Auto-scaling Applications in Edge Computing: Taxonomy and Challenges

Salman Taherizadeh
University of Ljubljana
Ljubljana, Slovenia
Salman.Taherizadeh@fgg.uni-lj.si

Vlado Stankovski
University of Ljubljana
Ljubljana, Slovenia
Vlado.Stankovski@fgg.uni-lj.si

ABSTRACT

The perspective of online services such as Internet of Things (IoT) applications has impressively evolved over the last recent years as they are becoming more and more time-sensitive, maintained at decentralized locations and easily affected by the changing workload intensity at runtime. As a consequence, an up-and-coming trend has been emerging from previously centralized computation to distributed edge computing in order to address these new concerns. The goal of the present paper is therefore twofold. At first, to analyze modern types of edge computing applications and their auto-scaling challenges to offer desirable performance in conditions where the workload dynamically changes. Secondly, to present a new taxonomy of auto-scaling applications. This taxonomy thoroughly considers edge computing paradigm and its complementary technologies such as container-based visualization.

CCS Concepts

Computer systems organization → Cloud computing • Computing methodologies → Distributed computing methodologies.

Keywords

Auto-scaling; Edge computing; Cloud; Internet of Things (IoT); Taxonomy

1. INTRODUCTION

Recently, emerging cloud-based software solutions such as the ENTICE project[1] desire to not only provide high-quality results, but also deliver the results as soon as possible. To this end, application providers have decided to optimize centralized, faraway cloud-based infrastructures through processing the data at the edge of the network close to the end-users.

Due to the distributed nature of edge computing environment, companies can run their application services on different edge nodes, connecting each one with the users in each region. Such modern framework is aimed to increase capabilities of resources through whether decreasing application response time or locating services near the source of the data e.g. connected devices. Therefore, edge computing as a new promising framework to leverage cloud-based resources has become the prevalent method for providing many different types of online services, such as smart home, remote healthcare and industrial automation over the Internet.
The edge computing trend has involved a growing demand through which organizations can support the Quality of Service (QoS) needed to run their applications at the edge of the network with a low-latency response time, and hence address the Quality of Experience (QoE) requirements of end-users. Despite edge computing potential, in situations where many conditions such as workload intensity may dynamically change, capabilities for auto-scaling applications orchestrated upon such frameworks is essential. Consequently, edge computing application providers should track dynamic changes of operational environments and improve the performance of services offered to end-users. To create an auto-scaling method for handling varied number of requests at runtime without any performance problem, various approaches have been proposed so far. For instance, vertical scaling [1] by resizing resources, e.g. computing power, memory and bandwidth. Or horizontal scaling [2] by adding more needed service instances into the pool of resources in order to share the workload, or possibly removing some of instances.

The significance of a taxonomy for auto-scaling techniques used by cloud-based applications has been discussed in many research works [3, 4, 5]. However, such auto-scaling studies have not addressed applications deployed based on edge computing frameworks which represent a new era of cloud computing. In this paper, modern types of edge computing applications which need auto-scaling behavior have been analyzed. Besides that, a taxonomy of auto-scaling approaches has been proposed that covers edge computing scenarios and their supporting technologies such as container-based visualization. The preliminary goal of this taxonomy is aimed at indicating recent challenges to auto-scaling applications in the context of edge computing frameworks.

The rest of the paper is organized as follows. Section 2 explains modern types of edge computing applications, continuing with the presentation of auto-scaling goals in Section 3. Section 4 presents a taxonomy of auto-scaling approaches that considers edge computing scenarios. In Section 5, we finally conclude this paper.

[1] The ENTICE project, http://www.entice-project.eu/

2. MODERN EDGE COMPUTING APPLICATIONS

Recent years have seen a dramatic growth in the number of devices or end-users connected to the Internet, using cloud-based applications and generating a massive amount of data to be sent and processed on the centralized cloud. Transmitting this huge amount of raw data over the Internet cannot make optimal use of network or centralized cloud infrastructures any more. As shown in Figure 1, in some situations, it is more efficient to process data close to the point of its source by edge nodes, and transmit only less time-sensitive data over the network to remote datacenters for long-term storage and big data analytics.

