Real-Time Adaptable Resource Allocation for Distributed Data-Intensive Applications over Cloud and Edge Environments

Jean-Didier Totow Tom-Ata and Dimosthenis Kyriazis

October 24, 2020

Real-time adaptable resource allocation for distributed data-intensive applications over cloud and edge environments

Jean-Didier Totow Tom-Ata
Department of Digital Systems
University of Piraeus
Piraeus, Greece
totow@unipi.gr

Dimosthenis Kyriazis
Department of Digital Systems
University of Piraeus
Piraeus, Greece
dimos@unipi.gr

Abstract— **Applications performance is strongly linked with the total load, the application deployment architecture and the amount of resources allocated by the cloud or edge computing environments. Considering that the majority of the applications tends to be data intensive, the load becomes quite dynamic and depends on the data aspects, such as the data sources locations, their distribution and the data processing aspects within an application that consists of micro-services. In this paper we introduce an analysis and prediction model that takes into account the characteristics of an application in terms of data aspects and the edge computing resources attributes, such as utilization and concurrency, in order to propose optimized resources allocation during runtime.**

Keywords—cloud computing, edge computing, resource allocation, real-time adaptation

I. INTRODUCTION

While cloud computing environments offer services, resources and tools towards application owners, application developers and infrastructure administrators, the generation of big amounts of data from various data sources raises the need for computing environments that are non-centralized and can serve the time- and latency- constrained requirements of users and applications. Such requirements have driven the emerge of the edge / fog computing paradigm, providing the required resources on the network edge. Such approach allows to offload the cloud by pushing computing units near the place workload are being generated. Applications deployed on the cloud/edge may be composed of more than one component which together fulfill some goal. Based on today's virtualization techniques, application's components can be modelled, deployed, and orchestrated separately. This adds flexibility to the application lifecycle and move the application modelling from monolithic to micro-service, decentralized- and distributed- based model.

Applications running on the cloud must satisfy the owner's expectations, otherwise the quality of service (QoS), which is the compliance of the application performance with the application's objectives, will be violated and thus result to a redeployment/adaptation plan. Since application's workload changes over time, depending on the number of users, number/nature of requests/task users are executing, cloud providers must implement method for adapting dynamically these applications to maintain QoS into the acceptance window. In the cloud domain, many cloud infrastructures have been created exploiting application workload to plan deployment which improve the quality of service by choosing the most suitable configuration. By configuration, we mean, the amount of resource allocated to the application such us memory, CPU (Central Processing Unit), bandwidth, the number of instances by application's component and the computation device type such as : CPU, GPU(Graphical Processing Unit), FPGA (Field-Programmable Gate Array). In these infrastructures, decisions are driven by data, therefore the name of data-driven infrastructure. These infrastructures address the economic issue in the cloud computing since the resource allocated to an application is proportional to the need. In the provider perspective, data-driven infrastructure enables better resource management by proceeding to the allocation on-demand.

With the increase number of devices connecting into the cloud (edge devices), the strategy consisting of pushing the workload processing near to the place they are being produced created the edge/fog. This strategy does not only offload cloud environments but also reduces the transportation cost and enable the optimization of the application performance from the user perspective (improvement of the latency). The edge/fog computing offers a distributed model which adds the complexity in the application modeling. We will introduce in this work, the adaptation of applications running on cloud/edge environments by exploiting the analysis of the application performance considering the resources allocation, the inter-component constraints and the users' locality. The latter is coupled with a distributed orchestration approach to address edge node resources constraints. This leads to the establishment of a model called application performance model, which allows the prediction of the application performance given a certain configuration (i.e. resources, architecture, and users distribution).

The remainder of the paper is structured as follows: Section II presents related work in the scope of real-time application adaptation and resources allocation on cloud and edge environments. Section III presents the theoretical framework, and the Section IV develops our approach in an algorithmic manner and Section V concludes the paper.

