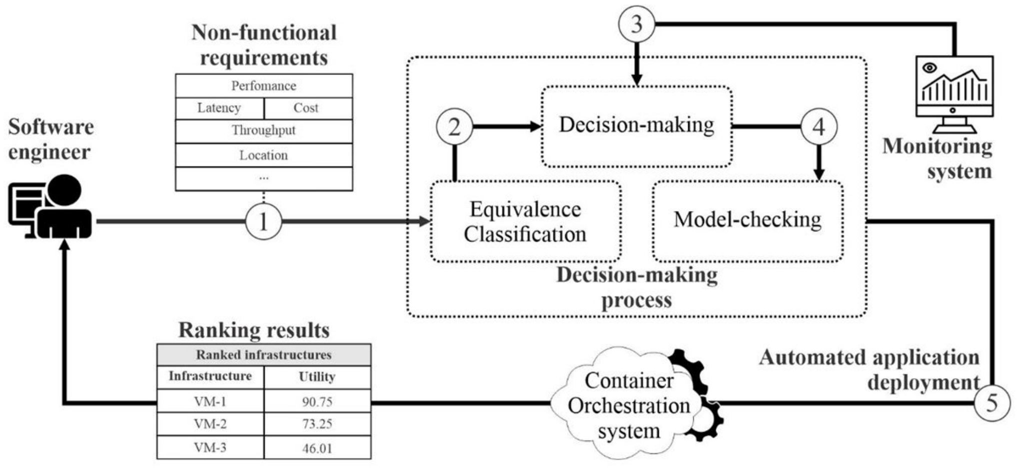

Fig. 1. Approach for automated deployment of cloud applications in the Edge-to-Cloud computing continuum

The second step is responsible for reducing the computational complexity of the method by reducing the number of computations for the decision-making process. As a result, the decision-making process considers only infrastructures that meet the hard constraints. At the beginning of this step, an automaton from all available deployment infrastructures is built. Each state of the automaton represents a deployment infrastructure. Once the automaton is built, an automated process, which classifies infrastructures into classes, is initiated. For instance, an equivalence class can be composed of all available deployment infrastructures that contain at least 8 CPU, or that are located in the territory of Russia.

The goal of the third step is to build a probabilistic model. In order to build the probabilistic model, multiple QoS metrics that represent the past and present performance of the infrastructures are used. These metrics are collected and stored in databases using a multi-tier monitoring system. However, this step only utilizes NFRs that are defined as soft constraints. At the end, the decision-making mechanism calculates the rank scores and sort all of the deployment infrastructures, which were included in this step.

The forth step verifies the results that are obtained from the previous. Using formal criteria and model-checking method, this step verifies the number of NFRs that are satisfied in the equivalence class. The estimated verification value is the output of the probabilistic model and represents a formal guarantee for achieving high QoS.

At the fifth step, the top-ranking infrastructure is automatically selected, where the microservice is going to be deployed using an orchestration tool (e.g. Kubernetes). This step is carried out under the assumption that the formal guarantee obtained for the top-ranking infrastructure, that is, the one with the highest score, is acceptable to the software developer.

Implementation

Applications in smart environments constantly generate and utilize large amounts of different formats and sizes of unstructured data. As a result, traditional cloud computing infrastructures cannot achieve the desired QoS. However, implementing a datacentric architecture, which offers moving the computing in close proximity to data sources, can be a solution to this problem.

Fig. 2 depicts a multi-tier architecture that was developed throughout the period of this research. It complies with the interoperability standards set by organizations such as: Cloud Native Computing Foundation (CNCF), Edge Computing Consortium Europe (ECCE), and OpenFog Consortium. The proposed design is composed of three tiers: Graphical User Interface, decision-making tier and computing tier with available Edge-to-Cloud infrastructures for the deployment of containerized microservices.

The Graphical User Interface (GUI) is an entry point for the software engineer, which is used to compose an application from containerized microservices, manage QoS requirements and input parameters for the deployment process, and initiate the deployment process. The GUI is implemented using EmberJS framework, which offers several views (e.g. component creation view and application composition view).

