Contents lists available at ScienceDirect

Internet of Things

journal homepage: http://ees.elsevier.com

# Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research

Mohammad S. Aslanpour [a,b], Sukhpal Singh Gill [c,*], Adel N. Toosi [a]

[a] Faculty of Information Technology, Monash University, Clayton Campus, Melbourne, Australia
[b] CSIRO DATA61, Australia
[c] School of Electronic Engineering and Computer Science, Queen Mary University of London, United Kingdom

## Introduction

Cloud computing has emerged as a utility in which a wide range of services from software to platform and infrastructure are offered. By the widespread usage of cloud services, diverse orchestration concerns are rising. For instance, it is outlined in [1] that the typical web applications are facing a growing demand during the working hours, with a severe variability at around noon, and a decreasing demand during the night to the morning. If such variability is not well controlled by a dynamic resource provisioning, the users may suffer from long latencies to receive the responses. Orchestration techniques include, but not limited to, resource allocation, resource provisioning, task scheduling, load balancing, Virtual Machine (VM) migration, task placement, server consolidation, service composition, data caching, service discovery, etc.

The presence of Internet of Things (IoT) demands even more careful consideration of orchestrations due to latency-sensitivity of many IoT applications (or services) and limitations specific to IoT. To reduce overall delay for IoT application and address their latency requirement, Edge and fog computing (hereafter, fog and edge) bring computing resources close to users and IoT devices and offer computation at the edge of network. In consequence, fog and edge not only minimize latency, but also address privacy, data locality, bandwidth consumption of IoT applications. While fog and edge are used interchangeably, we differentiate them in this paper. The former is composed of physical servers closer than the cloud and inherits cloud characteristics. The latter is closer to IoT devices and user, mostly inheriting IoT devices' characteristics. With fog and edge, orchestration is facing more challenges and diversities such as limited energy power and ephemeral resources. Challenges are increasing due to the added layers to the computation and the diversities are furthered as edge-specific techniques such as task offloading and resource sharing are appeared in such distributed environments.

### *Motivation and our contributions*

Orchestration development efforts are required to fulfill certain objective(s). The literature reported that several objectives such as latency minimization, cost reduction, energy management are targeted by the existing techniques [2-5]. For instance, the authors in [6] attempt to minimize the latency while reducing the cost for gaming applications through a dynamic resource provisioning technique. Several of such a technique have been developed for latency-sensitive applications, but targeting energy as an objective [7, 8]. Not only the objectives, but also the stakeholders of the service are diverse. One technique may target cloud, fog or edge providers to fulfill their requirements such as energy consumption while another may target users' requirements such as lower response time. Objectives fulfilling the users' requirements are also known as Service Level Objectives (SLO) which come under the umbrella of Quality of Service (QoS). Service Level Agreement (SLA) which acts as an official contract between users and providers is based on the same objectives. Hence, objectives and considered stakeholders for orchestration techniques must be considered before moving to development of the solution and then production environments.

Given such complexity, this question is raised that how the performance of orchestration techniques can be evaluated to ensure the accomplishment of their mission? To answer, it is necessary to evaluate the performance of the proposed technique using certain metrics which are called performance evaluation metrics (or shortly metrics). Measuring the capability and accuracy of the techniques becomes even more challenging when the adopted technique is multi-objective by which not only more measurements are required, but also the objectives may conflict with each other. Imagine that a resource allocation technique is claimed to be cost-effi-

* Corresponding author.
*E-mail addresses:* mohammad.aslanpour@monash.edu (M.S. Aslanpour); s.s.gill@qmul.ac.uk (S.S. Gill); adel.n.toosi@monash.edu (A.N. Toosi)

https://doi.org/10.1016/j.iot.2020.100273
2542-6605/

cient and keeps the number of resources at minimum, thus maximizing the latency and violating the SLA. This is further important when violating the SLA (e.g., users' requirements) can impose penalties for the provider. Resources which host a service can be a physical host/server, VM, container, etc. (hereafter, host). Motivated by this, it is apparent that, regardless of which orchestration technique is employed, identifying the right metrics for evaluating the performance and also selecting the most suitable metrics are essential.

Performance evaluation (or measurement) is an inseparable process for optimization and cloud to things (i.e., cloud, fog, edge and IoT) orchestration techniques are not an exception. Performance evaluation highlights the weaknesses and benefits of different approaches by which the developer can ensure if the technique is fulfilling the objectives. Here, certain Research Questions (RQ) are raised which we aim to answer in this paper:

***RQ1:*** What are the potential performance metrics applicable to the orchestration techniques in cloud, fog and edge computing?

***RQ2:*** How are the potential performance metrics implemented and measured in cloud to things?

**RQ3:** What are the challenges and open issues for future performance evaluation metrics?

We have conducted a review to identify and categorized the potential metrics, which can stand as a standard benchmark for the future research. Identified metrics are defined, elaborated and the measurement thereof is outlined, as well as touching their considerations in the literature. To make the review article further understandable, the identified performance metrics are categorized according to the well-known concept as MAPE-K (monitoring, analyzing, planning, and execution) introduced by IBM [6, 9, 10]. This effort helps cloud, fog and edge computing research community to select the right performance metrics to evaluate orchestration techniques. Further, they will be able to re-use and/or reevaluate the existing cloud-native metrics in the fog and edge as promising computing landscape for the future of IoT [11].

*Article structure*

The rest of the paper is organized as follows: In Section 2, a taxonomy of performance metrics is discussed to answer the RQ1 and the identified metrics are individually outlined in details to answer to the RQ2 in Sections 3 to 6. The discussion on the performance evaluations metrics, exploring challenges and open issues are highlighted in Section 7 to answer the RQ3. Finally, Section 8 concludes the paper.

## Performance evaluation metrics for cloud, fog and edge computing: a taxonomy

In this section, a generic model for cloud to things landscape is illustrated. The positioning of the optimization techniques in the model is depicted. Then, a taxonomy of measurable performance metrics for each layer of computing is presented. The cloud layer is divided to clarify different cloud models' requirements. Lastly, a brief definition thereof is provided.

*Computing model*

A generic model for cloud to things landscape is pictured in Fig. 1. According to the research community efforts and industrial advances, a fully-implemented real-world landscape for IoT applications would consists of four layers: (1) IoT layer,[1] (2) Edge Computing layer, (3) Fog Computing layer and (4) Cloud Computing layer. A MAPE-K loop can perfectly optimize the whole orchestration process whose relation to the landscapes is shown in the proposed model. This is a generic model and MAPE-K loop can alternatively be separately implemented for each layer as well.

*Mist computing layer*

or extreme edge, is formed when IoT devices are computation-enabled and can cooperate with each other akin to a mesh network. Raspberry-Pis are perfect examples of such devices which can provide computation as well as typical functionalities of IoT devices such as sensing and actuating. This augmented capability enables IoT devices to constitute a mesh-like network which can be managed either centrally by a nominated controller or distributed. Mist computing can solely be implemented without any communication with fog, edge or cloud.

*Edge computing layer*

is where gateways such as dedicated routers, switches or base stations located close to IoT devices to act as the gateway to the fog or cloud layer. The aim of this proximity is to minimize and manage the traffic. Edge can also be equipped with micro data centers, or so-called cloudlets, to bring the computation closer to the IoT devices as well as being a gateway. In this case, not only the traffic is minimized, but also the micro data centers can gather the sensed data of IoT, filter, and only send the reduced analyzed data to the fog or cloud for the sake of bandwidth preservation.

*Fog computing layer*

is benefiting from computations closer than cloud to the IoT layer and rather more powerful than that of edge layer, e.g. micro data centers. Fog is an intermediary layer to alleviate edge and cloud limitations.

[1] If computation happens on the IoT devices, this layer can be also called Mist computing which typically consists of micro-controllers and sensors.

