Microservices Adaptation using Machine Learning: A Systematic Mapping Study

Anouar Hilali, Hatim Hafiddi and Zineb El Akkaoui
InnovatiVE REsearch on Software, Systems and daTa,
Institut National des Postes et Télécommunications, Rabat, Morocco

Keywords: Self-adaptation, Microservices, Machine Learning.

Abstract: The Microservice architecture is increasingly becoming the preferred architecture of modern applications. The logically distinct components that make up microservices make continuous delivery easier compared to monolithic architectures. This feature however makes it difficult for engineers to control the underlying services and properly adapt them at run-time. Designing our microservices as self-adaptive systems helps us tackle this issue. Each microservice can then dynamically monitor and adapt its behavior to change certain aspects of itself to achieve self-adaptive goals. The use of statistical and Machine Learning (ML) techniques helps in this area in a lot of ways (e.g., predicting resource usage, anomaly detection, etc.). This paper aims to provide a state of the art of the use of ML in microservice adaptation, the main goal is to provide an overview of the field and identify the most frequent adaptation goals and the types of adaptation techniques used. In order to carry out a comprehensive analysis, a well-defined method of systematic mapping is performed to categorize, according to a detailed scheme, every paper relevant to this topic. The results can potentially shed light on areas where further investigation might be warranted.

1 INTRODUCTION

Nowadays, modern applications are changing from a monolithic architecture where the application is constructed as a single entity, to a more distributed architecture where complex applications are broken down into logically recognizable components; microservices (Khazaei et al., 2018a). The Microservice architecture is an architectural style in which an application is designed as a set of small services communicating with each other using a lightweight mechanism (Fowler, 2014). The decoupling of components that is characteristic of this type of architecture makes continuous delivery safer and cheaper contrary to many other architectural styles (Junior, 2018). However, several obstacles appear that make it difficult to manually manage applications at run-time (e.g., resource optimization, granularity of services, context, fault detection, etc.). A solution to this is to design these microservices as self-adaptive systems that can dynamically monitor and adapt their behavior to change certain aspects of themselves to meet certain goals (e.g., when operating conditions are not stable and optimal (Mendonça et al., 2019), when services need to be discovered in a changing context (Wanigasekara, 2015a), etc.). Self-adaptation techniques are known to be promising ways of tackling and managing run-time uncertainties (Sanctis et al., 2020a). We can distinguish four adaptation goals in self-adaptive systems: self-configuration (i.e., systems that configure themselves automatically), self-optimization (systems that are constantly looking for ways to optimize performance), self-healing (systems that detect and fix anomalies) and self-protection (systems that defend themselves from attacks) (Khazaei et al., 2018a). An autonomic management system contains the logic that tackles one or several of these adaptation goals. The four elements: Monitor, Analyze, Plan, and Execute achieve the necessary functions of any self-adaptive system. These elements share common Knowledge therefore the model is usually referred to as the MAPE-K model (Kephart and Chess, 2003). Applying statistical and ML techniques in order to better any aspect of self-adaptive systems helps in bringing about adaptation without human intervention. ML algorithms for instance, can be employed to predict resource usage patterns based on historical data of an application's microservices and effectively anticipate changes (e.g., auto-scaling, placement reconfiguration, etc.) (Junior, 2018).

The goal of this paper is to achieve a research area overview, identifying the most common microservices adaptation goals, techniques that are used in accomplishing these goals and the frequency of using ML algorithms in carrying this out. We therefore choose to perform a systematic mapping of the

Hilali, A., Hafiddi, H. and El Akkaoui, Z.
Microservices Adaptation using Machine Learning: A Systematic Mapping Study.
DOI: 10.5220/0010578905210532
In *Proceedings of the 16th International Conference on Software Technologies (ICSOFT 2021)*, pages 521-532
ISBN: 978-989-758-523-4

existent literature. Generally, a systematic mapping study structures the type of research, reports and results that have been published by classifying them and generates a visual summary of its results (Kitchenman and Charters, 2007). In this work, the discussion of the findings focus on supporting the understanding of what has been addressed by the research community and eventually propose future research guidelines in this field.

The remainder of this paper is structured as follows: section 2 details the research methodology used for the mapping study accomplished in this work, from establishing the research questions to the resulting systematic map. Then, in section 3 we provide an analysis and a discussion of the results extracted during the mapping study. In section 4 we discuss the threats to validity of this paper. Section 5 specifies similar works and situates our contribution. Finally, section 6 outlines the main contribution of this paper and considers some directions that could be potentially investigated by the research community.