II. RELATED WORK

BigDataStack [1] is an infrastructure management system that provides management of computers, storage, and network in an intensive data application [1]. Applications can be developed in BigDataStack by a definition that describes the application (playbook). This later contains the requirements and goals of the applications. The application uses OpenShift as executable software, therefore, applications are developed in docker containers and can automatically detect congestion from the application, data mode components, platform and decide the most appropriate

changes for the application. The customization provided by BigDataStack utilizes the reference technique to identify the resources required for initial development. Then, the data provided by the monitoring engine allows the platform to make the most appropriate decision to achieve the implementation objectives set in the SLA (Service Level Agreement). Application components can be in different geographical locations, however, development or rearrangement performed by BigDataStack is based on resource management.

A distributed Reinforcement Learning (RL) mechanism called iBalloon [2] for self-adapting Virtual Machine (VM) resources in which monitoring is necessary for autonomous orchestration and adaptation has been introduced in [8]. Cloud infrastructures offer flexible resources with horizontal or vertical scaling solutions to adjust application performance to changing workloads. However, such scaling approaches that use only infrastructure-related monitoring data can cause severe performance declines during workload changes at runtime. The authors argue that monitoring infrastructure-level metrics, such as memory and bandwidth, without considering application performance behavior (application-level monitoring) at runtime will complicate the resource allocation problem due to the lack of detailed measurements. In their work, according to the proposed vertical scaling approach, each VM can adjust the resource allocation in terms of CPU, memory, and bandwidth. The iBalloon architecture includes three fundamental elements: (1) a host that is responsible for allocating resources to VMs. (2) an app-agent that includes the iBalloon tracking section and reports runtime performance information; and (3) a decision-maker hosting an RL agent placed in each VM to automatically adjust resource capacity. However, iBalloon is limited because it does not consider other virtual resources e.g. storage, nor does it support other customization actions, e.g. relocation to improve application performance. This is an issue because to manage an application that develops in an edge computing framework, it is necessary to consider migrating component applications between heterogeneous infrastructure.

Islam [3] developed a precautionary cloud resource management approach that applied linear regression and neural networks to predict and meet future resource requirements. The research problem in this project is actually the analysis of time series tracking data to derive a prediction model and other characteristics of the tracking data. The proposed performance prediction model estimates the upcoming resource usage at runtime and is able to launch additional VMs to maximize application performance. The authors considered the predictive accuracy based on the performance of the application in terms of response time. This approach provides distributed scalability and can be improved to address the resource allocation of a single VM as well. At present, only CPU usage is used to predict model training and their approach could further include other types of resources, e.g. memory, disk and bandwidth.

A self-learning adaptation technique called FQL4KE [4], which is a fuzzy control method based on a reinforcement learning algorithm for optimal resilience policies has been introduced in [10]. This approach aims to automate the scaling process without utilizing a priori knowledge about the cloud application running. The proposed architecture includes a learning module that continuously upgrades the controller knowledge base by learning customization rules appropriate for the system. However, the proposed approach may not address different objectives. If the system goals change, the controller must re-learn everything from ginner. A more fundamental problem in real-world environments is the fact that the number of situations can be huge and therefore the learning process could become impossible due to time constraints on new computing standards.

Stankovski and all [5] proposed another interesting approach refers to a distributed self-adapting architecture that applies the concept of computing edges with container-based technologies, such as Docker and Kubernetes, to ensure QoS for time-critical applications. The basic idea is to develop the application in a container (case of using file upload) in different geographical locations in a way that the service is created, served and destroyed for each file upload request. For each container, the resources required for the host can be allocated when monitoring data and business strategies defined by the end user, application developer and/or administrator.

Melodic is a multi-cloud management system designed for intensive data application [6]. Melodic can optimize an application to meet user requirements by detecting obstacles from an enhanced learning algorithm. Melodic can develop different application components in different provider types (public and private cloud), so it supports multi-cloud management. The owner of the application provides an application profile in a meta-cloud application language (CAMEL). This file to be compiled contains the expectations of the application, the performance of the index, the descriptions of the data, etc. Melodic turns users' expectation and application goal into a limited problem that is analyzed and solved to find different feasible application configurations from the perspective of users and the cloud provider. Melodic chooses the configuration with the highest utility value.