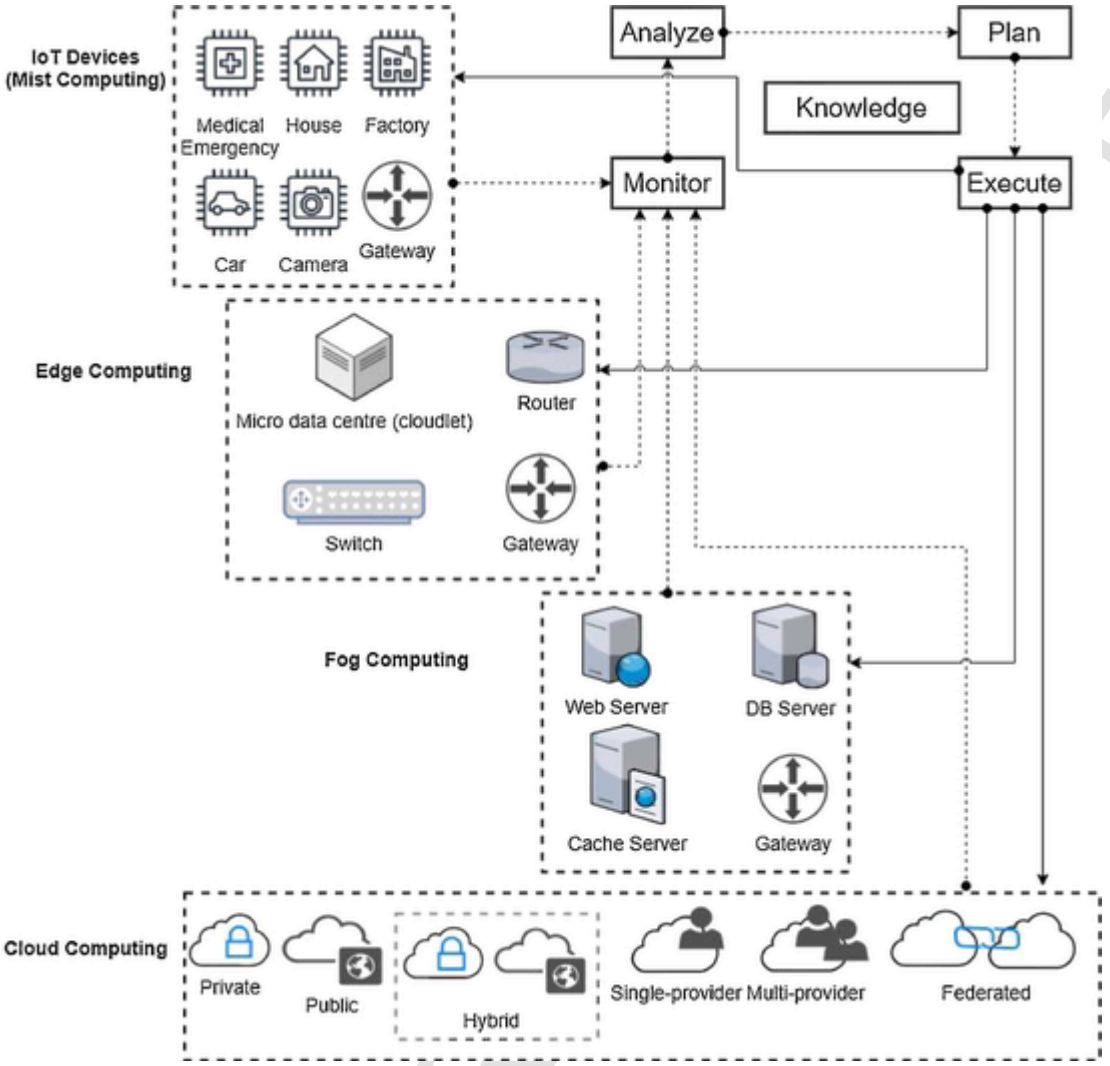



**Fig. 1.** Cloud to Things landscape utilizing MAPE-K loop to collect performance metrics and perform orchestration, for example, load balancing, task migration etc. .

*Cloud computing layer*

is acting as the backbone and providing persistence data storage as well as powerful and unlimited computation resources which are not available in mist, edge and fog layers. For example, assume that the IoT devices like thermometers are periodically measuring the temperature in a farm and sending it to persistent storage to be evaluated later. In such applications, edge and fog may not be reliable in terms of persistent storage. Moreover, latency is not a matter of concern for this application. Hence, cloud storage is preferred as it also is usually cheaper than edge and fog computing. Cloud computing layer can itself be categorized as: private, public, hybrid, single-provider, multi-provider and federated cloud. Each model of cloud may demand distinguished performance measurements to be considered. For instance, preserving SLA in terms of response time is more difficult in federated clouds where each provider may have its own SLA contracts.

*MAPE-K loop*

is a standard orchestration process which can nominate almost all orchestration optimization techniques such as auto-scaling or task scheduling. In this process, a *monitor* is responsible for collecting observed metrics ranging from application-level (e.g., user's requests or SLA) to system-level (e.g., optimization technique overhead) and hardware-level (e.g., CPU utilization). This is an ongoing process which updates the latest status of the whole system for the analyzing phase. The *analyzer* is responsible for enriching the collected data which can be augmented by Machine Learning methods. Then comes to the *planner* which must decide about the adaptation of the application and underlying resources, given the optimization technique. For example, the technique can be load balancing by which the load balancer decides on where to offload the incoming task, either to another IoT device or to edge, fog or cloud lay-

ers. Finally, the *executor* is authorized to perform the decision which can employ the network access to handle such actions. All the monitored data and activities of such components in MAPE-K are kept in a knowledge repository for upcoming decisions.

*Taxonomy*

Performance metrics heavily depend on the computing model/layer wherein the IoT application is orchestrated. For instance, in a mist model the availability of IoT devices to handle the offloaded tasks is a major concern while in cloud model/layer the availability is guaranteed. Instead, in the cloud model, evaluating the response time is critical. Both availability and response time are critical in not only Mist and cloud, but also in edge and fog. However, the degree at which each performance metric is considered is different. Moreover, as new computing paradigms such as fog and edge inherit certain characteristics of cloud they may share similarities in terms of possible performance metrics. Noticeably, certain metrics are naturally common in all layers but have particular requirements, such as resource utilization which can be measured in terms of containers, VMs or a hybrid of both depending on the corresponding computing layer. Such metrics are classified separately for each layer. Motivated by that, a taxonomy of identified performance metrics for the whole computing landscape is provided in Fig. 2. This taxonomy clarifies measurable metrics specific to each computing layer as well as those metrics which are commonly measurable for all layers. Then the cloud layer is dissected to identify measurable performance metrics for each model of cloud separately.

Cloud models, although sharing certain similarities, differ in particular characteristics. Hence, a taxonomy of performance metrics for cloud models is also essential to consider (see Fig. 3). According to specifics of each model of cloud, the potential performance metrics are scattered in the taxonomy. Some metrics are worth investigating only for particular models of cloud such as privacy which is critical for private clouds. In private clouds, confidential data related to an organization, for instance, is travelling be-


**Fig. 2.** A taxonomy of real-world performance metrics for evaluating IoT/Mist, Edge, Fog and Cloud Computing.

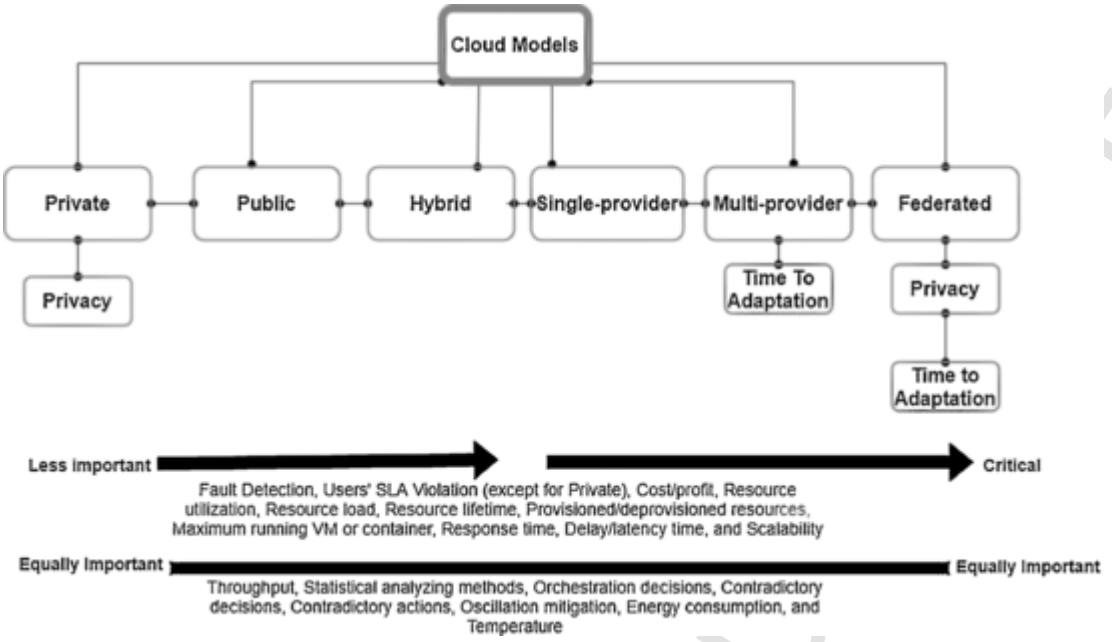



**Fig. 3.** Taxonomy of Metrics based on Cloud Models.

tween authorized users which demands avoiding data leakage. Additionally, in federated clouds, the privacy comes to play where multiple vendors are collaborating with each other with possibly distinguished privacy policies which can conflict. On the other hand, there are certain metrics which are cloud-specific and are equally important for any cloud model. Such metrics include: Throughput; Statistical analyzing methods; the number of Orchestration decisions, Contradictory decisions, Contradictory actions, and Oscillation mitigations; Energy consumption; and Temperature. However, there are other metrics which are essential to be evaluated for all cloud models, but with different degrees of importance. Such metrics include: Fault Detection, the number of Users' SLA Violation (except for Private), Cost/profit, Resource utilization, Resource load, Resource lifetime, Provisioned/deprovisioned resources, Maximum running VMs or containers, Response time, Delay/latency time and Scalability. The importance of those metrics becomes more and more when employing hybrid, multi-provider and federated models compared to private, public, and single-provider models. All metrics will be defined and explained in the following sections.

*Definitions*

A classification of identified metrics according to their measurable phase in MAPE-K is demonstrated in Fig. 4.

*Monitoring-related metrics*

Monitoring phase is where the performance of the whole system as underlying infrastructure to users can be watched. A well-known example is CloudWatch provided by AWS.

- *Resource Utilization:* It is defined as the percentage of resources which is consumed by the incoming workload. In other words, it shoes how busy the CPU is. Resources could be CPU, RAM, Memory, Bandwidth, etc.
- *Resource Load:* It is defined as the percentage of load on a physical host, VM, container, etc. It is a measurement of the number of tasks waiting on the CPU queue to be executed as well as those which are running at the moment. The above-mentioned CPU utilization referred to only the latter.
- *Throughput:* It is defined as the ratio of the number of arrival tasks to the processed tasks in a period of time.
- *VM Lifetime:* It is defined as the time the VM is rented.
- *Maximum Running Resource:* It is defined as the maximum number of resources being run.
- *Response time:* It is defined as the time taken between sending a request to the server and completion of the task execution.
- *Delay time (or Latency):* It is defined as the different between the actual completion time of a task and its desired completion time.
- *Network Congestion/Traffic Control:* It is defined as the incoming load that can be indicated by the number of incoming requests to it during the time.
- *The number of Damaged Tasks:* It is defined as the number of requests that did not succeed in receiving answer or at least timely answer.
- *SLA Violation:* It is defined as the number of or the percentage of tasks that experienced a delay time of more than what have been conceded. Further parameters can also be included in the SLA such as availability of resources.