The decision-making tier is responsible for estimating an optimal deployment infrastructure based on MDP. In addition, this module also verifies the deployment decision and analyzes possible scenarios of the redistribution of microservices from one infrastructure to another at some point in time in the future. This tier also incorporates a container-orchestration system for automated application deployment, which initiates the deployment process after MDP estimates and verifies optimal deployment infrastructure.

The Edge-to-Cloud computing tier is composed of IoT devices, monitoring components and infrastructures that are used for deployment of containerized microservices and data. The infrastructures in this tier are used to store and process data in the computational continuum. Depending on the purpose and requirements of the deployed application, the proposed architecture allows to deploy application's containers in close proximity to data resources (i.e. Edge), the Fog or the Cloud infrastructures.

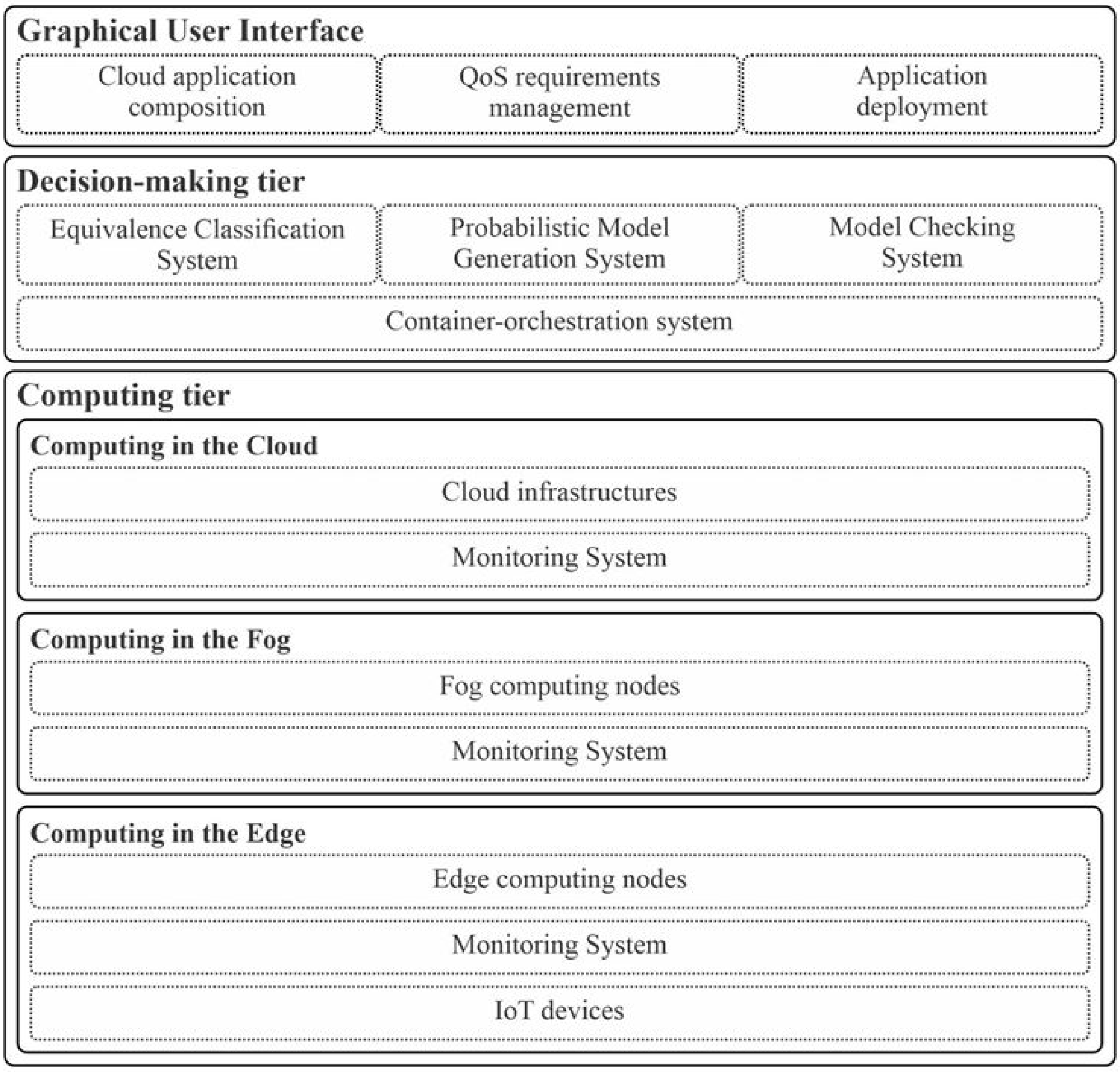

Fig. 2. High level system architecture

Deployment decision-making mechanism

The deployment decision-making mechanism is composed of two systems: monitoring system and MDP-based system for decision-making and verification.

The MDP-based system for decision-making and verification is composed of three subsystems: Equivalence Classification Subsystem (ECS), Probabilistic Model Generation Subsystem (PMGS) and Model Checking Subsystem (MCS). ECS initiates the decision-making process. First it retrieves all available infrastructures and generates an initial model, which is used as an input parameter for equivalence classification. Then, ECS discovers the infrastructures that satisfy the hard constraints and assigns them to the equivalence class. The composed equivalence class as an output from ECS is forwarded to PMGS. PMGS performs decision-making based on MDP method. First, PMGS generates a finite probabilistic model, where each state in the model is a different member of the equivalence class. Then, PMGS calculates the transition probability values, state rewards and state utility. Both subsystems, ECS and PMGS are developed using Java-based development technologies, such as: Java Jersey for RESTful web services and Apache Maven for software management.

MCS performs model-checking over the model and the output from PMGS based on probabilistic computation tree logic (PCTL). In other words, MCS checks the extent to which the selected optimal infrastructure will satisfy application's NFRs regarding the specific restrictions that were set by the software engineer. Therefore, MCS assures the engineer that the mechanism provides the optimal infrastructure for the application. This subsystem integrates the PRISM model-checker into the mechanism, which is used to analyze probabilistic models. To execute the model-checking, PRISM imports models from configuration scripts.