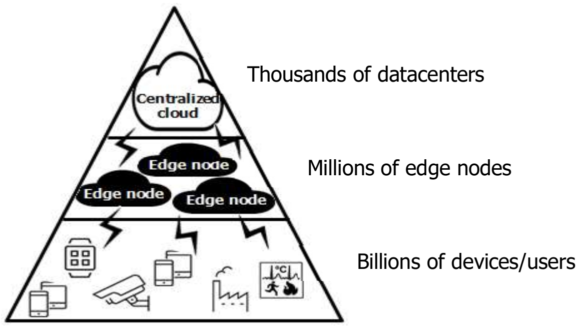

Figure 1. Edge computing framework.

In this section, we analyze a set of modern types of edge computing applications. Such applications have as one of their main challenges the need to implement auto-scaling mechanisms that can support application adaptation during their runtime. Some of these application types are overlapping in scope; nevertheless, it seems important to keep various views on edge computing application types open as much as possible.

2.1 IoT applications

IoT is a paradigm where things, objects and sensors have a pervasive presence in the Internet. The next generation of IoT systems, for example smart cities, would be self-adaptive in terms of scalability which entails the implementation of IoT applications with no human intervention during the operation [6]. They should be able to detect runtime changes of operational environment and determine their own way of reacting to such changes, e.g. drastic increase in the number of connected sensors or in the amount of sensed data. Objects or devices are more widely getting to be connected to IoT applications on the Internet than ever. This fact motivates edge computing paradigm to locally provide an efficient utilization of communication bandwidth and computing resources in each geographic region along with a low-latency response time for IoT applications. Therefore, IoT and edge computing will co-exist, complement, and support each other.

2.2 Microservice applications

Microservices are small, independent, highly decoupled processes which communicate with each other to form complex applications that exploit language-independent APIs. Container virtualization technologies (such as CoreOS, Kubernetes, OpenShift Origin and Docker Swarm) can be seen as enablers of microservices within highly dynamic environments, e.g. edge computing scenarios. Because, it is possible to deploy containerized microservices in different hosting environments faster and more efficiently than using Virtual Machines (VMs). Decomposing a single application into smaller containerized microservices allows application providers to distribute the computational load of services among different resources, e.g. different edge nodes or even geographical locations. In comparison with Service Oriented Architecture (SOA), microservices are usually organized around business priorities and capabilities, and they have the ability of independent deployability [7] and often employ the use of simplified interfaces, such as Representational State Transfer (REST). Resilience to failure could be the next characteristic of microservices. Since in this modern architecture, every request will be separated/translated to various service calls. For instance, a bottleneck in one service brings down only that service and not the entire system. In such a situation, other services will continue to handle requests normally.

2.3 Time-critical applications

Due to the importance of running environments and the distributed nature of time-critical applications such as early warning systems [8], these applications are extremely difficult to be developed and deployed on the cloud. Although cloud environments are able to offer elastic virtualized resources for supporting time-critical applications, cloud technology is not yet a panacea for these types of intricate applications without edge computing architectures. This is because edge computing frameworks allow these types of applications to improve the response time and properly react to the changing conditions in each region compared to traditional centralized cloud architectures. The distinguishing characteristic of edge computing in this concept is its dense geographical deployment of application instances that offloads computations from datacenters and traffic from the core network. However, edge nodes, in practice, may be resources made for specific purposes (e.g. filtering or encrypting data) restricted by their limited scaling capabilities at run-time.

3. GOALS OF AUTO-SCALING ADAPTATION

In this section, objectives which can be achieved by auto-scaling applications are explained. Moreover, key challenges to be considered in order to address these goals within edge computing scenarios are defined.

3.1 To ensure users' needs for QoE

As more and more opportunities emerge to move applications to the edge of the network, service quality will become a significant differentiator between application providers. In particular, QoE, as observed by the customer, has the potential to become the directive role to control service quality [9]. Since QoE is a precise measurement of users' satisfaction with the service, better QoE can bring more customers to the system and more revenue for the application provider. The major challenges to this goal are as follows:

- Edge nodes usually have hardware resource limitations in reality; however they should be able to handle increasing workloads at runtime.
- Different users sharing an edge node in one region have different desires e.g. conflicting desires of speed and security.
- Providing better QoE needs further complicated technologies.