2 RESEARCH METHODOLOGY

In order to achieve the goals set above, our systematic mapping study process followed Kitchenman's approach. This type of study is helpful when it is discovered that very little evidence is likely to exist on a certain topic (Kitchenman and Charters, 2007). Figure 1 represents the underlying steps of performing a systematic mapping study. Each step of the process has an outcome, the last step produces the systematic map. In the upcoming section we will describe each step in detail and apply it to our particular case.

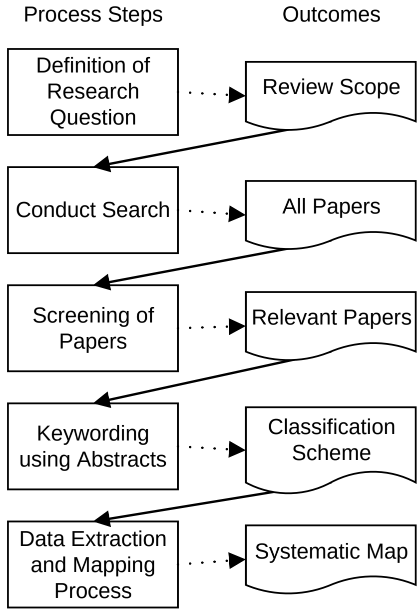

Figure 1: Steps for performing a systematic mapping study.

2.1 Definition of Research Questions

Initially we must define the research scope by identifying goals that are expressed in research questions. We chose to follow structured criteria to frame our questions (Kitchenman and Charters, 2007), the following questions were stated:

RQ1: What are the microservices adaptation targets (i.e., sub-goals of those mentioned in the introduction) that the research community tackles?

RQ2: Are the adaptations on an application or an architecture level?

RQ3: What adaptation techniques were used?

Based on a classification scheme that we detail later on, we can answer more precise questions:

RQ4: If machine learning was used, what algorithms were picked?

RQ5: What level of empirical evidence was attained by researchers?

2.2 Conduct Search for Primary Studies

In order to answer our questions, we must identify the key studies by setting up a set of search strings (Petersen et al., 2008). Multiple digital libraries were chosen to extract relevant research using the two criteria suggested by Kitchenman; Population (i.e. An application area) and Intervention (a software methodology, tool, technology or procedure that addresses a specific issue) (Kitchenman and Charters, 2007):

- Population: Microservices.
- Intervention: Adaptation, self-adaptation, self-healing and context-aware.

Our Keywords can be deduced from each facet, the population and intervention facets lead us to a search string similar to this: *("microservices" OR "microservice" OR "micro-service" OR "micro-services") AND ("adaptation" OR "adaptive" OR "self-adaptation" OR "self adaptation" OR "self-adaptive" OR "self adaptive" OR "context-aware" OR "context aware" OR "self-healing" OR "self healing").*

The search query slightly varies from one digital library to the other in order to comply with each library's query rules.

The choice of what digital libraries to use was based on (Breton et al., 2007)'s identification of the most relevant digital libraries to software engineers. The choice of the search string was made in order

Table 1: Search queries used on different digital libraries and their respective results.

Digital Library	Number of results
ACM Digital Libray	16
IEEExplore	88
ScienceDirect	10
SpringerLink	149
Google Scholar	12

to select all papers that discuss the adaptation of microservices regardless of tools used (e.g. Artificial Intelligence algorithms). We didn't consider the comparison facet because it didn't make sense in our case, while the outcome facet would restrict our research and wouldn't allow us to paint a broad overview of the research area. The results are accurate as of 10th of January 2021, the number of total publications is 275. Table 1 shows the full results.