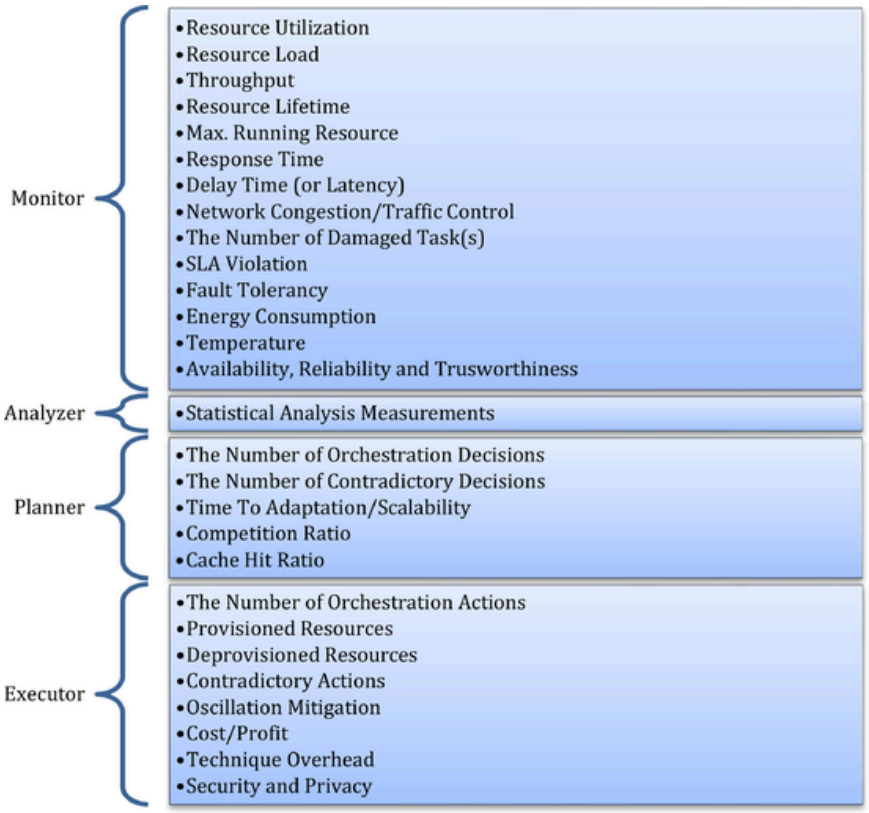



**Fig. 4.** Classifying the identified metrics according to MAPE-K loop.

- *Fault Tolerance:* It is defined as is the ratio of the number of faults detected to the total number of faults existing. Faults may be due to software or hardware [9].
- *Energy Consumption:* It is defined as the amount of energy consumed by a resource to finish the execution of workload [9].
- *Temperature:* It is defined as the degree or strength of heat generated by underlying infrastructure as data center when executing tasks [12].
- *Reliability:* It is defined as the ability of the mechanism to perform well under certain circumstances, even with occasional problems [9].
- *Availability:* It is defined as the existence of the system at any time of issuing a request by users [9].
- *Trustworthiness*: the degree at which the optimization technique can handle untrusted devices.

*Analyzing-related metrics*

***Analyzing*** is performed to make a prediction out of the monitored parameters such as response time, resource utilization etc. Metrics available here are mostly those considering series of observations like time series which can be analyzed (or predicted) by methods such as machine learning. In the literature, Auto Regressive Integrated Moving Average (ARIMA), Artificial Neural Network, Single Exponential Smoothing, Double Exponential Smoothing, etc. are of widely used forecasting methods. In this regard, the optimization technique's performance in analyzing this phase is evaluated by calculating statistical criteria such as Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), Decision Feedback Predictor (PRED) [11].

- *Statistical Analysis Measurements*: It comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data.

*Planning-related metrics*

***Planning*** is the phase in which the optimization decision such as VM placement or VM migration is announced.

- *The number of orchestration decisions*: It is defined as the total number of orchestration decisions such as scaling or offloading made by planner.
- *Contradictory decisions*: It is defined as how many times the already made decisions are reverted due to inaccurate decisions.
- *Time to Adaptation/Scalability*: It is defined as time taken to reach a stable and healthy situation after applying the decisions. *Scalability* comes under the umbrella of adaptation time where it shows the potential the ability of the orchestration to shrink out and

in respond to the workload. This can be seen either in adding the number of resources in a cloud landscape or in adding IoT devices to the network in IoT landscape.

- *Competition Ratio*: It is defined as the ratio in which the resources compete for taking a request or vice-versa.
- *Cache Hit Ratio*: It is defined as the success of a resource or service caching mechanism in reducing the transmission of data across the network.

*Execution-related metrics*

***Executing*** phase is the final phase, in which orchestration actions are actually performed.

- Orchestration Actions: It is defined as the frequency of performing the adopted decisions made by a planner.
- *Provisioned Resources*: It is defined as the number of resources (e.g., VM, container, etc.) provisioned throughout the run-time.
- *Deprovisioned Resources: It is defined as the number of resources (*e.g.*, VM, container, etc.) deprovisioned throughout the run-time.*
- *Contradictory actions*: It is defined as how many times the already performed actions are reverted due to inaccuracy.
- *Oscillation mitigations*: It is defined as the resistance to avoid fluctuation in decisions and actions.
- *Cost*: It is defined as the cost (e.g., monetary or complexity) imposed to user or provider as a result of the orchestration technique performance during the run-time.
- *Profit*: It is defined as the money that is saved by the efficient performance of the orchestration technique.
- *Technique's Overhead/Lightness*: It is defined as the overhead that the orchestration can impose to the system. In terms of lightness, it can be defined as the time the orchestration takes to be up and ready.
- *Security and privacy:* It is defined as the degree at which the orchestration can keep both data and services away from malicious activities [9].

**Monitor-related metrics explanation**

*Resource utilization*

Resource utilization represents the intensity of load on a VM, for instance, and can be indicated by percentage, for instance 78%. VM utilization is the proportion of resource capacity, e.g. CPU; that is being or has been used to execute tasks. It also describes the demand on the resource and shows whether the utilization level is low or high [13] [14]. Under another definition, resource utilization is a ratio of actual time spent by resource to execute workload [9]. In cloud scale, utilization is the total amount of resources actually consumed in the data centers. The objective is to utilize the resources effectively is to maximize the resource provider revenue and profit while keeping users' satisfaction [13, 15]. Note that it is not the case that the higher average CPU utilization, the more efficient the orchestration is. A high level of CPU utilization might be a repercussion of load accumulation on other resources, stemming from inefficient load or resource management [2]. It is asserted that resource utilization has a linear relationship with energy consumption [10]. The following formula is proposed to calculate resource utilization: Resource Utilization = *actual time spent by resource to execute workload/total uptime of resource* [9]. CPU is just an example of resources and RAM, storage, and even bandwidth can be considered and evaluated in terms of utilization level.

*Resource load*

Resource load measurement demonstrates the load or the number of tasks that a host has to execute, so this metric can be calculated qualitatively or quantitatively. Put it another way, this measures the number of tasks running on the VM and waiting in queue of the VM. This process is handled by task scheduler. The range of the value for this can experience higher than 100% because of inefficient scaling techniques.

*Throughout*

Throughout means the ability of orchestration to assign hosts so that they can execute tasks timely [9]. This metric is calculated by dividing the number of executed tasks into arrived tasks in a period of time. Under another definition, throughout is a total amount of tasks that are executed successfully within given time period in cloud computing [16]. This metric can give an overall view of the performance of the system, but cannot guarantee the lower response time and cost.

*Resource lifetime*

Resource lifetime is a metric to calculate the time a VM have been alive, in minute, hour, etc. Meaning that the time a VM has been rented [17]. Higher lifetime, while lower renting cost, is the main goal of scaling techniques since having more stable VMs results users in less waiting time for VMs being launched.

*Maximum running resources*

The maximum number of hosts rented using the orchestration mechanism evidences how much it is resistant against heavy and fluctuated workload in peak of load, especially when facing flash crowd. However, adopting an extremely conservative method can cause SLA violation. Moreover, as cloud providers provide three types of VMs, on-demand, reserved and spot, this metric can be assigned to calculate them separately [11]. Alternatively, in IoT/Mist, the wider range of nodes may be employed.

*Response time*

Response time indicates the time from arrive a task at load admission to returning corresponding answer to user. Response time is also called execution time is time required to execute the workload completely [9]. It is also known as completion time, which is required for the specific cloudlets or tasks to complete the job [13]. It is a ratio of difference between workload finish time and workload execution start time to number of workloads [9]. This metric is more effective in evaluating performance testing, productivity applications, and graphics oriented workloads [9]. Also, execution time is calculated similar to delay time [18]. The detailed description of metrics for that can be found in [19].

*Delay time*

Delay time is the unpleasant period of time a task is waiting to be executed. For instance, if expected response time of a task is 2 s and actual response time is 3 s, the delay time of this task is 3–2 = 1 s. These metrics work for situation in which the actual response time for the given task is already measured. Delay time and latency time are being used interchangeably [14].

*Network congestion/traffic control*

The load produced by end-users in a period of time is a metric indicating how busy VMs have been and can show the end-users' behavior during the time. Precisely, when a user' request to the application is sent, it will be received by load admission responsible for monitoring workload traffic. Load admission counts the number of arrived tasks. Workload is the amount of processing to be done or handled within given time period. In simple, it is the ability to handle or process work in cloud computing [20].

*Damaged tasks*

This damage can be divided into two types: canceled tasks and failed tasks. Task cancelation, for instance, occurs when resource provisioner decides on releasing a host having one or some tasks on it. In this situation, the tasks are canceled and moved to load balancing for finding another host. There is another metric as failed task that shows how many tasks has been failed and have not produced any answer to user. This happens when the host which is running the task immediately crashes or the task reaches its timeout. In the Web, the Web requests have a timeout that if they cannot be executed during this period, they have to be failed. Task failing is inevitable when flash crowd occurs and auto-scaler is not well-defined for such situations. The number of missed deadlines was calculated by [18].

*SLA violation*

SLA is an official and signed contract between service provider and end-users. SLA violation means the number of times this contract is violated by service provider. A main element of SLA is delay time. In this case, if SLA contract for delay time is defined 1 s and a task is executed in 1.3 s, it is called a SLA violation. Every cloud provider wants to deliver their best services to fulfill the requirement of the cloud user and avoid the SLA violence [13]. SLA violation rate is possibility of defilement of SLA [9]. This metric is more effective in evaluating performance testing workloads [9]. SLA violation for edge computing and particularly for IoT applications is not well-defined yet [21].

*Fault tolerance*

It is the ratio of number of faults detected to the total number of faults existing. Faults may be due to software or hardware. Faults may be software or hardware. The following formula is used generally for the calculation: Fault Tolerance Rate = Number of Faults Detected/Total number of Faults [9]. Fault detection rate expected to decrease with increase in number of workloads.