3.2 To meet Service Level Agreements (SLAs) between users and application providers

Nowadays, adaptive adjustment of resource allocation at the edge of the network in order to satisfy SLA requirements should be reached in a convenient way to cover the users' expectations. The aim of SLA management is providing a monitoring framework that can support dynamic adaptation of applications by a trade-off among different QoS metrics. In scenarios where the operational environment (such as the rate of arrival of requests) continuously varies, SLAs can be indispensable to guarantee that service quality is constantly provided at favorable levels in spite of dynamicity of conditions. To reach this goal, the following challenges should be considered:

- Diversity in different types of applications co-located on the same edge node, e.g. some applications are compute-intensive while others may be network-intensive.
- Costly options in SLA can affect user expectations for the performance of edge computing applications.
- Attackers may cause excessive SLA penalties.
- Different users sharing the same edge node have different objectives e.g. conflicting objectives of price and quality.

3.3 To deliver always-on services

Users' key requirement in the context of long-running applications, such as massive data processing, is using highly available services. Given the unstable cloud environments, to provide always-on services, application components can be replicated across geographically distributed infrastructures as one solution [10]. To run indefinitely, long-running services need multiple servers with a required amount of resources to achieve acceptable performance. However, there are different challenges in adapting such systems as described below:

- Bandwidth, storage and computing resources are expensive.
- Organizations have various regulations of using or not using new technologies, for example legislations in the location of data storage.
- Replication of servers within edge computing frameworks has its own technical issues e.g. temporary inconsistencies.

3.4 To provide affordable on-demand solutions

Cost issues and other associated parameters such as storage capacity, computing power and data transfer rate are outstanding challenges for edge-based service providers. Workload monitoring—and hence the task of performing effective resource allocation such as dynamic consolidation of VMs or containers—plays a significant role in the distributed environment e.g. edge computing frameworks. For example, a system may switch off application instances when the number of requests is decreasing. The challenge then becomes an issue of performance maintenance as such a cost optimization could increase the danger that the service provider might break the QoS/QoE promise to the customer [11]. However, the following challenges have become major issues in such scalable edge computing applications:

- Tech-giants may cover more regions and provide better quality with lower price.
- Cost optimization could result in QoS/QoE degradation.
- New efficient technologies are sometimes unaffordable.

4. TAXONOMY OF AUTO-SCALING APPLICATIONS IN EDGE COMPUTING

Figure 2 presents our proposed taxonomy of auto-scaling applications. This new taxonomy particularly covers all adaptation aspects of applications orchestrated within edge computing frameworks.

4.1 Cloud framework

Cloud frameworks to deploy and orchestrate online applications can be considered from two viewpoints: centralized datacenter and edge computing. Furthermore, load-balancing techniques used in each type of framework have been also explained.

4.1.1 Centralized datacenter

Using only one cloud infrastructure to process the workload or store the data is called centralized cloud environment. Centralized cloud solution may increase the risk of service availability and the possibility of security problems [12]. Load balancing which remains still as another open research problem is a method to distribute workload among multiple application instances to decrease response time, increase application throughput, obtain optimal resource usage, and prevent overload on an individual instance.

In the centralized datacenter framework, conventional load-balancing techniques can be used. Load-balancing algorithms within centralized cloud framework which are mainly non user-centric do not need to consider the end-user's conditions such as location, network quality, etc. So far, these types of load balancers support distribution algorithms such as round-robin, weighted round-robin, least-connections, random and so on.

4.1.2 Edge computing

Some types of software systems pose significant requirements such as location-awareness for asset tracking systems or enhanced response time for mobile-based e-learning applications. To this end, in order to satisfy such requirements, a promising trend has evolved from traditional, centralized cloud computing to distributed edge computing frameworks close to users.