2.3 Screening of Papers for Inclusion and Exclusion

Inclusion and exclusion criteria were used to only focus on papers that are relevant to answering our domain questions. A percentage of these criteria are based on the fact that our main aim is microservice architectures. Other criteria are based on practical issues.
The inclusion criteria were:
IC1: publications produced in English.
IC2: publications that are peer-reviewed and published in journals, conferences and workshops.
IC3: publications that explicitly mention our keywords in the title, abstract and/or keywords.
These exclusion criteria were used:
EC1: publications where the authors focus on architectures other than microservices.
EC2: publications such as books, chapters of books or similar ones.
EC3: publications that are not in the computer science field.
Duplicates are automatically dealt with using the merge duplicates feature on the open source research tool Zotero. Initially criteria IC1 was applied for all queries. No filters were used for IEEE. Criteria IC2, IC3, EC2 and EC3 were applied using the built-in filters of the mentioned digital libraries. *"Research Article"* and *Computer Science* filters were applied for ScienceDirect and ACM Digital Library and finally *"Conference Paper"* and *"Article"* filters were applied for SpringerLink. We then filtered through the publications manually, reading abstracts and if necessary introductions and conclusions. We ranked the papers based on their relevance to our objectives by labeling each paper with a number (0 would be irrelevant, 1 would be somewhat relevant and 2 as in relevant). We double checked the somewhat relevant papers a second time and discarded the irrelevant ones. Initially the number of relevant publications went from 275 to 129 based on reading abstracts. Then the final result was reduced to 62 publications that are useful for our purposes. Figure 2 illustrates this process.

2.4 Keywording of Relevant Papers

Before going ahead and setting up a classification scheme, the authors went through all the relevant papers and made sure they were all aligned with our objectives. We not only relied on reading the abstracts but at times had to carefully read the full paper. In order to properly cover all aspects of our research, the papers were classified using the following classification scheme:
Adaptation Targets (RQ1). In order to answer RQ1, we classified papers based on what area of microservices were adapted.
Adaptation Techniques (RQ2). To classify the techniques used in each research and address RQ2.
Adaptation Level (RQ3). To tackle RQ3, we also classified papers based on what level of adaptation were the techniques aimed at.
Use of Machine Learning (RQ4). To classify papers on whether they used machine learning in their adaptations and if yes, what type of algorithms were used in their solution.
Evidence (RQ5). We chose an existing classification of research approaches to classify our papers (Wieringa et al., 2006) and address RQ5 on what type of empirical evidence is provided by each paper:

- *Validation Research*: A paper where the techniques that are examined are innovative and are not yet applied in practice.
- *Evaluation Research*: A paper where the techniques discussed were implemented in practice and an evaluation of these techniques are conducted.
- *Solution Proposal*: A paper where a solution to a given problem is presented. It can be either new or built on top of existing approaches. The solution is also illustrated by a specific example.
- *Philosophical Papers*: Papers where a novel perspective is proposed by structuring a given field.
- *Opinion Papers*: Papers where the author express a personal opinion on the validity of a certain technique or how it should be implemented.

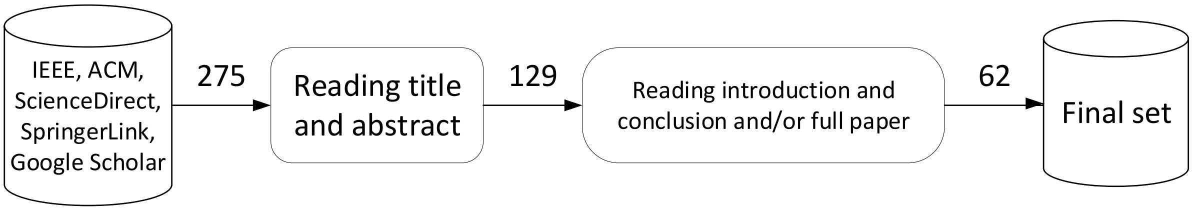

Figure 2: Inclusion and exclusion criteria process.

- *Experience Papers*: Papers where an explanation of how a technique has been done in practice is given. It relies on the personal experience of the authors.

2.5 Data Extraction and Mapping Process

We conclude our review with the data extraction step. All 64 papers relevant to us were fully read. Based on the research questions defined, an Excel data extraction table was developed to document the data extraction process. Thanks to the produced graphs, we are now able to have a proper overview of what the research community focuses on the microservices adaptation research area. Table 3 illustrates the detailed scheme.

3 MAIN FINDINGS

All articles were published between September 2015 and January 2021. Figure 3 shows an increase in the number of published articles on the the topic of Microservice adaptation. The decrease between 2020 and 2021 is due to the fact that the results were as of the first month of 2021.