The main difference between the aforementioned approaches and the one proposed in this paper refers to the orchestration approach for an edge computing network. Thus, it optimizes the application execution both proactively and reactively in terms of performance, while at the same time delivering a better user experience by incorporating the user's latency indicator into the performance model.

III. THEORETICAL FRAMEWORK

A. General architecture of a cloud/edge network

The figure below illustrates the general architecture of a cloud/edge environment. Devices generating data are connected to edge gateway or edge node directly which has an access point and a computing power. Applications serving those devices should be deployed in such way to fit node in a radio and computational perspective. Each edge node requires an edge node core: set of components providing the node the ability to receive usage request, registering devices information and routing users' request to applications.

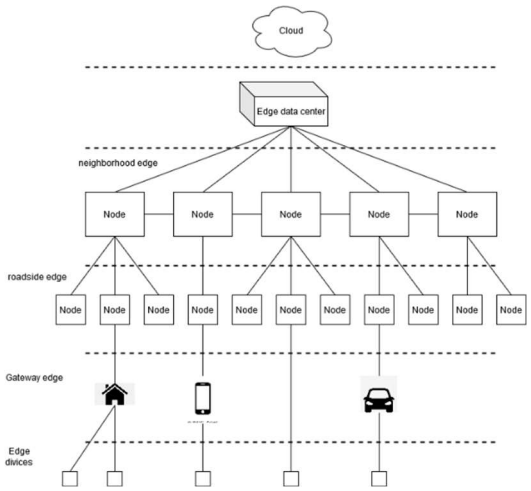

Fig. 1: General architecture of an edge network, hierarchical representation

The edge node core also implements methods for collecting information (monitoring) [7], metrics by measuring in real-time the state of each performance indicator. These metrics are exposed to the orchestrator located to the cloud for a better understanding and management of the entire platform. Our approach for collecting metrics consists of the exploitation of the service discovering provided by metric collector engines [8].

B. Real-time adaptation

Application's configuration can be modified dynamically based on QoS' evolution by triggering a scaling signal. The term adaptation or modification refers to the update of the resource allocate to the application, instance scaling, instance and users migration. The collection and analysis of a Service Level Objective (SLO) and all metrics related to the expression of the application performance through a monitoring mechanism is a crucial operation in order to obtain the necessary data required for applying the most suitable change on the infrastructure level [9]. This raises the need for the infrastructure to realize the data-driven model described in the following figure.

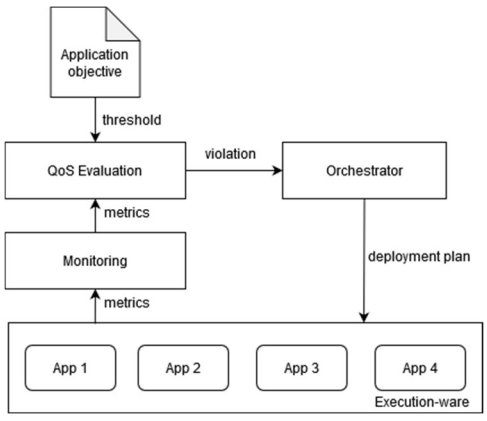

Fig. 2: Data-driven model

In the edge/fog environment, the goal is to handle the workload near the edge devices to minimize the latency comparing to the utilization of cloud nodes. In this context, the orchestrator except from the optimization of the application by fitting the minimum hardware requirements and the application's objectives need to implement technics consisting of detecting users and edge node locality. In a distributed environment, application composed of more than one component may have deployment pattern where different application components are not deployed on the same node due to node hardware limitation or all components are not replicable. Since some applications present internal constraints (i.e. constraints between the application's components) such us low latency or high throughput between application components and since those constraints have an impact to the application performance, the modeling of the orchestrator reasoning becomes a challenging task. Many applications require a very low latency such us self-driving, some others require a very high availability. These applications may present malfunctioning by the redeployment time or by the period where there are running under performance. The re-adaptation time represents another sensitive parameter to consider when designing an orchestrator for such cases.