*Energy consumption*

Energy is amount of energy consumed by a resource to finish the execution of workload. With the increasing number of resources and energy-constrained IoT devices, the value of energy consumption also increases [9]. With this in mind, the term energy-efficiency is widely-used for optimizations for this metric, which is a ratio of number of workloads successfully executed in a data center to to-

tal energy consumed to execute those workloads [10]. This metric is more effective in evaluating performance testing workloads [9]. Energy-aware autonomic resource scheduling (EARTH) is an autonomic resource management technique which schedules the resources automatically by optimizing energy consumption and resource utilization [22]. The detailed description of metrics for sustainable and energy-aware cloud computing can be found in [5, 23].

*Temperature*

It is a degree or strength of heat present or generate in the computing environment. It refers to the heat generation in data center when tasks are executing on the underlying infrastructure which mostly implies to cloud infrastructures, having powerful computation resources, not energy-constrained IoT devices or edge nodes [24, 25]. The detailed description of metrics for thermal-aware cloud computing can be found in [26].

*Reliability*

Reliability of nodes in virtual environment is changing adaptively. Node is reliable only if the node can perform well under uncertain situations such as failure in particular functions, otherwise node is not reliable [9]. The following formula can be utilized to calculate reliability: Reliability = MTBF + MTTR, where Mean Time Between Failures (MTBF) is ratio of total uptime to number of breakdowns and Mean time between failures (MTBF) = Total Uptime/ Number of Breakdowns. Mean Time To Repair (MTTR) is ratio of total downtime to number of breakdowns: MTTR = Total Downtime/Number of Breakdowns This metric is more effective in evaluating Websites, storage and backup services and mobile computing services [9]. This metrics is further critical in edge layer where the churn nodes exist. Churn nodes are those hosts which continuously can join and leave the network [27].

*Availability*

It is an ability of a system to ensure the data is available with desired level of performance in normal as well as in fatal situations excluding scheduled downtime. It is defined as the ratio of MTBF to Reliability. Practical formulizations are proposed in [9, 13]. This metric is more effective in evaluating websites, endeavor software, online transactional processing, central financial services, critical Internet applications and mobile computing services [9]. Under another explanation, it is the combination of resource's accessibility, maintainability, reliability, security and serviceability [13]. The detailed description of metrics for reliability and availability can be found in [9].

*Trustworthiness*

This is highly the case for mist computing in which IoT nodes would leave the network without any notice despite identifying themselves as a trusted device. Such occurrences can confuse the orchestrator to whether rely on the device or not in order for offloading tasks. As the future of IoT, the Social IoT (SIoT) requires critical considerations of this metrics, wherein the hosts cannot be trustable.

**Analyzer-related metrics explanation**

*Statistical analysis methods*

Analyzer duty is to analyze monitored parameters and is in charge of producing more accurate values from monitored parameters to be used by the planner. At this phase, predictions methods and machine learning techniques are mostly used. Therefore, the most relevant metrics that can be identified here are statistical methods. Precisely, to evaluate the accuracy of predictions, metrics like MAPE, MAE, MSE, $R^2$, Root Mean Square Error (RMSE), Average, Median, Tail, PRED [11] are widely used [5]. With the advances in Deep Learning and the idea of Federated Machine Learning at the edge, conventual metrics require to be reexamined in the context of edge computing [28].

**Planner-related metrics explanation**

*Orchestration decision*

This is an effective metric to understand how many decisions have been made by orchestrator. Certainly, if the number for decisions is low, it has been more confident in decision makings. The higher the number of decisions is, the more overhead on the system will be. On the other hand, a higher number of decisions may imply that the decision maker is conscious of the changes in the services and hosts situations.

*Contradictory decisions*

Another metric in planner phase is measuring *contradictory decisions*, showing the number of times an orchestrator works confusingly. Take autonomic resource provisioning as an example. If the provisioner makes two distinguish decisions (provisioning and deprovisioning) in two subsequent epochs, this is called contradictory. Lower value for this metric indicates that decision maker is working efficiently when facing fluctuation in workload.

*Adaptation time and scalability*

The time taken from planning (e.g., VM placement) to establishing the adaptation of the service is called adaptation time. That is to say, when SLA is violated, we call this situation maladaptation of service, requiring adaptation decisions. Precisely, the time an orchestration decision is announced to the revival of SLA status is called time to adaptation. Efficient orchestrators reduce time to adaptation for services.

Scalability is another metric in relation to the planner performance evaluation. Scalability measures how well the orchestrator is grown and shrunk in response to the service demand and available hosts. This differs from application scalability [14]. It is possible to name it a measurement of the reconfiguration overhead of the infrastructure [14].

*Competition ratio*

In certain situations, when deciding to modify the underlying hosts in order to optimize an object, the hosts may compete to acquire the demanded resources. This is important for an orchestration to deal with such subsequences. Measuring the performance of orchestrator in dealing with such situations is equally important. In [29], the competition ratio for evaluating task scheduling techniques in edge computing is measured.

**Executor-related metrics explanation**

*Orchestration actions*

This metric is measured to evaluate the overhead of orchestrator's actions, meaning that if the orchestrator is capable of handling the fluctuations in load while mitigating unnecessary actions. It can be also a measure of accuracy in mitigating fault decisions made by planner. For instance, a faulty task offloading decision of planner can be rejected by executor if it perceives that destination host is not able to handle additional services.

*Provisioned and de-provisioned resources*

Provisioned resources metric shows the number of hosts requested from the provider, whether cloud or edge, by executor. De-provisioned resources metric indicates the opposite action. This is a major metric when it comes to auto-scaling techniques and is widely-used in the literature [6, 31-34]

*Contradictory actions*

Contradictory actions refer to measuring the number of times in which two different actions (e.g., scale-up and scale-down) are performed subsequently by executor. This misbehavior of executor is because of being unaware of its past actions and fluctuation in workload.

*Oscillation mitigation*

If one wants to only measure the ability of orchestrator to reduce contradictory actions, oscillation mitigation metric is the answer. This metric indicates the extent at which an orchestrator manages to avoid contradictory actions which is combination of both undesirable effects. Oscillation occurs when orchestration actions are carried out too quickly, before being able to see the impact on each action on the application [4]. In resource allocation techniques, for instance, a cooldown time is defined after each action to mitigate oscillation [2].

*Cost and profit*

Cost in cloud [13] is calculated based on the renting cost that the orchestrator imposes. In cloud, renting cost is calculated in a pay-as-you-go mode. In addition to renting cost, it is common to define SLA violation penalty that can be added to renting cost. Alternatively, some researchers calculate the obtained profit [9]. Cost is an amount that has to be paid against the usage of resource in cloud computing. It is profit and revenue for the cloud providers and expense for the cloud users [13]. The cost and pricing are as-

pects that are often associated with public clouds due to their resource acquisition nature. The definition of a way to price the use of resources in the cloud is not yet standardized, and each service provider has usually its own strategy. Total costs, infrastructure costs, and costs per hour are examples identified in this review [14]. There is always a cost related to the utilization of some elasticity mechanism. This cost could be a financial cost or an operational cost [14]. The detailed description of metrics for cost can be found in [9]. Measuring cost and profit in fog may be feasible due to its cloud-inherited characteristics, but in edge and IoT/Mist it needs further explorations.

*Technique overhead or lightness*

Regardless of certain optimizations in the system, the orchestrator should itself be light and easy to use. This is an applicable metric in real-world experiments which demands efficient development of orchestrators. An orchestrator which requires only experts in software engineering to be deployed or requires a great amount of resources to be hosted will not be applicable in production environments. Assume that a task offloading mechanism for IoT/mist computing which is supposed to be hosted on energy and computation-constrained devices such as Raspberry Pis. If the orchestrators need multiple CPU cores and demands high memory footprints, it cannot be practically deployed. Hence, the lightness of the orchestrator is a vital metric to be considered.

*Security and privacy*

Security is ability to protect the data stored on cloud by using data encryption and access controls [9, 35]. This metric is more effective in evaluating endeavor software, online transaction processing, central financial services and productivity applications [9]. The detailed description of metrics for security can be found in [35].

## Discussions, challenges and future directions

*Discussions*

Table 1 depicts how the aforementioned metrics are utilized in the literature, which could help the researchers to relate each research work to a particular metric. While there exist hundreds of published works on optimizations for cloud to things computing, the paper selection policy for this table considers highly cited papers mostly to explore a wider range of research efforts. The spread of evaluated metrics across the table does not give an overall view of the popularity trend for particular metrics. Overall, the resource utilization, response time, energy consumption and cost are of paramount important metrics accordingly and are the most commonly evaluated metrics in the literature. Following the MAPE-k loop, as expected, those metrics which are measurable through the monitor are regularly considered as the first-class citizens of performance evaluations. Analyzer-related metrics which are essentially statistical metrics are only considered in a narrow range of attempts. The planner-related metrics stand at the middle-level of importance. While such metrics like contradictory decisions can highly influence the performance of an orchestration, the next phase executor can be more influential and reverted incorrect decisions. The executor-related metrics such as cost are well-investigated in cloud-related works, but their real consideration in edge and fog computing is a not clear. This is mostly due to the difficulty in implementing cost measurements in such a dynamic and federated environment. Hence, monitor and executor are the home to measure the majority of metrics due to their connections to the outside world such as sensors and actuators, respectively.

In terms of landscape-specific metrics, it is obvious that particular metrics can be more critical for edge and fog than cloud. Metrics such as availability, reliability and trustworthiness, and security and privacy can of course be evaluated for cloud-specific optimizations, but the scalability of cloud and secure connections to the cloud services are becoming more and more acceptable by the advances in communication technologies and unlimited provided resources. In contrary, the unstable and mobile nature of edge nodes demands careful evaluation of optimizations in terms of the aforementioned metrics.

Further critical analysis on the literature is provided in Table 2 where the effort they have proposed and the potential gaps for the corresponding work are demonstrated. For instance, in [36] a survey on edge computing is conducted while the main focus is on mono-objective orchestrations; In [6], resource provisioning is proposed to preserve the QoS by reducing the average of response time. However, while their focus was on cloud landscape, such solutions should consider the tail latency when they want to bring such functionalities in to the edge. In [9], the orchestration is performed based on monolithic applications and considered always-on hosts while realizing such efficiencies in edge is not feasible. In edge, the hosts are energy and computation-constrained and cannot tolerate always-on services.