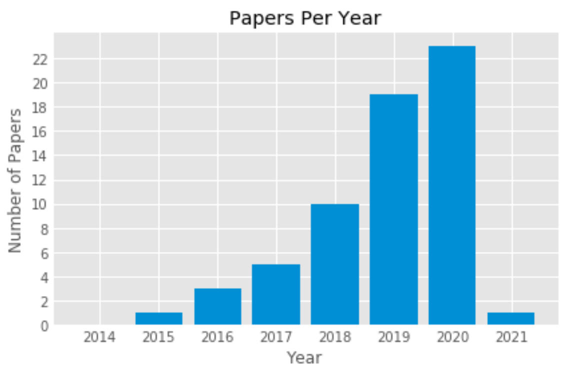

Figure 3: The number of published papers per year since 2013.

3.1 RQ1: Adaptation Targets

Of all the texts that were reviewed, 42% (24 papers) tackled Resource Scheduling or Utilization and Placement, followed by discussions involving Auto-scaling and Elasticity at 21.0% (17 papers), then Context at 11.1% (9 papers), Fault/Anomaly Detection and Exception Handling at 9.9% (8 papers) and Granularity of Services at 3.7% (3 papers). Finally The Other section with 12.3% (11 papers) constitutes solutions tackling Orchestration/Collaboration, Task Workflow/Scheduling, Data Transfer/Filtering/Acquisition, Security/Fire-walling, Configuration/Recovery, Latency and Network Resource Consumption. Figure 4 illustrates the distribution of goals. Table 2 details each goal, its definition and the number of relevant papers.

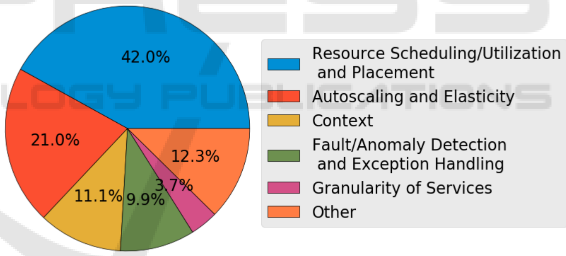

Figure 4: The distribution of the adaptation goals in relevant papers.

We notice from the extracted data that there is a huge focus on optimizing for performance. A combined 63.0% between resource optimization and elasticity. We believe this might be due to the fact that industry professionals are mainly interested in optimizing resources and researchers follow through by proposing solutions for their concrete optimization issues.

3.2 RQ2: Adaptation Level

Looking at the distribution of the adaptation level of papers, we notice that the majority of papers (37 at 59.7%) present a solution acting exclusively on an application level (i.e., solutions dealing with services, containers and applications). Followed by papers (16

Table 2: The different definitions of adaptation goals and the number of papers discussing them.

Adaptation Target	Definition	Number of Papers
Resource Scheduling/Utilization, Placement	Optimizing the use of resources (e.g., CPU, memory, bandwidth, etc.) and the adaptive placement of containers.	34
Auto-scaling and Elasticity	Automatically adjusting the capacity to insure steady performance.	17
Context	The ability to collect data about a system's surrounding environment (e.g., Location, Temperature, etc.).	9
Fault/Anomaly Detection, Exception Handling	Detecting anomalies and errors in the faulty microservices and handling exceptions.	8
Granularity of Services	Identifying the optimal microservice boundaries.	3
Orchestration/Collaboration	Automatic coordination of containers.	2
Task Workflow/Scheduling	Assigning tasks to the proper resources.	2
Data Transfer/Filtering/Acquisition	The ability to manipulate data.	2
Security/Fire-walling	Protection against malicious attacks.	2
Configuration/Recovery	Having the proper system configuration (e.g., reliability, availability, performance, etc.) and the ability to recover after failing.	2
Latency	The delay for data to go from its source to its destination.	1
Network Resource Consumption	The underlying network's resource consumption.	1

at 25.8%) with both an application and architectural (i.e., solutions dealing with how to design and architect your system) approach then purely architectural solutions with 9 papers (14.5%). Figure 5 illustrates this.

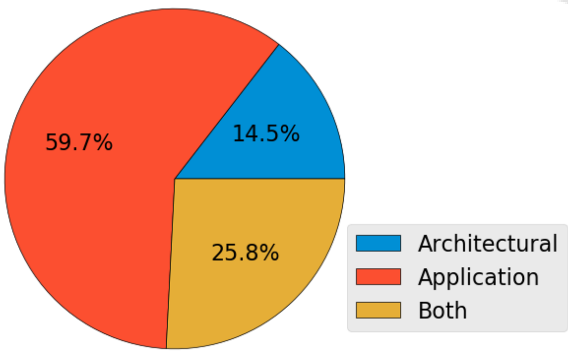

Figure 5: The distribution of the adaptation level in relevant papers.