C. Performance model

Before proceeding to the design of the proposed approach (adaptation), we briefly present some baseline facts.

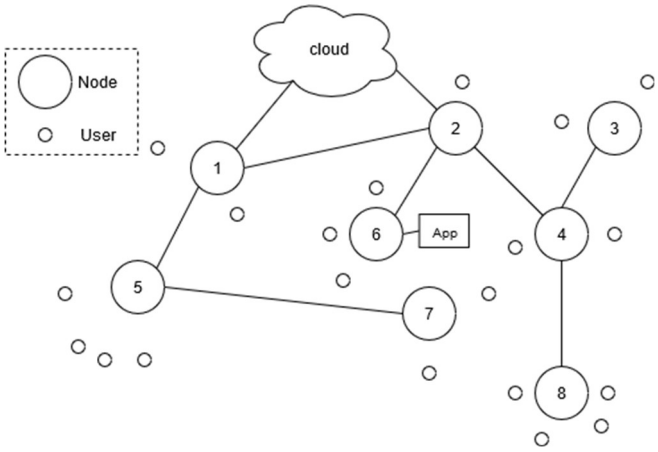

Fig. 3: Edge/cloud architecture, network representation

According to the picture above, the main ground facts / assumptions are summarized as follows: (i) Each node is an access point having a server with a memory and a computation power. Nodes can run applications if the total amount of resource required are satisfied. (ii) The latency between each node and between nodes and the cloud is equal. (iii) The latency between each user and the access point is equal and is at least 10 times higher compared to the latency between two nodes. (iv) User connects by default to the nearest node, but the orchestrator can trigger the reallocation of any user to any node.

Since the orchestrator can handle non-monolithic applications, our approach considers each component separately. The ideal scenario is when every application's component can be deployed into the same node for avoiding inter component constraints violation.

IV. Architecture, Design, Modeling

Before any optimization, the solver (orchestrator) should assign all application components with a set of nodes. This results to a constraint problem where the inputs are application components, nodes, inter-components constraints, links properties (e.g. latency, bandwidth, etc). The outcome of the constraint problem is a map containing different combinations of each application component with the corresponding node candidate. The best candidate is selected based on performance criteria. The constraint problem is formulated as follows, where resources refer to CPU, memory, etc:

***N* = {n_1 …n_k},** variable, set of available nodes
***C*= {c_1 …c_l},** variable, set of application components
***S* = {(n_1 … n_k) *x* (c_1 … c_l)},** solution universe
N.resource >= C.min_resource_required

Do t = c_1 to c_l
  Do j= c_1 to c_l
    If t != j and constraint_exist(t, j)
      get_latency(t,j) <= get_latency(n_t,n_j)
    End If
  End Do
End Do

Based on the above, our approach for solving this challenge consists of representing the application performance as a function of the application's metrics. Given the application **X**, exposing **a** and **b** as the most significant application performance indicators for instance the application response time (time required by the application to accomplish a task) and the latency from the user perspective. The application **X** is running on a node with a configuration such as: memory (**M**), CPU (**C**), number of instances (**I**), average number of bound (**S**) (average number of steps from users to the node running the application). Thus, the application performance expression will be the following:

$$p = \frac{1}{a*b}$$

This expression can be improved as follows:

$$p = \frac{1}{A*B}$$

with:

$$A = c_1*a'$$
$$B = c_2*b'$$

where $\mathbf{c_1}$ and $\mathbf{c_2}$ comprise between 0 and 1, and are the coefficients chosen by the application owner expressing weights for metrics **a** and **b**. Our strategy will correlate metrics **A** and **B** with the application configuration that implies the establishment of the following expressions:

$$A = F_a(M, C, I, S)$$
$$A = F_b(M, C, I, S)$$

For a linear regression as a correlation method, the performance model is obtained through the following two expressions:

$$A = d_{a1}*M + d_{a2}*C + d_{a3}*I + d_{a4}*S + K_a$$
$$B = d_{b1}*M + d_{b2}*C + d_{b3}*I + d_{b4}*S + K_b$$

where $\mathbf{d_{a1}}$ … $\mathbf{d_{a4}}$ and $\mathbf{d_{b1}}$ … $\mathbf{d_{b4}}$ are coefficients and $\mathbf{K_a}$/$\mathbf{K_b}$ are intercepts**.** The above expressions are obtained from the linear regression with the multiple variables. Based on the **A**, **B**, the orchestrator computes the performance of the application based on its model. The capability of the algorithm to output parameters (coefficients and constants) reflecting the best the application behavior depends on the number of configurations present in the dataset. From these expressions, the orchestrator obtains information about the weight of each configuration property, thus realizing an enhancement of the application performance. According to the non-linearity of parameters influencing the cloud/edge computing applications (e.g. users' load, data center/device temperature etc.), we established correlation using different algorithms.

V. Experimentation, Initial result

The approach develop in this work about performance's representation of a cloud application has been applied in real cases. We will present in this section the result obtained showing the correlation between metrics involved to application performance and configuration provided by the cloud provider. Applications were deployed on a cloud environment where the reasoner exploiting performance model concept was analyzing in real-time the evolution of the application performance indicators for deciding the most suitable configuration which will lead to the satisfaction of the application owner requirements. While an application is running on our environment, metrics related to the infrastructure, data transaction and application specific are being collected and exposed to the reasoner for building correlation model. Dataset preparation represents an important task for achieving satisfying result. We used Prometheus as metrics collector for gathering metrics. We were interested to metrics of the environment docker swarm. The orchestrator exposes the application's configuration properties (CPU Allocated, memory allocated, number of instances and the average latency computed using the number of hops between users and the Nodes executing the replica serving users). Applications deployed included Prometheus exporter for exposing application related metrics. The below picture shows the architecture used for our experiments.

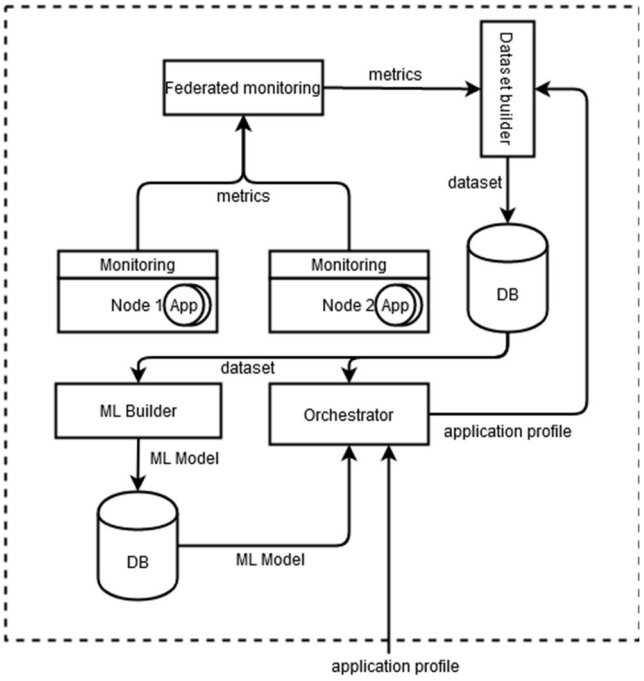

Fig.3: Infrastructure architecture for experiments

TABLE I. ORCHESTRATION OF A WEBSERVICE APPLICATION

Load (requests/sec)	*Memory (Mo)*	*CPUs*	*Replica*
20	32	1	1
40	64	2	2
80	128	3	3
	248	4	4

TABLE II. ORCHESTRATION OF AN APACHE SPARK APPLICATION

Load (users)	*Memory (Mo)*	*CPUs*	*Replica*
100	8	4	1
			2
	16	8	3

After the execution, dataset was built from Prometheus metrics. Performance column must be construct based on its expression. Based on the execution outcomes, the below table shows the accuracy of predicting performance based on configuration by algorithm.