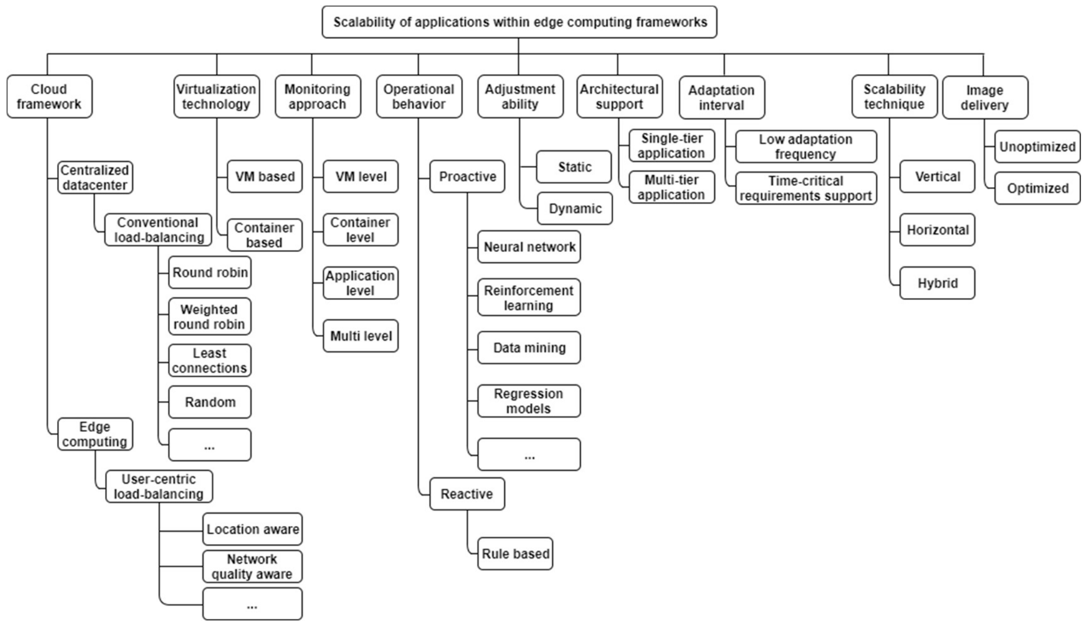

Figure 2. Taxonomy of auto-scaling applications in edge computing.

In the edge computing framework, load-balancing techniques typically should be user-centric. In both industry and academia, the role of intelligent resource scheduling has been considered to be as hard as a Nondeterministic Polynomial (NP) optimization problem [13] which is an NP-hard problem. This issue dramatically grows if the complexity of application and hence the number of monitoring metrics measured in the scalable environment increases. Therefore, load-balancing algorithms within edge computing frameworks which are principally user-centric can suffer from a dimensionality challenge when the size of load-balancing optimization problem grows. Different capabilities for such load-balancing algorithms mainly include location and network quality awareness in general. The location-aware mechanism may customize a user's QoE through dynamically connecting the client to the nearest server based on the location. However, the closest server does not ensure the best possible QoE for the user as network conditions intrinsic to the connection between the nearest server and the end-device is not always guaranteed. According to the network quality-aware approach, dynamic adaptations to the user's network conditions can be accomplished by applying network edge-specific information. For instance, this type of load-balancing approach can contribute towards selecting the best edge node to connect to end-devices in relation to the network quality conditions.

4.2 Virtualization technology

Nowadays, the capability for a software system to dynamically expand or shrink the total number of application instances in the resource pool in order to accommodate varied workloads can be achieved via two types of virtualization technology: VM-based and container-based approaches.

4.2.1 VM based

Hypervisor-based virtualization technologies are able to support standalone VMs which are independent and isolated of the host machine. Each VM instance has its own operating system and a set of libraries, and operates within an emulated environment provided by the hypervisor. If the system uses VMs to run application services, VM-level monitoring becomes mandatory.

4.2.2 Container based

Different from VMs, the utilization of containers does not need an operating system to boot up that has gained growing popularity in the edge computing frameworks. Table 1 provides a comparison between container-based and VM-based virtualization.

Table 1. Container-based vs VM-based virtualization

Feature	**Containers**	**VMs**
Needs	Container engine e.g. Docker	Hypervisor e.g. Xvisor
Weight	Lightweight	Heavyweight
Boot Time	Fast	Slow
Footprint	Smaller	Bigger

Since their nature is lightweight, deployment of containerized services at runtime can be accomplished as dynamic as possible. Consequently, various container management frameworks, e.g. Google Container Engine [14] and Amazon EC2 Container Service [15], have attracted increasing attention to virtualize online applications.

4.3 Monitoring approach

The importance of a monitoring approach fully-configured for an auto-scaling objective has been discussed in research works in various contexts: VM-level, container-level, application-level and multi-level monitoring.

4.3.1 VM level
The purpose of VM-level monitoring is aimed at checking the status of a VM where an application instance is hosted. This type of monitoring provides functionalities to investigate different parameters such as basic availability, memory usage, CPU utilization, disk space capacity and network traffic volume of each VM instance.