When the deployment infrastructure is selected and verified, the deployment decision-making mechanism generates a YAML script with deployment instructions for the container-orchestration system. The instructions described in the YAML script provide information on the deployment infrastructure, applications for deployment, backup and replication policies.

The proper work of the MDP-based system for decision-making and verification strongly depends on the monitoring system. The monitoring system is a set of monitoring components, such as: monitoring probes, monitoring agents, monitoring server, databases and knowledge bases. It plays an important role in the proposed mechanism, because it is used to collect input data for the decision-making process (e.g. throughput, latency, CPU and memory utilization) and ensures that any application satisfies the QoS requirements at runtime. Usually, the monitoring system begins to work once an infrastructure becomes available to the system. The monitoring system was implemented by using Jcatascopia, NetData and Prometheus monitoring systems.

Monitoring Agents are lightweight components that control the collection of metrics from virtual machine and container instances. Monitoring Probes are metric collectors managed by the Monitoring Agents. They are designed to collect low-level and high-level metrics. Monitoring Probes send metrics to the appropriate Monitoring Agent either periodically or when a specific event occurs. The Monitoring Server is used to collect the metrics from the Monitoring Agent and forward them to a database. The Monitoring Server must be installed on a host that meets the database hardware requirements. In other words, the host must provide enough memory, processor and disk resources.

To store the metrics from the monitoring system, it is necessary to implement a time series database (TSDB). For the purposes of this study, we integrated Apache Cassandra, which is an open source TSDB.

A Knowledge Base (KB) is used to collect complex information, which is required by ECS and PMGS as input parameters. The KB that was implemented was Apache Jena Fuseki, which collects information about the selected infrastructure, such as: location of the infrastructure, information about the type of application, assessment of the quality of experience and information about the deployment, in the form of RDF semantic triples. Utilizing such a KB allows to perform analysis of long-term trends or conduct a variety of strategic analyzes, such as, utilization trends.