TABLE III. IDENTIFIED RELATIONS

Performance model algorithm	*Accuracy*
Linear regression	56%
Decision Tree Regressor	80.6%
Random Forest Regressor	80.3%
Ada Boost Regressor	80.3%
Gradient Boosting regressor	80%

We can predict the application performance with an 80% of accuracy given a configuration using a Decision Tree Regression algorithm.

VI. CONCLUSIONS AND FUTURE WORK

Dynamic adaptation of resources in a cloud / edge environment is a very promising subject whose importance only grows with the development of applications on different types of devices considering the dynamic nature of user operations. This area is also of great interest to cloud service providers as dynamic adaptation allows for intelligent and very economical management of resources available in the cloud. During our first experiments, we observed the importance of constructing a mathematical expression describing the evolution of various performance indicators with the resources available. The correctness of this representation depends on the number of configurations present in the dataset. Since prediction is a crucial factor in the allocation of resources, the continuation of our work will give great importance to the development of a prediction model that does not require a large amount of data so that this functionality can be used a few minutes after the initial deployment. We will also perform additional work in the scope of the performance of edge devices given their resource-constraint characteristics.

ACKNOWLEDGMENT

The research leading to the results presented in this paper has received funding from the European Union's funded Projects BigDataStack under grant agreement no 779747 and MORPHEMIC under grant agreement no 871643.

REFERENCES

[1] D. Kyriazis et al. BigDataStack: A holistic data-driven stack for big data applications and operations. In Proc. of 2018 IEEE BigData Congress, pages 237–241, 2018.

[2] Rao, J. , Bu, X. , Xu, C. , Wang, K. , 2011. A distributed self-learning approach for elastic provisioning of virtualized cloud resources. In: Proceedings of 2011 IEEE 19th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE, Singapore, pp. 45–54 .

[3] Islam, S. , Keung, J. , Lee, K. , Liu, A. ,2012. Empirical prediction models for adaptive resource provisioning in the cloud. Future Gener. Comput. Syst. 28 (1), 155–162 . Issariyapat, C. , Pongpaibool, P. , Mongkolluksame, S. , Meesublak, K. , 2012. Using Na- gios as a groundwork for developing a better network monitoring system. In: Proceedings of PICMET'12 Conference on Technology Management for Emerging Technologies (PICMET). IEEE, Vancouver, Canada, pp. 2771–2777 .

[4] Jamshidi, P., Sharifloo, A.M., Pahl, C., Metzger, A., Estrada, G., 2015. Self-learning cloud controllers: fuzzy q-learning for knowledge evolution. In: Proceedings of International Conference on Cloud and Autonomic Computing (ICCAC). IEEE, Boston, USA, pp. 208–211. doi: 10.1109/ICCAC.2015.35.

[5] Stankovski, V., Taherizadeh, S., Trnkoczy, J., Suciu, G., Suciu, V., Martin, P., Wang, J., Zhao, Z., 2015. Dynamically reconfigurable workflows for time-critical applications. In: Proceedings of International workshop on Workflows in sup- port of large-scale science (WORKS 15). ACM, Austin, USA, pp. 1–10. doi: 10. 1145/2822332.2822339.

[6] Kritikos, Chrysostomos Zeginis, Joaquin Iranzo, Roman Sosa Gonzalez, Daniel Seybold, Frank Griesinger & Jörg Domaschka, Multi-cloud provisioning of business processes Kyriakos, November 2019, https://link.springer.com/article/10.1186/s13677-019-0143-x

[7] Confais, B., Lebre, A., Parrein, B., 2016. Performance analysis of object store systems in a fog/edge computing infrastructures. In: Proceedings of the 8th International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, Lux- embourg, pp. 294–301. doi: 10.1109/CloudCom.2016.0055 .

[8] Prometheus, service discovery, https://prometheus.io/docs/guides/file-sd/

[9] NEC Laboratories Europe, Mauricio Fadel, Bin Cheng, Reinforcement Learning based Orchestration for Elastic Services