Since 53 papers (85.5%) have an application level approach, we conclude that most research focuses on solutions that tackle issues on the level of services, applications or containers. This makes sense since there is a wide variety of possible contributions to have contrary to presenting solutions from an architectural and design optic.

3.3 RQ3: Adaptation Techniques

From the present study, we believe that custom techniques are the preferred approach for most researchers. In addition, ML algorithms (i.e., classical machine learning and deep learning) are the techniques of choice for numerous researchers in achieving adaptation targets. 10 papers employ a classical machine learning algorithm in their solution. Reinforcement learning, Deep learning and heuristics are used in 5 papers each. In 4 papers, researchers extend the existing Kubernetes tools. A modified autoscaler is introduced in 2 papers and so is the Non-Dominated Sorting Genetic Algorithm (in order to make a container allocation strategy and automatically manage elasticity). The remaining papers (19) have the following techniques:

- *Description language*: A custom language and an appropriate platform to aid in the implementation of self-adaptive microservices,
- *Resource scheduling algorithms*: Policies used to assign resources accordingly for energy efficiency purposes,

- *External services*: In-house developed software and delegating run-time management to the cloud,
- *Extended Berkeley Packet Filter*: Used to intercept key Linux system calls for efficient monitoring,
- *Scaling policy derivation tool*: Used to design and evaluate 5 auto-scaling policies,
- *Cloud-Edge-Dew architecture*: A custom architecture that optimizes the advantages of Cloud and Edge Computing to contribute microservices for end user devices,
- *Custom orchestrator*: A custom orchestrating tool for microservices,
- *Horizontal offloading mechanism*: A custom mechanism based on the OpenFog reference architecture where fog nodes can horizontally offload computations to multiple less loaded nodes,
- *Hierarchical scalable adaptive cloud monitoring architecture*: Custom architecture that monitors physical and virtual infrastructure, realizes scalability of monitoring based on microservices and adjusts the monitoring interval and data transmission strategy (Wang et al., 2020).
- *Latency aware microservice mashup algorithm*: A custom service mashup approach that focuses on network resource consumption in edge network in addition to efficient service mashup,
- *Micro-controllers*: A controller that can be composed or configured at run-time based on the adaptation goals of the target system,
- *Model based approach*: A custom method called Microservices Recovery Action Selection that adapts to frequent microservices changes without the need for historical data of previous failures,
- *Microservice ambients using aspects*: A modeling concept that considers microservice boundaries as an adaptable first-class entity, it is also based on the aspect-oriented architectural metamodeling approach of ambients (Hassan et al., 2017),
- *Architecture and planning engine at run-time*: An interactive and iterative planning engine that provides insight into which granularity adaptation strategy is adequate at run-time (Hassan, 2019),
- *Custom framework*: A self-adaptive root cause diagnosis framework in order to analyze several metrics collected from given microservices,
- *Event processing techniques*: Used as an integrated part to an appropriate architecture to better smart decision-making,
- *Application partitioning and task offloading algorithm*: Which comes up with an application partitioning decision at run-time,
- *Application-infrastructure co-programming model and architecture*: Which provides a controllable environment for the creation of application logic and enable reconfigurability of computing resources at run-time according to the needs of a specific application (Štefanič et al., 2019) and
- *Vehicular-OBUs-As-On-Demand-Fogs*: A custom framework that efficiently uses clusters of vehicles to benefit from on-board units in order to optimize resources.

3.4 RQ4: Use of Machine Learning

Here we notice that although small, there is a decent interest in using Artificial Intelligence in order to better microservice adaptation. 40.3% (25 papers) of papers utilize either classical machine learning or deep learning algorithms, the remaining 37 papers (59.7%) use other techniques mentioned in section 3.3. The following algorithms were used:

- Data Classification algorithms,
- Support Vector Machine,
- Bayesian Network,
- Decision Tree,
- Naive Bayes classifier,
- Contextual Bandit Reinforcement Learning,
- Stacked Long Short-Term Memory Networks,
- K-means,
- Planning algorithms,
- Extreme Learning Machine,
- Deep Q-Learning,
- Fuzzy Lattice Reasoning,
- Artificial Neural Network and
- Gaussian Process Regression.

Figure 6 illustrates the distribution of what targets researchers go to when utilizing ML as an adaptation technique. We notice that resource optimization is predominately the most common one with 53.1% of total papers that use ML.