Experimental evaluation

The approach described in this study was experimentally evaluated with a typical cloud scenario for uploading and storing files in the Cloud. The file storage application is designed to be deployed on an infrastructure, used and terminated each time the user needs to use it. The File Storage application is a Java servlet web application that processes requests to upload files to a server. The approach in this work allows one to choose the thresholds of different NFRs for each upload operation, since a container instance can be initiated in a different cloud infrastructure for each file upload operation.

For this experimental evaluation, the application was implemented as a Docker container, which is an advanced technology for application virtualization. Furthermore, the following NFRs were used: infrastructure location (Europe), latency (less than 100 ms), throughput (more than 4 Gbit/s), packet loss (less than 2 %) and quality of experience (more than 4). In this evaluation, infrastructure location was used as a hard constraint, whilst other attributes were used as soft constraints. Also, it was assumed that there are several deployment scenarios. After the initial deployment of the application, the software engineer had two different workload requirements, such as 1000 (deployment 2), 1500 (deployment 3) requests every five seconds. The experimental workload was created using the *httperf* tool. The software engineer was able to deploy the application in one of 42 available Fog-Cloud infrastructures or in one of 2 Edge infrastructures. Edge infrastructures were hosted near to the data sources (i.e. the application user), and Fog-Cloud infrastructures were hosted on the Google Cloud Platform, Amazon AWS EC2 and ARNES in 6 different locations: Ljubljana, Frankfurt, London, Tokyo, Sydney and Oregon. The experimental evaluation results are enlisted in Table.

Experimental evaluation deployment results

Infrastructure	Deployment 1		Deployment 2		Deployment 3	
	Rank	Utility	Rank	Utility	Rank	Utility
RaspberryPi 3	11	0.0	11	0.0	12	0.74010
RaspberryPi 4	5	1.02093	4	1.02093	4	1.02093
arnes	12	0.0	12	0.0	11	0.80221
g1-small	10	0.93593	8	0.93593	7	0.93593
n1-standard-1	7	1.01909	6	1.01909	6	1.01909
n1-standard-2	**1**	**1.12443**	10	0.86495	3	1.02220
n1-standard-4	6	1.01965	5	1.01965	5	1.01965
n1-standard-8	4	1.03024	3	1.03023	2	1.03024
a1.medium	8	1.01688	9	0.93204	8	0.93204
a1.large	3	1.11038	2	1.11038	**1**	**1.11038**
a1.xlarge	2	1.11571	**1**	**1.11571**	10	0.85815
a1.2xlarge	9	1.01473	7	1.01473	9	0.85853

According to the results of the experimental evaluation, the infrastructure n1-standard-2 offered the highest utility value for deploying the application. However, after an additional workload was applied to this infrastructure (1000 requests every five seconds), several quality thresholds were violated, so the application was transferred to the a1.xlarge infrastructure. Following the same steps, with a workload of 1,500 requests, the application was again redeployed to another infrastructure (i.e. a1.large), which guaranteed high QoS.

Conclusion

The goal of this work was to design a QoS-aware approach that guarantees high QoS for cloud applications in highly dynamic and heterogeneous Edge-to-Cloud environments. QoS are relevant requirements that software engineers at the deployment stage of cloud applications have to comply with. This study offers an approach that can be used for this purpose and which can be integrated into software development tools.

The results of the experimental evaluation elaborate that the proposed approach is universal enough for the automated deployment of applications with different QoS requirements in various infrastructures. This means that the proposed approach is not limited to a specific set of NFRs or types of applications.

The approach can be further expanded by improving deployment algorithms to implement multi-tier application deployment operations across multiple infrastructures, where each application tier is deployed in a different Edge-to-Cloud infrastructure.

REFERENCES

1. **Hu J., Gu J., Sun G., Zhao T.** A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. *Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms and Programmin*g, 2010, Pp. 89–96.

2. **Li L.E., Woo T.** Dynamic load balancing and scaling of allocated cloud resources in an enterprise network, *US Patent App. 12/571,271*, Mart 31, 2011.