4.3.2 Container level
Container-level monitoring is a new research topic within edge computing use cases. If application providers prefer to use container-based virtualization as an alternative to VM-based virtualization to benefit from a lightweight mechanism for deploying, instantiating, migrating and scaling services in the cloud, container-level monitoring becomes compulsory. This type of monitoring system, which undoubtedly supports applications orchestrated within edge computing frameworks, monitors containers and exposes runtime status of key metrics such as memory, CPU, storage and bandwidth usage of each container instance.

4.3.3 Application level
Application-level monitoring is able to check the status of the application performance (e.g. application throughput or response time) at runtime. Application-level monitoring is necessary for achieving a high level of reliability and required for the adaptation actions to provide scalability of services. This type of monitoring dynamically complements application profiling in the context of both cloud and edge computing.

4.3.4 Multi level
In order to come up with a fine-grained auto-scaling mechanism, monitoring systems that are able to collect information from different levels have been more supportive of dynamic adaptation of virtualized applications than single-level monitoring techniques. To this end, in addition to monitoring virtualized resources (e.g. CPU, memory, disk, etc.), considering other levels of monitoring, such as application, is significant.

4.4 Operational behavior
Auto-scaling methods in cloud environments can be categorized into two types: proactive and reactive approaches.

4.4.1 Proactive
Proactive methods [16] are designed to predict the required amount of resources in the near future according to the historical monitoring data such as current intensity of workload, QoS of the application, and assessment of operational cost. Proactive approaches employ learning algorithms such as neural network, reinforcement learning, data mining, regression models and queuing theory to predict the amount of needed resources or the workload. However, it may take too long to converge towards a stable driven model, and hence the application possibly delivers poor performance quality and QoE to the end-users during the beginning stage of learning procedure. Therefore, these approaches can be affected by the limitation that they are not able to handle sudden workload variations in time. Moreover, proactive approaches suffer from inaccuracy which may cause whether over-provisioning or severe performance drops.

4.4.2 Reactive
Reactive approaches [17] are not capable of anticipating future system behavior. Instead, they responsively support dynamic provisioning or de-provisioning of resources when a specific system change is detected, e.g. when the CPU run queues start filling up, and hence an auto-scaling action needs to take place. Time to react on changes in a certain condition is a challenging problem of these methods. Reactive approaches usually depend on a set of thresholds. For example, if the value of a specific metric, e.g. average CPU usage, exceeds a threshold, more CPU capacity will be added to the current system. Ensuring that the predefined values of such thresholds are effectively set has been a challenging issue as they can heavily influence the application performance at runtime. This type of adaptation mechanism is widely used by many cloud-based solutions such as OpenNebula [18] that offers the OneFlow component to allow administrators to specify auto-scaling rules based on monitoring metrics.

4.5 Adjustment ability
One of the main challenges and technical issues in proposing an auto-scaling technique is to what extent the adaptation approach is self-adjustable to changes in the operational environment. In this regard, existing methods are classified into two categories: dynamic and static adaptation approaches.

4.5.1 Static
In a static auto-scaling adaptation approach, the configurations used in the provisioning mechanism are established in advance and these settings are fixed and will not be changed during the execution time. As an example, the scaling policy called "THRESHOLD(X%, Y%)" [19] is a common rule-based provisioning method which adds an instance if the aggregated utilization of an application tier reaches the predefined Y% threshold and switches off an instance when it falls below the predetermined X% threshold for a default number of successive intervals. These static thresholds remain a challenging issue because it tends to make the resource utilization stable on Y% which means 100-Y% resource waste.

4.5.2 Dynamic
On a dynamic basis, an auto-scaling adaptation approach is able to autonomously adapt itself depending on the execution environment status observed by the monitoring system. This capability can be achieved through the dynamic setup of auto-scaling policies or learning algorithms used in the adaptation approach. Generally, dynamic approaches may focus on maintaining the resource usage at 100% utilization without any performance degradation.

4.6 Architectural support
The auto-scaling solutions through which virtualized resources can be autonomously provisioned are explored via two different application architectures: single-tier and multi-tier application.

4.6.1 Single-tier application
In a single-tier architecture, the whole application is composed of only one component to provide the application's functionality such as storage capacity.