*Challenges*

*Federated sla view*

Is solely optimizing particular metrics sufficient? The short answer is no even though an optimization as a VM migration can reduce the power consumption significantly. Applicable solutions must consider the optimization from different and conflicting aspects to confirm the effectiveness. SLA is the first and foremost criteria to guarantee the effective performance of optimization techniques in cloud to things landscape. Service or resource providers may claim that their product is, for instance, available 99.99%, but there is no guarantee for that. Similarly, an optimization technique such as task placement may advertise itself as energy-aware, but

**Table 1**
Mapping of Proposed Taxonomy with Exiting Literature of Cloud, Fog and Edge Computing.

| Performance Evaluation Metrics | **Study** | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [4] | [18] | [14] | [37] | [1] | [36] | [9] | [13] | [38] | [12] | [26] | [39] | [35] | [19] | [40] | [41] | [42] | [2] | [43] |
| **Resource Utilization** | √ | √ | √ | | √ | √ | √ | √ | √ | √ | | | | | | | √ | √ | |
| **Resource Load** | √ | | | | | | | | | | | | | | | | | √ | |
| **Throughput** | | | √ | | | | √ | √ | √ | √ | | | | | | | | | |
| **Resource Lifetime** | | | | | | | | | | | | | | | | | | √ | |
| **Max. Running Resource** | | | | | | | | | | | | | | | | | | √ | |
| **Response Time** | √ | √ | √ | √ | | | √ | √ | √ | √ | | | | √ | | | √ | √ | |
| **Delay Time (or Latency)** | | | | | | √ | | | | | | | | √ | √ | | | √ | |
| **Network Congestion/Traffic Control** | | | | | | | | | | | | | | | | | | | |
| **Damaged Tasks** | | √ | | | | | √ | | | √ | | | | | | | √ | √ | |
| **SLA Violation** | | | | √ | | | √ | | | √ | √ | √ | | | | | | √ | √ |
| **Fault Tolerance** | | | | | | | √ | | | √ | | | | | | | | | |
| **Energy Consumption** | | | √ | | √ | | √ | √ | | √ | √ | √ | | √ | | | √ | | √ |
| **Temperature** | | | | | | | | √ | | | √ | | | | | | | | |
| **Availability, Reliability and Trustworthiness** | | | √ | | √ | | √ | √ | | √ | | | | | √ | | | | |
| **Statistical Analysis Metrics** | | | | | | | | | | | | | | | | √ | | | |
| **Orchestration Decisions** | | | | | | | | | | | | | | | | | | √ | |
| **Contradictory Decisions** | √ | | | | | | | | | | | | | | | | | √ | |
| **Time to Adaptation/Scalability** | | | √ | | | | | | | | | | | | √ | | | √ | |
| **Competition Ratio** | | | | | | | | | | | | | | | | | | | |
| **Cache Hit Ratio** | | | | | | | | | | | | | | | | | | | |
| **Multi-tenancy** | | | | | | | | | | | | | | | √ | | | | |
| **Mobility** | | | | | | | | | | | | | | | | | √ | | √ |
| **Orchestration Actions** | | | | | | | | | | | | | | | | | | √ | |
| **Provisioned Resources** | | | | | | | | | | | | | | | | | | √ | |
| **Deprovisioned Resources** | | | | | | | | | | | | | | | | | | √ | |
| **Contradictory Actions** | √ | | | | | | | | | | | | | | | | | √ | |
| **Oscillation Mitigations** | √ | | | | | | | | | | | | | | | | | √ | |
| **Cost/Profit** | | √ | √ | √ | √ | | √ | √ | √ | √ | √ | √ | | | | | √ | √ | |
| **Technique's Overhead** | | | | | | | | | | | | | | | | | | | |
| **Security and Privacy** | | | | | | √ | √ | | | √ | | | √ | | √ | | | | |

**Table 2**
Critical Analysis about Existing Studies.

| Study | Existing Gaps | Possible Recommendations |
|---|---|---|
| [4] | Self-correction prediction and multi-objective auto-scaling using AI for the trade-off between performance and cost can be accomplished using Deep Learning. | We can extend our model from cloud computing to another emerging computing models such as fog and edge to reduce the latency and response time of cloud applications dynamically |
| [18] | There is a need to develop autonomic resource provisioning technique for cloud computing | Autonomic resource provisioning for fog/edge computing |
| [14] | There is a need to extend the computing landscape to support IoT applications at the edge, not only cloud. | Revisiting the proposed metrics for cloud auto-scaling in the new landscape which demands certain considerations such as mobility and energy-constraint devices. |
| [37] | Splitting Web applications into three tiers of web, business and database cannot completely solve the problem of always-on applications and dependent tiers. | Serverless computing as a solution to always-on problem and transferring from monolithic architecture to microservices can assist energy-limited devices running the web applications at the edge. |
| [1] | Considering even multiple cloud providers cannot be the real-world solution for the future of IoT applications running on end-user devices, mostly not connected to the Internet. | Federated cloud-to-edge computing which is not owned by cloud providers and is brokered through Social IoT networks can solve the vendor lock-in challenge. |
| [36] | There is a need to accommodate all proposed objectives of resource allocations to satisfy the real-world IoT requirements. | A multi-objective resource allocation for not only cloud, but cloud-to-thing landscape is required. |
| [9] | There is need to develop resource provisioning-based scheduling framework for cloud computing. | QoS-aware autonomic resource management for fog/edge computing using MAPE-k loop |
| [13] | There is a need to re-examine cloud-native resource allocation techniques in edge computing. | A feasibility study to explore the identified techniques in edge computing is required. |
| [38] | The feasibility of auto-scaling techniques in edge computing is not evaluated. | The cloud-native auto-scaling solutions need to consider IoT devices as well, not only VMs and containers. |
| [12] | There is a need of a technique for self-optimization and self-configuration of cloud resources | Design an approach for self-management of resources in fog and edge computing |
| [26] | There is a need to implement proposed technique on real-cloud environment. | IoT-Fog based Automatic Thermal Profile Creation Technique for Cloud Data Centers using Artificial Intelligence |
| [39] | There is a need to incorporate Scalability as a performance parameter | Develop a holistic resource management technique to increase the use of renewable energy and quantify the impact of energy usage on climate change. |
| [35] | There is a need to measure the impact of various attacks on QoS at runtime | Design a Self-Protection Approach in Resource Management in fog and edge computing |
| [19] | Analyze and enhance the scalability of ROUTER to allow large number of devices to be integrated without failures | Design a Fog Enabled Cloud based Intelligent Resource Management Approach for Smart city, healthcare, agriculture, transport system. |
| [40] | The challenges of adaptation of novel IoT application to the proposed lightweight virtualization are underrated. | Design of service preparation mechanisms which can transform monolithic IoT applications to microservices. |
| [41] | The feasibility of the proposed resource provisioning technique is not clear in the edge computing. | Reexamining the technique in edge computing considering the mobility and energy constraints of computing nodes. |
| [42] | Not a solution for multi-tenancy in edge and subsequent challenges for cost estimations is provided. | Leveraging Smart Contracts can realize the cost estimation in a federated edge environment. |
| [2] | The proposed auto-scaling solution highly depends on the heavyweight virtualizations as VMs and also only considers the average of latency, not tail latency. | Containers and unikernels investigations can augment the auto-scaling technique as they are well-suited for energy- and resource-constrained edge devices. Also, tail-sensitive auto-scaling is the solution to time-critical IoT applications such as driverless cars. |
| [43] | Task scheduling for edge by containers is proposed while the proposal is suffering from start-up delay for containers. | Warm start-up technique which is originally used for serverless computing can solve the problem. |

to what extend? The negative side of optimizations is that they should be quantitative, not just better performance compared to the common solutions. The answer to this negative side is to stablish SLA for performance metrics. Metrics such as response time can be defined clearly so that the service provider is responsible for refunding the violations. The same is for availability, throughput, resource lifetime and so on. However, the problem is not solved yet, as such metrics are conflicting with each other and a real-world applicable SLA must consider the relation between every single SLA. This means that the response time SLA handler in a load balancing technique, for example, must take care of cost SLA handler. In other words, SLA handlers should communicate with each other, otherwise the optimization of solely response time would not confirm effectiveness of a load balancing technique. Such SLAs can be named federated SLA in which multiple metrics are contributing and are following the same goal. In the literature, mostly a single SLA is being considered which can be a main reason for not finding an all-in-one-solution for each optimization technique like auto-scaling in cloud. Motivated by the paramount importance of Federated SLA (F-SLA) and the complexity of realizing its ambitions, a related component as F-SLA Monitor seem to become a first-class citizen of every optimization technique in cloud to things.

*Experimental environments view*

Collecting observations of the system, from cloud to IoT devices, is a matter of concern which demands further identifications. The observations collection highly depends on the experimental environments: (1) public commercial, (2) custom testbed or (3) simulators. In public commercial environments such as Amazon AWS or Microsoft Azure the orchestration is generally done by the provider. There are particular services such as EC2 VMs wherein the customer can deploy their own IoT application on and imple-

ment corresponding orchestration. In this situation, the orchestrator is able to measure the application-level and VM-level measurements, but still the underlying infrastructure as hosts and data centers are out of control of the orchestrator. Measuring the infrastructure is essential for power consumption, availability and related evaluations. GRID' 5000 is another widely-used testbed for cloud-related orchestrations such as load balancing [44].

Custom testbeds are open to orchestrator to measure the whole computing system. Practical platforms such as OpenStack are widely-used in the orchestration domain. By supporting containers, such platforms can be deployed at the fog and edge layers, extending the platform-native measurements to the edge of the network. Using such testbeds, developers can also utilize standard tools such as HAProxy or Nginx for careful measurements of performance metrics. HAProxy, for instance, is periodically in a fine-grained manner collecting more than seventy metrics, mostly application-specific. Interestingly, with the emerge of Service Mesh techniques and the possibility of bringing the fully distributed microservice to the edge, now it is further feasible to measure the detailed metrics for evaluating the orchestrator performance. However, despite the possibility of extending such tools in case not supporting certain measurements, implementing particular IoT-specific metrics is not feasible in practice, largely due to unknown, conflicting and diverse aspects of IoT applications. Add to those limitation the time and cost to evaluate orchestrations in large scale experiments.