3.5 RQ5: Evidence

The level of empirical evidence in the relevant papers is dominated by solution proposals at 54 papers

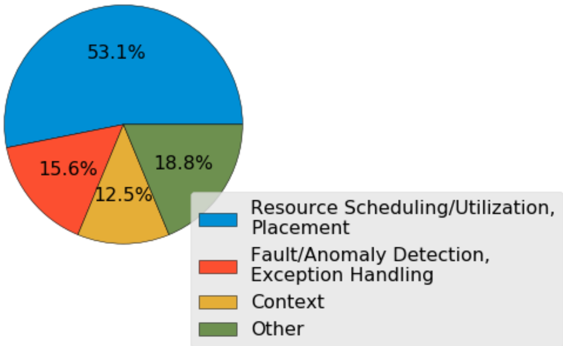

Figure 6: Distribution of the most commonly tackled targets when using ML.

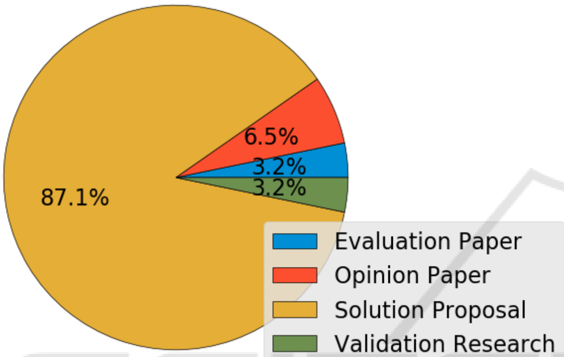

Figure 7: The distribution of types of papers.

(87.1%), followed by 4 (6.5%) opinion papers, 2 evaluation papers and 2 validation research papers (3.2%). Figure 7 shows the distribution.

This means that the research community is mainly focused on solving concrete microservice adaptation problems and supporting their solutions with real examples. The small number of papers (6 at 9.7%) where practical applications are not provided (i.e., validation research, philosophical, opinion and experience papers) indicates that there is virtually no to little to no interest in discussions that do not present a valid practical example.

4 THREATS TO VALIDITY

Assessing Threats to Validity is clinical in order to ensure quality empirical studies in Software Engineering (Zhou et al., 2016).

Construct Validity: The discussion surrounding our main findings in section 3 is only valid for the papers provided. We therefore made sure to include all the possible relevant papers. In order to realize this, we used all the databases that are relevant to software engineering research mentioned in section 2.2. We made sure to answer the appropriate research questions and deduce the adequate complete search terms, we for instance considered all the possible variations of our facets (e.g., microserices, self-adaptation). We also avoided the threat of inappropriate inclusion and exclusion criteria in our screening by considering the tile, abstract and keywords. In addition, we decided as an initial safety measure not to exclude papers that haven't been fully investigated until there has been an exhaustive reading.

Internal Validity: In order to alleviate the issues surrounding data extraction and classification, we took the proper time-frame to, at times, read introductions and conclusion and some times the entirety of a given paper if it wasn't possible to extract data initially from the abstract. We also used an Excel table to properly store the extracted data to generate relevant statistics:

External Validity: We made sure not to restrict the time span of the resulting studies. To improve external validity in the future it is perhaps advised to have full access to all relevant papers by contacting their respective authors.

Conclusion Validity: The study's replicability is possible thanks to the search method details provided in section 2. The threat of primary study duplication was also avoided by using the open search tool Zotero and always double checking our data.

5 RELATED WORK

Numerous systematic reviews tackled self-adaptation. One paper outlines the machine learning techniques used to approach self adaptation based on the concerns, aspects and purposes of choice (Saputri and Lee, 2020), although exhaustive we believe it wasn't applied to a specific type of architecture. Multiple papers discuss self-adaptation through the lens of a specific context (e.g., mobile app (Grua et al., 2019), cyber-physical system (Muccini et al., 2016), etc.). Although there is a discussion of self-adaptation in Service Oriented Architecture (Romay et al., 2011) there are no equivalent for the Microservices Architecture. To the best of our knowledge there were no Systematic Mapping Reviews discussing the use of ML techniques for the adaptation of microservices.