3. **Randles M., Lamb D., Taleb-Bendiab A.** A comparative study into distributed load balancing algorithms for cloud computing. *Proceedings of the IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, 2010, Pp. 551–556.

4. **Chaczko Z., Mahadevan V., Aslanzadeh S., Mcdermid C.** Availability and load balancing in cloud computing. *Proceedings of the International Conference on Computer and Software Modeling.* Singapur, 2011, Vol. 14.

5. **Manvi S., Shyam G.K**. Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. *Journal of Network and Computer Applications*, 2014, Vol. 41, Pp. 424–440.

6. **Jennings B., Stadler R.** Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, 2015, Vol. 23, No. 3, Pp. 567–619.

7. **Luong N.C., Wang P., Niyato D., Wen Y., Han Z.** Resource management in cloud networking using economic analysis and pricing models: A survey. *IEEE Communications Surveys & Tutorials*, 2017, Vol. 19, No. 2, Pp. 954–1001.

8. **Jain N., Menache I**. Resource management for cloud computing platforms, *US Patent 9,450,838*, Sept. 20, 2016.

9. **Singh S., Chana I.** Q-aware: Quality of service based cloud resource provisioning. *Computers & Electrical Engineering*, 2015, Vol. 47, Pp. 138–160.

10. **Chaisiri S., Lee B.S., Niyato D.** Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*, 2012, Vol. 5, No. 2. Pp. 164–177.

11. **Zhang L., Li Z., Wu C.** Dynamic resource provisioning in cloud computing: A randomized auction approach. *Proceedings of the IEEE Conference on Computer Communications*, 2014, Pp. 433–441.

12. **Paščinski U., Trnkoczy J., Stankovski V., Cigale M., Gec S.** QoS-aware orchestration of network intensive software utilities within software defined data centres. *Journal of Grid Computing*, 2018, Vol. 16, No. 1, Pp. 85–112.

13. **Mijumbi R., Serrat J., Gorricho J.L., Latre S., Charalambides M., Lopez D.** Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 2016, Vol. 54, No. 1, Pp. 98–105.

14. **Karim R., Ding C., Miri A.** An end-to-end QoS mapping approach for cloud service selection. *Proceedings of the IEEE 9th World Congress on Services*, 2013, Pp. 341–348.

15. **Garg S.K., Versteeg S., Buyya R.** A framework for ranking of cloud computing services. *Future Generation Computer Systems*, 2013, Vol. 29, No. 4, Pp. 1012–1023.

16. **Goncalves Junior R., Rolim T., Sampaio A., Mendonca N.C.** A multi-criteria approach for assessing cloud deployment options based on non-functional requirements. *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, 2015, Pp. 1383–1389.

17. **Zheng Z., Wu X., Zhang Y., Lyu M.R., Wang J.** QoS ranking prediction for cloud services. *IEEE Transactions on Parallel and Distributed Systems*, 2012, Vol. 24, No. 6, Pp. 1213–1222.

18. **Guerrero C., Lera I., Juiz C.** Genetic algorithm for multi-objective optimization of container allocation in cloud architecture. *Journal of Grid Computing*, 2018, Vol. 16, No. 1, Pp. 113–135.

19. **Štefanič P., Kimovski D., Siciu G., Stankovski V.** Non-functional requirements optimisation for multi-tier cloud applications: An early warning system case study. *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation*, 2017, Pp. 1–8.

20. **Guo X., Wang Y., Wang X.** Using objective clustering for solving many-objective optimization problems. *Mathematical Problems in Engineering*, 2013, Vol. 2013.

21. **Trenz M., Huntgeburth J., Veit D.** The role of uncertainty in cloud computing continuance: Antecedents, mitigators, and consequences. *ECIS*, 2013, P. 147.

22. **Tchernykh A., Schwiegelsohn U., Alexandrov V., Talbi E.G.** Towards understanding uncertainty in cloud computing resource provisioning. *Procedia Computer Science*, 2015, Vol. 51, Pp. 1772–1781.

23. **Yang J., Lin W., Dou W.** An adaptive service selection method for cross-cloud service composition. *Concurrency and Computation: Practice and Experience*, 2013, Vol. 25, No. 18, Pp. 2435–2454.