Simulators are the third, but under particular circumstance the first, experimental environment for evaluating orchestrators and consequently collecting observations and measuring the performance metrics. Several simulators have been introduced for cloud, fog, edge, and IoT and a combination thereof. They mitigate the complexity of implementations, yet they provide merely an abstraction of the computing environment. After a comprehensive literature review, a taxonomy of available simulators was found which are categorized according to their target landscape (see Fig. 5). The obvious lesson we can learn by this taxonomy is the focus of research on the cloud. This is mainly due to earlier emergence of cloud than other landscapes. Given (1) the diverse applications of IoT which demand their own customized simulators; (2) the diverse edge devices, ranging from routers to micro data centers, which are not well-identified yet; and most importantly, (3) the importance of optimizations for orchestration techniques, it is highly likely that further simulators for fog, edge and IoT landscapes will be developed, other than cloud. One of the major reasons that simulators are gaining more and more popularity is the impossibility of measuring particular metrics in alternative solutions. Motivated by this mission

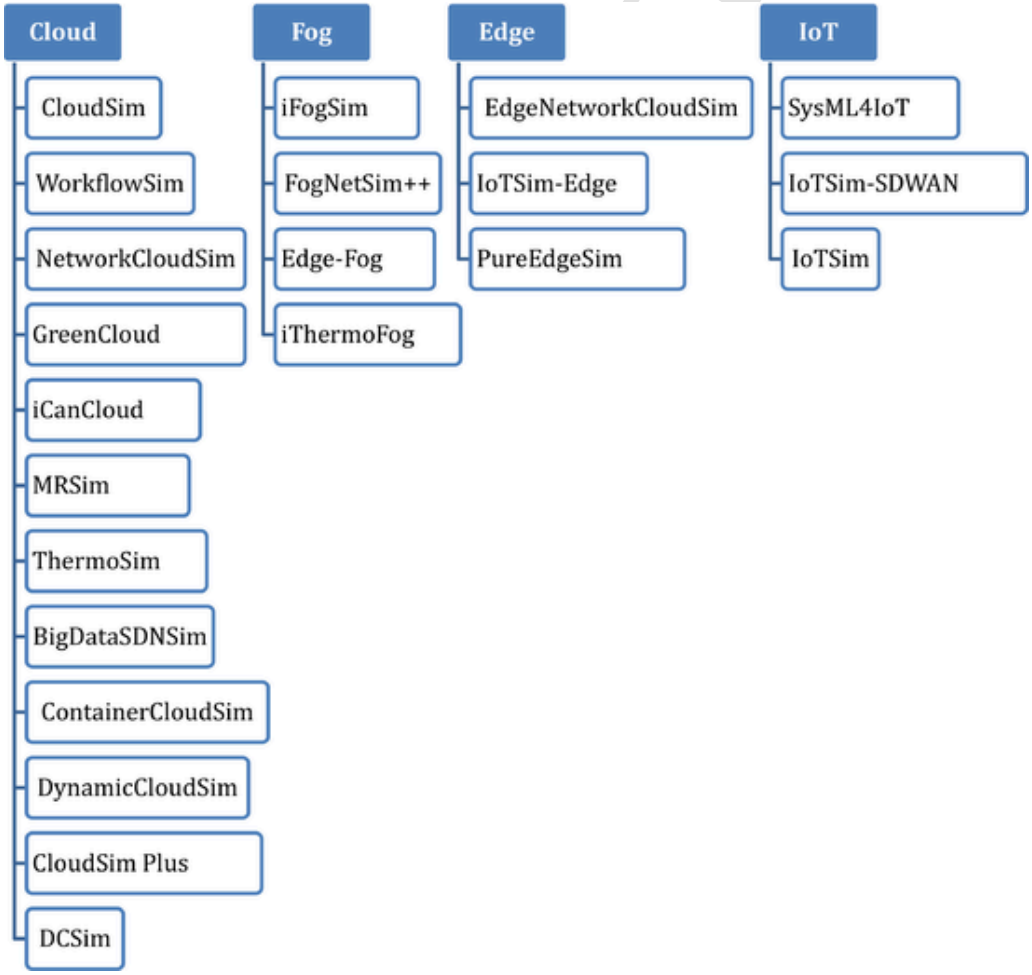



**Fig. 5.** A Taxonomy of Developed Simulators by Research Community.

of simulators, it is worth investigating what performance metrics they propose for each landscape. This investigation can reveal how much the community is concerned about particular measurements, and how much such metrics have potentially been impossible in public commercial computing environments and custom testbeds. We conducted the investigation and revealed the supported metrics by each simulator which is shown in Fig. 6. There exist general simulation tools such as MATLAB as well which are used for analytical evaluations mostly.

A comparison between the performance metrics supported by existing simulators [45-47] is made and demonstrated in Fig. 5. Cloud-domain simulators are supporting a wider range of metrics including cost and SLA violation which were not seen in other categories. Fog and edge simulators stand on the second level of active support, greater support than IoT/Mist simulators. Common supported metrics for all landscapes revealed to be: resource utilization, energy, response time and latency which are considerably investigated in the literature. Those metrics seem to be unsolved and critical measurements for the researchers. Major metrics which need further implementations in the IoT, edge and fog, would be the cost and SLA violation. Such metrics are by far easier to implement in cloud as the SLA and cost can be well-defined by cloud providers. However, emulating and realizing such metrics in IoT landscape, for instance, demands significant effort and is facing fundamental challenges. Precisely, in IoT landscape, there might be different service providers within the ad hoc network which are not aware of the network nodes, of monetization of the service they provide, of the SLA of the guest's service and so on.

Given the complexity of the real implementation, the existing simulator or upcoming simulators might be the pioneers for bringing SLA violation and cost measurements for IoT, edge and fog computing as in simulation environments many complex variables can be omitted, unconsidered or only considered at some extent. Second, although edge simulators fail to support cost in terms of money, some simulators such as Edge-Fog consider a predefined value for the execution time of tasks in edge. This approach can also be starting point for considering cost measurements in real experiments.

*Applications pipeline view*

From applications pipeline perspective, the metrics can be categorized as those which are measurable in application, business and database layers of the applications as well as those which are common for all layers (see Fig. 7). This categorization reveals how the metrics can be measured during the execution of task/request when travelling around application's pipeline. Take a cloud-hosted web application as an example, the requests come first to the application layer where the Graphical User Interface, load admission and subsequently the load balancer, such as Nginx is to distribute the requests. Given the responsibility of load balancers, monitoring-related metrics such as throughput, delay time and damaged tasks can be measured. Second layer is business (not necessarily), where the processing and execution of requests are performed by underlying infrastructure as VMs or containers running the logic unit of the application pipeline. Metrics such as resource utilization are commonly measured in business layer, due to having the direct access to computation resources. Third layer is database which is populated with users' data and metrics as cache hit ratio as well as resource utilization can be measured there. A comprehensive categorization of metrics related to each layer of applications is demonstrated in Fig. 7.

Applications' architecture can influence the measurement approaches. Given the advances in Service Oriented Application (SOA) and the emergence of microservices, the conventual application pipeline may not necessarily exist. Instead, each microservice which is a responsible for providing a certain service of the application may inherit the characteristics of all layers. One more step in application deployment is Function-as-a-Service (FaaS) wherein the services of an application, here functions, may also not demand partic-

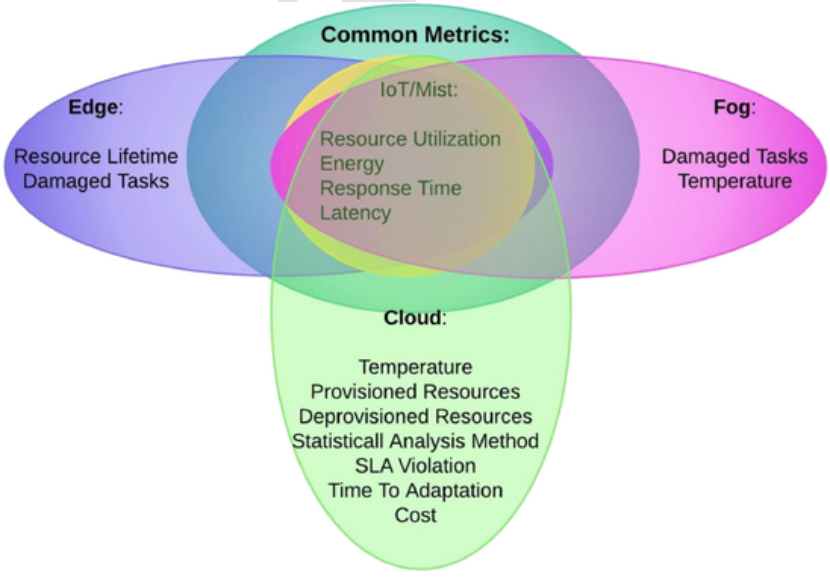



**Fig. 6.** Performance Metrics Supported by Existing Simulators.

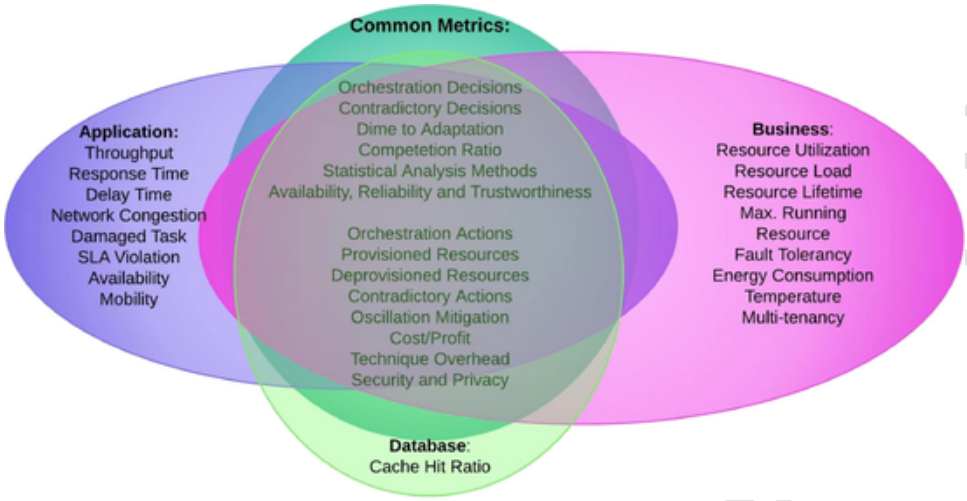



**Fig. 7.** Mapping Performance Metrics to Application Pipeline.

ular layers such as database given the stateless nature of Serverless computing. Hence, measurement of metrics for such generation of applications may significantly demand revisiting applications pipeline and the requirements of services and functions, not the application as a whole.