4.6.2 Multi-tier application
In a multi-tier architecture, multiple software tiers that are connected to each other and collectively move the whole system towards a common goal represent the application. Therefore, the auto-scaling approach should be able to consider multi-tier aspects of the application such as different dynamic provisioning policies at each tier. However, nowadays a new-fashioned software engineering approach offers a discipline to develop such multi-tier applications based on a set of different loosely coupled independent services running predominantly in containers. Along this line, each service has its own infrastructure and specific application-level parameters to be measured.

4.7 Adaptation interval
Different criteria such as the length of monitoring interval, prediction rate and instance's start-up time can significantly affect the level of auto-scaling adaptation agility.

4.7.1 Low adaptation frequency
In the domain of scalable applications, there is always a trade-off between the monitoring overhead and the length of monitoring time interval. The measurement interval should be defined effectively in a way that the monitoring framework has a low communication overhead and requires little processing power and memory capacity. However, a low level of measurement ratio may lead to missing dynamic changes of operational environments, and hence the system is not capable of adapting to a new situation to continue its operation without any performance issue. Moreover, for the adaptation methods which apply machine learning algorithms, one of the main challenges is the time and computation necessary to reach learning convergence to every new state during runtime execution. Therefore in some cases, it is not possible to have a high rate of adaptation frequency that may affect the agility of the dynamic resource management over time.

4.7.2 Time-critical requirements support
Online applications, such as IoT early warning systems, have time-critical requirements, for example minimal response time, and require appropriate support to provide assured QoS. This is a challenge because performance is difficult to keep up if the execution conditions continuously change. In these scenarios, the adaptation approach should be able to rapidly scale the system capacity at any size. It should be noted that the level of virtualization either based on VMs or containers is a significant factor to be considered. In edge computing scenarios, containers can be used to address the requirements of time-critical applications in a broad range of domains. Container-related technologies have recently achieved an increasing interest as more lightweight, scalable and easily extensible virtualization methods in comparison with VM-based technologies.

4.8 Scalability technique
In case of any deterioration of the application performance due to variations in workload at runtime, there are various auto-scaling mechanisms such as vertical, horizontal and hybrid approaches.

4.8.1 Vertical
Vertical scaling involves resizing resource capacity allocated to the same computing node, for example, adding more memory, bandwidth or processing power to be able to handle an increasing workload. Since maximum resource capacity such as CPU or memory is limited, this technique could be extended to consider also other approaches e.g. horizontal scaling to afford unlimited amount of workload intensity.

4.8.2 Horizontal
Horizontal scaling means the addition or deletion of application instances to the service cluster. Horizontal scaling approach takes more time to be performed than vertical scaling technique. Because, in this case, the instance's start-up time or shutdown time limits should be considered. Therefore, vertical scaling could be more suitable for sudden, temporary peaks in the application workloads.

4.8.3 Hybrid
The combination of both vertical and horizontal scaling techniques is possible in terms of allocated resources. This approach is called hybrid in which both techniques can be applied to the same application in order to take advantages of horizontal and vertical scaling mechanisms.

4.9 Image delivery
Application instances are created typically using provider-specific templates, so-called VM or container images that are stored in repositories. A fast provisioning of application instances with varied workloads ensuring QoS depends on the image repository architecture and placement policies. To this extent, imaged delivery can be either optimized or not.

4.9.1 Unoptimized
VM or container images are currently stored in centralized repositories without considering application features and their requirements at run-time. This fact may cause slow deployment and instantiation overhead. Furthermore, users need to manually manage images' storage which is a cumbersome, time-consuming, error-prone process especially if users work with different cloud providers.

4.9.2 Optimized
In an edge computing solution, image delivery should be optimized that is a significant requirement for auto-scaling. To this end, the ENTICE project provides a solution for the optimized deployment of VM or container images across distributed repositories at different locations. Hence, ENTICE decreases volume of data needed to move, shortens the distance that the data must travel, reduces transmission cost, shrinks latency, improves resource usage and other desired QoS-related characteristics.

5. CONCLUSION

In recent years, edge computing has attracted an increasing amount of attention from both academia and industry. This is primarily due to edge computing framework's capability to process the data at the edge of the network. Consequently, modern applications such as IoT use cases which generate massive amount of data can benefit from edge computing, for example shorter response time, efficient use of resources in each geographic region, location-awareness and so on. However, due to the dynamic environment of edge computing scenarios in which workloads vary over time, such applications should provide turnkey scalability, able to handle large amount of requests sent by end-users and objects (e.g. environmental sensors, smart cameras, vehicles, etc.). Along this line, the paper presents a taxonomy for auto-scaling applications orchestrated within edge computing frameworks. The paper also presented several research challenges that could be considered to enhance the performance of such self-adaptive applications at runtime.

6. ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreements No. 644179 (ENTICE project: dEcentralised repositories for traNsparent and efficienT vIrtual maChine opErations) and No. 643963 (SWITCH project: Software Workbench for Interactive, Time Critical and Highly self-adaptive cloud applications).

7. REFERENCES

[1] Farokhi, S., Jamshidi, P., Lucanin, D., and Brandic, I. 2015. Performance-based Vertical Memory Elasticity. In *Proceedings of 2015 IEEE International Conference on Autonomic Computing (ICAC)*. Washington, 151-152.

[2] Naskos, A., Stachtiari, E., Gounaris, A., Katsaros, P., Tsoumakos, D., Konstantinou, I., and Sioutas, S. 2015. Dependable horizontal scaling based on probabilistic model checking. In *Proceedings of 15th IEEE International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. Shenzhen, China, 31-40.

[3] Chen, T. and Bahsoon, R. 2016. Survey and taxonomy of self-aware and self-adaptive autoscaling systems in the cloud. *arXiv preprint*. arXiv:1609.03590.

[4] Qu, C., Calheiros, R.N., and Buyya, R. 2016. Auto-scaling web applications in clouds: a taxonomy and survey. *ACM Computing Survey*, 1-35.

[5] Alipour, H., Liu, Y., and Hamou-Lhadj, A. 2014. Analyzing auto-scaling issues in cloud environments. In *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering (CASCON '14)*. Ontario, 75-89.

[6] Koprivica, M. 2013. Self-adaptive requirements-aware intelligent things. *International Journal of Internet of Things 2, 1*, 4 pages. DOI:10.5923/j.ijit.20130201.01.

[7] Stubbs, J., Moreira, W., and Dooley, R. 2015. Distributed Systems of Microservices Using Docker and Serfnode. In *Proceedings of the 7th International Workshop on Science Gateways (IWSG)*. IEEE, Budapest, 34-39.

[8] Taherizadeh, S., Jones, A.C., Taylor, I., Zhao, Z., Martin, P. and Stankovski, V. 2016. Runtime network-level monitoring framework in the adaptation of distributed time-critical Cloud applications. In *Proceedings of the 22nd International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*. Las Vegas, USA.

[9] Hobfeld, T., Schatz, R., Varela, M., and Timmerer, C. 2012. Challenges of QoE Management for Cloud Applications. *IEEE Communications Magazine 50, 4*, 28-36.

[10] Kumari, B.K. and Swapna, S. 2015. Stability Based Service for Secure Cloud Storage. *International Jounal of INNOVATIVE Technology 3, 3*, 399-403.

[11] Dib, D. 2014. Optimizing PaaS provider profit under service level agreement constraints. *Ph.D. Dissertation*. Universite Rennes I, Rennes. HAL Id: tel-01124007

[12] Evangeline, M.S. and Prasad, A.S. 2014. Scalable and Secure Multi Cloud Architecture for IaaS to Address the Performance Issues. *International Journal of Computer Applications 105,16*,1-4.

[13] Xu, J., Tang, J., Kwiat, K., Zhang, W., and Xue, G. 2013. Enhancing survivability in virtualized data centers: A service-aware approach. *IEEE Journal on Selected Areas in Communications 31, 12*, 2610-2619.

[14] Google Container Engine, https://cloud.google.com/container-engine/.

[15] Amazon EC2 Container Service, https://aws.amazon.com/ecs/.

[16] Ghobaei-Arani, M., Jabbehdari, S., and Pourmina, M.A. 2017. An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach. *Future Gener. Comput. Syst.*

[17] Al-Sharif, Z. A., Jararweh, Y., Al-Dahoud, A., and Alawneh, L. M. 2016. ACCRS: autonomic based cloud computing resource scaling. *Cluster Computing*, 1-10.

[18] OpenNebula, http://www.opennebula.org/

[19] Gandhi, A., Dube, P., Karve, A., Kochut, A., and Zhang, L. (2014) Adaptive, Model-driven Autoscaling for Cloud Applications. ICAC, Vol. 14, 57-64.