24. **Su G., Chen T., Feng Y., Rosenblum D., Thiagarajan P.** An iterative decision-making scheme for Markov decision processes and its application to self-adaptive systems. *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*, 2016, Pp. 269–286.

25. **Llerena Y.R.S., Su G., Rosenblum D.S.** Probabilistic model checking of perturbed mdps with applications to cloud computing. *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, Pp. 454–464.

26. **Mardani A., Jusoh A., Nor K., Khalifah Z., Zakwan N., Valipour A.** Multiple criteria decision-making techniques and their applications - A review of the literature from 2000 to 2014. *Economic Research*, 2015, Vol. 28, No. 1, Pp. 516–571.

27. **Tsoumakos D., Konstantinou I., Boumpouka C., Sioutas S., Koziris N.** Automated, elastic resource provisioning for nosql clusters using tiramola. *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, 2013, Pp. 34–41.

28. **Naskos A., Stachtiari E., Gounaris A., Katsaros P., Tsoumakos D., Konstantinou I., Sioutas S.** Dependable horizontal scaling based on probabilistic model checking. *Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2015, Pp. 31–40.

Received 29.01.2020.

СПИСОК ЛИТЕРАТУРЫ

1. **Hu J., Gu J., Sun G., Zhao T.** A scheduling strategy on load balancing of virtual machine resources in cloud computing environment // Proc. of the 3rd Internat. Symp. on Parallel Architectures, Algorithms and Programming. 2010. Pp. 89–96.

2. **Li L.E., Woo T.** Dynamic load balancing and scaling of allocated cloud resources in an enterprise network // US Patent App. 12/571,271, Mart 31, 2011.

3. **Randles M., Lamb D., Taleb-Bendiab A.** A comparative study into distributed load balancing algorithms for cloud computing // Proc. of the IEEE 24th Internat. Conf. on Advanced Information Networking and Applications Workshops. 2010. Pp. 551–556.

4. **Chaczko Z., Mahadevan V., Aslanzadeh S., Mcdermid C.** Availability and load balancing in cloud computing // Proc. of the Internat. Conf. on Computer and Software Modeling. Singapur, 2011. Vol. 14.

5. **Manvi S., Shyam G.K.** Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey // J. of Network and Computer Applications. 2014. Vol. 41. Pp. 424–440.

6. **Jennings B., Stadler R.** Resource management in clouds: Survey and research challenges // J. of Network and Systems Management. 2015. Vol. 23. No. 3. Pp. 567–619.

7. **Luong N.C., Wang P., Niyato D., Wen Y., Han Z.** Resource management in cloud networking using economic analysis and pricing models: A survey // IEEE Communications Surveys & Tutorials. 2017. Vol. 19. No. 2. Pp. 954–1001.

8. **Jain N., Menache I.** Resource management for cloud computing platforms // US Patent 9,450,838. Sept. 20, 2016.

9. **Singh S., Chana I.** Q-aware: Quality of service based cloud resource provisioning // Computers & Electrical Engineering. 2015. Vol. 47. Pp. 138–160.

10. **Chaisiri S., Lee B.S., Niyato D.** Optimization of resource provisioning cost in cloud computing // IEEE Transactions on Services Computing. 2012. Vol. 5. No. 2. Pp. 164–177.

11. **Zhang L., Li Z., Wu C.** Dynamic resource provisioning in cloud computing: A randomized auction approach // Proc. of the IEEE Conf. on Computer Communications. 2014. Pp. 433–441.

12. **Paščinski U., Trnkoczy J., Stankovski V., Cigale M., Gec S.** QoS-aware orchestration of network intensive software utilities within software defined data centres // J. of Grid Computing. 2018. Vol. 16. No. 1. Pp. 85–112.

13. **Mijumbi R., Serrat J., Gorricho J.L., Latre S., Charalambides M., Lopez D.** Management and orchestration challenges in network functions virtualization // IEEE Communications Magazine. 2016. Vol. 54. No. 1. Pp. 98–105.