Not only the application architecture, the underlying infrastructure is hosting the application needs further considerations. Given the changing and uncertain nature of IoT and edge nodes hosting the application, whether conventual pipeline or microservices and functions, the measurement of metrics demands re-examining cloud-native solutions. While in cloud the metrics are simply measured by accessing immobile physical hosts, in edge such measurements demand consideration of parameters like mobility and unreliability of underlying infrastructure. For instance, measuring SLA violation in cloud is done by considering the number of requests sustaining delay time more than expected, while in edge the node churn [27] and trustworthiness [48], to name a few, have to be considered. Otherwise, the SLA violation fails to represent a realistic measurement fully conforming to edge landscape.

*Future directions*

*Industrial steps are required*

While service providers such as Amazon and Google are defining SLA for their services, such definitions are at early stages and would not consider different metrics. For instance, Amazon EC2 service guarantees an hourly uptime of 90% for individual instances each clock hour. They will not charge the user if the 90% uptime is not realized. This is the only commitment they provide. There could be doubts about the performance of the instance even when it is available. Amazon does not refund the money, and damage to the business due to the unavailability is not considered. The frequent occurrences of such failures would not affect the SLA violation policies of Amazon and the only disregard the last hour usage of the instance as a penalty. The potential solution would be to first identifying measurable metrics to then enrich the SLA for public service providers like Amazon. The taxonomy provided in this survey can be a great benchmark and standard for such advancements.

*Warm-up steps for transferring metrics from simulators to real-world*

As discussed, there might be particular metrics being used in simulation environments which are not at the time transferrable to the real-world evaluations or at least demand huge efforts. Not only this, but also there might be metrics newly identified as a subsequence of intersection of IoT and novel computing landscapes or of new applications of IoT. This is because almost every gadget and machine are entering the IoT world day by day. The problem of transferring simulators metrics and defining newly identified metrics may be difficult to address without involving warm-up metrics. In this approach, instead of fully quantitative specified metric (e.g., response time = 100 ms), a degree of acceptance performance is defined for the corresponding metric to be obeyed by the orchestrator (e.g., 80 ms < response time < 120 ms). This warm-up approach can join the application catalog as well because some metrics would not necessarily require staying at certain values in certain situations. For instance, a thermometer IoT device which is periodically sensing temperature and sending to the cloud, demands availability of cloud as a performance metric but not necessarily 100% of the time. The sensed data can be kept on IoT side and later on be sent to the cloud. Hence, a range of response time can work for such applications instead of an exact time. By this approach every metric can be first relaxed and then with further investigations the quantitative solutions will be found. As a first attempt to explore this warm-up step, authors in [49] proposed Quality of Results (QoR) as a starting point to bring every individual newly identified metric to the real-world implementations.

*Realizing application catalog is a must*

Efforts in orchestration are mostly considering particular metrics according their own understanding of the application's requirements. They evaluate a web service by response time, for instance. However, it is the customer who demands adaptation of an application, who wants pay service cost, and who should define the acceptable performance. Moreover, the application's requirement may vary depend on the computing landscape. For instance, an object detection (e.g., bird detections) IoT application in a farm may require locality of task execution while the same application in driverless cars (e.g., car detections) may demand ultra-low latency. Given that, realizing a documented application catalog containing SLA specifications of the application is essential to be realized. The application catalog represents the application and customer demands and the responsibilities of the orchestrator. Further step to realize an all-in-one-solution for the communication of applications with orchestrators is to define and standardization a machine-readable SLA language so that every performance metric can be considered. The language should be pluggable to any platform such as public clouds, edge computing and mist computing. As a starting point in this domain, [50] proposed a grammar for SLA specifications. They did not explore how the specifications can cooperate with each other to avoid conflict, though.

***Trade-off between penalty and qos loss tolerance***

While preserving QoS seems to be always a must for orchestrations to respect to, in some cases the Quality of Experience (QoE) can be a great reason to compromise a performance metric. QoR refers to the overall evaluation of a metric so that the final result is fairly acceptable. For instance, after decomposing a latency-sensitive IoT application, like face detection in Smart City scenarios, into several microservices, particular microservices may only be responsible for saving the detected objects into the cloud. The main concern about the latency in such microservices is to quickly perform object detection followed by corresponding actuations, and saving the detected objects can be done separately without any priority. QoR refers to identifying such functions and not imposing penalty as is defined for other microservices in the application as SLA Violation. Overall, careful dissection of not only the application, but also the underlying host's requirements can help differentiate trivial and non-trivial parts of the system to facilitate and compromise the evaluation of an orchestration technique. This is important because guaranteeing the whole performance metrics for certain services under particular situations may be difficult. As a useful starting point, the effort made by [49].

*Forget static metrics and fulfill elastic metrics*

Defining static metrics to fulfill a certain value for the given metric, which cannot the practical or efficient way for the future of dynamic IoT applications. For instance, considering the restrict response time of 100 ms for all equipment in a factory augmented by Industry IoT (IIoT) may not be necessary. Why not letting the equipment function with less pressure when the production line is not much busy to increase its lifetime? As a solution, dynamic thresholds for response time in such situations may both fulfill the latency requirements and equipment lifetime. It has been identified that perfect opportunities can migrate from static measurements to dynamic and even adaptive metrics which can help the system performance from other aspects as well.

*The end of mono-objective optimizations era*

Mono-objective optimization proposes to optimize only one objective, while multiple critical metrics may be underrated. A latency-aware auto-scaling, for instance, may alleviate the latency by resource overprovisioning. However, such a mechanism imposes severe renting cost for the application provider in terms of renting surplus provisioned resources. Hence, latency-aware solutions solely cannot be perfect solution. This is the case for the whole metrics. Considering that, it is apparent that real-world orchestration techniques have to consider multi-objective optimizations to be practical in production environments. The effort made by [51] gives a clear overview of this issue.

*Avoid the average measurements as tail-related metrics are must*

While an overwhelming majority of works on cloud, fog and edge optimizations target the simple measurements and patterns of the corresponding metric such as time to adaptation, other meaningful behavior of the metric needs further consideration. Of widely-used metrics is response and delay time which are mostly measured in terms of the average. However, what about the tail? Take a driverless car application which is responsible for activating the brake when hazards are perceived as an example. If the orchestration technique for this latency-sensitive application is committed to respect the average of latency less than 100 ms, the activation may accrue after 1 second in some situations and in 20 ms in others. Of course, the orchestration may at the end preserve the average latency of 100 ms or even lesser, but in case of activating the brake after 1 second, a disaster might happen. Hence, for such applications the tail of latency must be measured not the average. This is the case for measuring multiple metrics such as response time, delay time, time to adaptation, resource lifetime and any metric related to time. This issue emphasizes the importance of tail-tolerant orchestrations for IoT applications as well.

## Summary and conclusions

Orchestration techniques such as resource allocation, resource provisioning, task scheduling, load balancing, Virtual Machine (VM) migration, task placement, server consolidation, service composition, data caching, service discovery, etc. play an important role for cloud, fog and edge computing optimizations. Developing the technique is half the story while evaluating its performance under specific conditions is critical to realize if the technique can comply with the objectives and is applicable in real-world or not.

To evaluate such orchestration techniques, certain performance metrics and standards must be identified, developed and leveraged to address the concerns. Addressing those challenges demands a clear understanding of potential metrics which can be used in this optimization domain, for which the developers mostly struggle. In this paper, we have discussed various performance and metrics for cloud, fog and edge computing in a comprehensive way, which could be useful for future studies. Several open issues are also highlighted to provide with the researchers a clear understanding of the future directions in performance evaluation of orchestration techniques in cloud to things.

## Uncited references

[30].

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

We would like to thank the editors, area editor and anonymous reviewers for their valuable comments and suggestions to help and improve our research paper.

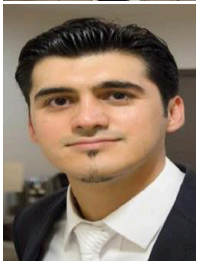
## References

[1] Y Al-Dhuraibi, F Paraiso, N Djarallah, P Merle, Elasticity in cloud computing: state of the art and research challenges, IEEE Trans. Serv. Comput. 11 (2) (2018) 430–447.
[2] M S Aslanpour, M Ghobaei-Arani, A Nadjaran Toosi, Auto-scaling web applications in clouds: a cost-aware approach, J. Netw. Comput. Appl. 95 (2017), doi:10.1016/j.jnca.2017.07.012.
[3] S Singh, I Chana, A survey on resource scheduling in cloud computing: issues and challenges, J. grid Comput. 14 (2) (2016) 217–264.
[4] T Lorido-Botran, J Miguel-Alonso, J A Lozano, A review of auto-scaling techniques for elastic applications in cloud environments, J. Grid Comput. 12 (4) (2014) 559–592.
[5] S S Gill, R Buyya, A taxonomy and future directions for sustainable cloud computing: 360 degree view, ACM Comput. Surv. 51 (5) (2018) 1–33.
[6] M S Aslanpour, M Ghobaei-Arani, M Heydari, N Mahmoudi, LARPA: a learning automata-based resource provisioning approach for massively multiplayer online games in cloud environments, Int. J. Commun. Syst. (2019) e4090.
[7] L Zhou, C-H Chou, L N Bhuyan, K K Ramakrishnan, D Wong, Joint Server and Network Energy Saving in Data Centers for Latency-Sensitive Applications, 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2018, pp. 700–709.
[8] C-H Chou, L N Bhuyan, D Wong, μDPM: dynamic Power Management for the Microsecond Era, 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2019, pp. 120–132.
[9] S S Gill, I Chana, M Singh, R Buyya, CHOPPER: an intelligent QoS-aware autonomic resource management approach for cloud computing, Cluster Comput (2017) 1–39.
[10] S Singh, I Chana, M Singh, R Buyya, SOCCER: self-optimization of energy-efficient cloud resources, Cluster Comput 19 (4) (2016) 1787–1800.
[11] S S Gill, et al., Transformative effects of IoT, Blockchain and Artificial Intelligence on cloud computing: evolution, vision, trends and open challenges, Internet of Things 8 (2019) 100118.
[12] S S Gill, I Chana, M Singh, R Buyya, RADAR: self-configuring and self-healing in resource management for enhancing quality of cloud services, Concurr. Comput. Pract. Exp. (2018) e4834.
[13] S H H Madni, M S A Latiff, Y Coulibaly, Recent advancements in resource allocation techniques for cloud computing environment: a systematic review, Cluster Comput 20 (3) (2017) 2489–2533.
[14] E F Coutinho, F R de Carvalho Sousa, P A L Rego, D G Gomes, J N de Souza, Elasticity in cloud computing: a survey, Ann. Telecommun. des télécommunications 70 (7–8) (2015) 289–309.
[15] Q Zhang, L Cheng, R Boutaba, Cloud computing: state-of-the-art and research challenges, J. internet Serv. Appl. 1 (1) (2010) 7–18.
[16] S Mustafa, B Nazir, A Hayat, S A Madani, Resource management in cloud computing: taxonomy, prospects, and challenges, Comput. Electr. Eng. 47 (2015) 186–203.
[17] R N Calheiros, R Ranjan, R Buyya, Virtual machine provisioning based on analytical performance and QoS in cloud computing environments, Parallel processing (ICPP), 2011 international conference on, 2011, pp. 295–304.
[18] S Singh, I Chana, Q-aware: quality of service based cloud resource provisioning, Comput. Electr. Eng. 47 (2015) 138–160.
[19] S S Gill, P Garraghan, R Buyya, ROUTER: fog enabled cloud based intelligent resource management approach for smart home IoT devices, J. Syst. Softw. 154 (2019) 125–138.
[20] M Abdullahi, M A Ngadi, Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment, PLoS ONE 11 (6) (2016) e0158229.
[21] S Tuli, R Mahmud, S Tuli, R Buyya, Fogbus: a blockchain-based lightweight framework for edge and fog computing, J. Syst. Softw. 154 (2019) 22–36.
[22] S Singh, I Chana, EARTH: energy-aware autonomic resource scheduling in cloud computing, J. Intell. Fuzzy Syst. 30 (3) (2016) 1581–1600.
[23] S S Gill, et al., Holistic resource management for sustainable and reliable cloud computing: an innovative solution to global challenge, J. Syst. Softw. 155 (2019) 104–129.
[24] S-Y Jing, S Ali, K She, Y Zhong, State-of-the-art research study for green cloud computing, J. Supercomput. 65 (1) (2013) 445–468.
[25] S K Garg, R Buyya, Green cloud computing and environmental sustainability, Harnessing Green IT Princ. Pract. (2012) 315–340.
[26] S S Gill, et al., ThermoSim: deep learning based framework for modeling and simulation of thermal-aware resource management for cloud computing environments, J. Syst. Softw. (2020) 110596.
[27] A J Ferrer, J M Marques, J Jorba, Ad-Hoc Edge Cloud: a Framework for Dynamic Creation of Edge Computing Infrastructures, 2019 28th International Conference on Computer Communication and Networks (ICCCN), 2019, pp. 1–7.
[28] X Wang, Y Han, C Wang, Q Zhao, X Chen, M Chen, In-edge ai: intelligentizing mobile edge computing, caching and communication by federated learning, IEEE Netw 33 (5) (2019) 156–165.
[29] H Tan, Z Han, X-Y Li, F C M Lau, Online job dispatching and scheduling in edge-clouds, IEEE INFOCOM 2017-IEEE Conference on Computer Communications, 2017, pp. 1–9.
[30] Y Gan, C Delimitrou, The Architectural Implications of Cloud Microservices, IEEE Comput. Archit. Lett. 17 (2) (2018) 155–158.
[31] M S Aslanpour, S E Dashti, M Ghobaei-Arani, A A Rahmanian, Resource provisioning for cloud applications: a 3-D, provident and flexible approach, J. Supercomput. (2017), doi:10.1007/s11227-017-2156-x.
[32] M S Aslanpour, S E Dashti, Proactive Auto-Scaling Algorithm (PASA) for Cloud Application, Int. J. Grid High Perform. Comput. 9 (3) (Jul. 2017) 1–16, doi:10.4018/IJGHPC.2017070101.

[33] M S Aslanpour, S E Dashti, SLA-Aware resource allocation for application service providers in the cloud, 2016 2nd International Conference on Web Research, ICWR 2016, Apr. 2016, pp. 31–42, doi:10.1109/ICWR.2016.7498443.
[34] P D Kaur, I Chana, A resource elasticity framework for QoS-aware execution of cloud applications, Futur. Gener. Comput. Syst. 37 (2014) 14–25.
[35] S S Gill, R Buyya, SECURE: self-protection approach in cloud resource management, IEEE Cloud Comput 5 (1) (2018) 60–72.
[36] W Z Khan, E Ahmed, S Hakak, I Yaqoob, A Ahmed, Edge computing: a survey, Futur. Gener. Comput. Syst. 97 (2019) 219–235.
[37] C Qu, R N Calheiros, R Buyya, Auto-scaling web applications in clouds: a taxonomy and survey, ACM Comput. Surv. 51 (4) (2018) 73.
[38] T Chen, R Bahsoon, X Yao, A survey and taxonomy of self-aware and self-adaptive cloud autoscaling systems, ACM Comput. Surv. 51 (3) (2018) 61.
[39] J F Pérez, L Y Chen, M Villari, R Ranjan, Holistic workload scaling: a new approach to compute acceleration in the cloud, IEEE Cloud Comput 5 (1) (2018) 20–30.
[40] R Morabito, V Cozzolino, A Y Ding, N Beijar, J Ott, Consolidate IoT edge computing with lightweight virtualization, IEEE Netw 32 (1) (2018) 102–111.
[41] S Islam, J Keung, K Lee, A Liu, Empirical prediction models for adaptive resource provisioning in the cloud, Futur. Gener. Comput. Syst. 28 (1) (2012) 155–162.
[42] K Toczé, S Nadjm-Tehrani, A taxonomy for management and optimization of multiple resources in edge computing, Wirel. Commun. Mob. Comput. (2018) 2018.
[43] K Kaur, T Dhand, N Kumar, S Zeadally, Container-as-a-service at the edge: trade-off between energy efficiency and service availability at fog nano data centers, IEEE Wirel. Commun. 24 (3) (2017) 48–56.
[44] A N Toosi, C Qu, M D de Assunção, R Buyya, Renewable-aware Geographical Load Balancing of Web Applications for Sustainable Data Centers, J. Netw. Comput. Appl. (2017).
[45] J Byrne, et al., A review of cloud computing simulation platforms and related environments, International Conference on Cloud Computing and Services Science, 2, 2017, pp. 679–691.
[46] M C Huebscher, J A McCann, A survey of autonomic computing—Degrees, models, and applications, ACM Comput. Surv. 40 (3) (2008) 7.
[47] M Ashouri, F Lorig, P Davidsson, R Spalazzese, Edge Computing Simulators for IoT System Design: an Analysis of Qualities and Metrics, Futur. Internet 11 (11) (2019) 235.
[48] S Wang, et al., A Case for Adaptive Resource Management in Alibaba Datacenter Using Neural Networks, J. Comput. Sci. Technol. 35 (1) (2020) 209–220, doi:10.1007/s11390-020-9732-x.
[49] Y Li, Y Chen, T Lan, G Venkataramani, Mobiqor: pushing the envelope of mobile edge computing via quality-of-result optimization, 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 1261–1270.
[50] A Alqahtani, P Patel, E Solaiman, R Ranjan, Demonstration abstract: a toolkit for specifying service level agreements for IoT applications, arXiv Prepr. arXiv1810.02749 (2018).
[51] F A Salaht, F Desprez, A Lebre, An overview of service placement problem in fog and edge computing, ACM Comput. Surv. 53 (3) (2020) 1–35.

**Mohammad Sadegh Aslanpour** is a PhD student in Monash University and CSIRO's DATA61, Australia. He obtained his MSc degree in Computer Engineering Islamic Azad University, Tehran Science and Research (Sirjan) Branch, Iran in 2016. He also obtained his Bachelor and Associate degrees in Computer-Software in 2012 and 2010, respectively. From 2011 to 2019, he worked in the industry, IT Department of Jahrom Municipality, Iran as Software Engineer. He is also serving as Editorial Board Member and Reviewer for some international high-ranked journals. His-research interests include orchestration of Cloud, Fog and Edge Computing, Serverless Computing, and Autonomous Systems. For more details, please visit his homepage: aslanpour.github.io.

Sukhpal Singh Gill is a Lecturer (Assistant Professor) in Cloud Computing at School of Electronic Engineering and Computer Science, Queen Mary University of London, UK. Prior to this, Dr. Gill has held positions as a Research Associate at the School of Computing and Communications, Lancaster University, UK and also as a Postdoctoral Research Fellow at CLOUDS Laboratory, The University of Melbourne, Australia. His-research interests include Cloud Computing, Fog Computing, Software Engineering, Internet of Things and Healthcare. For further information, please visit www.ssgill.me.



Adel Nadjaran Toosi is a Lecturer in Faculty of Information Technology, Monash University, Australia. Before joining Monash University, he was a research fellow at the School of Computing and Information Systems, The University of Melbourne. He received his Ph.D. degree from the University of Melbourne in 2015. His-thesis was one of the two theses nominated for the University Chancellor's Prize and John Melvin Memorial Scholarship for the Best Ph.D. Thesis in Engineering. His-research interests include Distributed Systems, Cloud Computing, Software-Defined Networking (SDN), Green Computing, and Soft Computing. For more details, please visit his homepage: http://adelnadjarantoosi.info.