14. **Karim R., Ding C., Miri A.** An end-to-end QoS mapping approach for cloud service selection // Proc. of the IEEE 9th World Congress on Services. 2013. Pp. 341–348.

15. **Garg S.K., Versteeg S., Buyya R.** A framework for ranking of cloud computing services // Future Generation Computer Systems. 2013. Vol. 29. No. 4. Pp. 1012–1023.

16. **Goncalves Junior R., Rolim T., Sampaio A., Mendonca N.C.** A multi-criteria approach for assessing cloud deployment options based on non-functional requirements // Proc. of the 30th Annual ACM Symp. on Applied Computing. 2015. Pp. 1383–1389.

17. **Zheng Z., Wu X., Zhang Y., Lyu M.R., Wang J.** QoS ranking prediction for cloud services // IEEE transactions on parallel and distributed systems. 2012. Vol. 24. No. 6. Pp. 1213–1222.

18. **Guerrero C., Lera I., Juiz C.** Genetic algorithm for multi-objective optimization of container allocation in cloud architecture // J. of Grid Computing. 2018. Vol. 16. No. 1. Pp. 113–135.

19. **Štefanič P., Kimovski D., Siciu G., Stankovski V.** Non-functional requirements optimisation for multi-tier cloud applications: An early warning system case study // 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation. 2017. Pp. 1–8.

20. **Guo X., Wang Y., Wang X.** Using objective clustering for solving many-objective optimization problems // Mathematical Problems in Engineering. 2013. Vol. 2013.

21. **Trenz M., Huntgeburth J., Veit D.** The role of uncertainty in cloud computing continuance: Antecedents, mitigators, and consequences // ECIS. 2013. P. 147.

22. **Tchernykh A., Schwiegelsohn U., Alexandrov V., Talbi E.G.** Towards understanding uncertainty in cloud computing resource provisioning // Procedia Computer Science. 2015. Vol. 51. Pp. 1772–1781.

23. **Yang J., Lin W., Dou W.** An adaptive service selection method for cross-cloud service composition // Concurrency and Computation: Practice and Experience. 2013. Vol. 25. No. 18. Pp. 2435–2454.

24. **Su G., Chen T., Feng Y., Rosenblum D., Thiagarajan P.** An iterative decision-making scheme for Markov decision processes and its application to self-adaptive systems // Proc. of the Internat. Conf. on Fundamental Approaches to Software Engineering. 2016. Pp. 269–286.

25. **Llerena Y.R.S., Su G., Rosenblum D.S.** Probabilistic model checking of perturbed mdps with applications to cloud computing // Proc. of the 2017 11th Joint Meeting on Foundations of Software Engineering. 2017. Pp. 454–464.

26. **Mardani A., Jusoh A., Nor K., Khalifah Z., Zakwan N., Valipour A.** Multiple criteria decision-making techniques and their applications - A review of the literature from 2000 to 2014 // Economic Research. 2015. Vol. 28. No. 1. Pp. 516–571.

27. **Tsoumakos D., Konstantinou I., Boumpouka C., Sioutas S., Koziris N.** Automated, elastic resource provisioning for nosql clusters using tiramola // Proc. of the 13th IEEE/ACM Internat. Symp. on Cluster, Cloud, and Grid Computing. 2013. Pp. 34–41.

28. **Naskos A., Stachtiari E., Gounaris A., Katsaros P., Tsoumakos D., Konstantinou I., Sioutas S.** Dependable horizontal scaling based on probabilistic model checking // Proc. of the 15th IEEE/ACM Internat. Symp. on Cluster, Cloud and Grid Computing. 2015. Pp. 31–40.

Статья поступила в редакцию 29.01.2020.

THE AUTHORS / СВЕДЕНИЯ ОБ АВТОРАХ

KOCHOVSKI Petar
КОЧОВСКИ Петар
E-mail: petako_bt@hotmail.com

DROBINTSEV Pavel D.
ДРОБИНЦЕВ Павел Дмитриевич
E-mail: drob@ics2.ecd.spbstu.ru