6 CONCLUSIONS

In this paper, we provide the results concerning the systematic mapping study of the use of ML in Microservice adaptation. this paper outlines important observations:

- recognizing the most focused on microservices adaptation targets in the research community;

- highlighting the techniques of choice that authors go to;
- figuring out the level of empirical evidence achieved by the researchers' presented solutions;

Our study, with the help of the developed classification scheme, provides the most commonly tackled issues in microservice adaptation targets; resource optimization and elasticity. It also indicates that although many authors decide to develop a unique adaptation technique, there is also a growing interest in using ML algorithms as techniques to achieve adaptation targets. Our findings also reflect that the majority of researchers provide practical solutions to concrete issues.

Our objective was to provide a comprehensive mapping of the use of ML in microservices adaptation. We intend to go more in depth and specify the strengths and weaknesses of specific techniques in specific contexts, explore opportunities where adaptation techniques can intersect and be more efficient at general self-adaptation and finally add to the growing discussion by introducing concepts that might help us make abstractions to adaptation targets and build self-adaptive microservices that comprise of all the main goals (i.e., self configuration, self-optimization, self-healing and self-protection), which will be our future work.

REFERENCES

Abdullah, M., Iqbal, W., Bukhari, F., and Erradi, A. (2020). Diminishing Returns and Deep Learning for Adaptive CPU Resource Allocation of Containers. *IEEE Transactions on Network and Service Management*, 17(4):2052–2063. Conference Name: IEEE Transactions on Network and Service Management.

Barna, C., Khazaei, H., Fokaefs, M., and Litoiu, M. (2017). Delivering Elastic Containerized Cloud Applications to Enable DevOps. In *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 65–75.

Bellur, U., Narendra, N. C., and Mohalik, S. K. (2017). AUSOM: Autonomic Service-Oriented Middleware for IoT-Based Systems. In *2017 IEEE World Congress on Services (SERVICES)*, pages 102–105.

Breton, P., Kitchenman, B., Budgen, D., Turner, M., and Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. j syst softw. In *Journal of Systems and Software*.

Casalicchio, E. (2019). A study on performance measures for auto-scaling CPU-intensive containerized applications. *Cluster Comput*, 22(3):995–1006.

Chegini, H. and Mahanti, A. (2019). A Framework of Automation on Context-Aware Internet of Things (IoT) Systems. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, UCC '19 Companion, pages 157–162, New York, NY, USA. Association for Computing Machinery.

Coulson, N. C., Sotiriadis, S., and Bessis, N. (2020). Adaptive Microservice Scaling for Elastic Applications. *IEEE Internet of Things Journal*, 7(5):4195–4202. Conference Name: IEEE Internet of Things Journal.

De Sanctis, M., Bucchiarone, A., and Marconi, A. (2020). Dynamic adaptation of service-based applications: a design for adaptation approach. *J Internet Serv Appl*, 11(1):2.

Donca, I., Corches, C., Stan, O., and Miclea, L. (2020). Autoscaled RabbitMQ Kubernetes Cluster on single-board computers. In *2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–6.

Florio, L. and Nitto, E. D. (2016). Gru: An Approach to Introduce Decentralized Autonomic Behavior in Microservices Architectures. In *2016 IEEE International Conference on Autonomic Computing (ICAC)*, pages 357–362.

Fowler, M. (2014). Microservices.

Grua, E. M., Malavolta, I., and Lago, P. (2019). Self-adaptation in mobile apps: a systematic literature study. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*.

Guerrero, C., Lera, I., and Juiz, C. (2018a). Genetic Algorithm for Multi-Objective Optimization of Container Allocation in Cloud Architecture. *J Grid Computing*, 16(1):113–135.

Guerrero, C., Lera, I., and Juiz, C. (2018b). Resource optimization of container orchestration: a case study in multi-cloud microservices-based applications. *J Supercomput*, 74(7):2956–2983.

Hassan, S. (2019). *Modelling and evaluation of microservice granularity adaptation decisions*. PhD Thesis, University of Birmingham.

Hassan, S., Ali, N., and Bahsoon, R. (2017). Microservice Ambients: An Architectural Meta-Modelling Approach for Microservice Granularity. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 1–10.

Hassan, S. and Bahsoon, R. (2016). Microservices and Their Design Trade-Offs: A Self-Adaptive Roadmap. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 813–818.

Herrera, J. and Moltó, G. (2020). Toward Bio-Inspired Auto-Scaling Algorithms: An Elasticity Approach for Container Orchestration Platforms. *IEEE Access*, 8:52139–52150. Conference Name: IEEE Access.

Houmani, Z., Balouek-Thomert, D., Caron, E., and Parashar, M. (2020). Enhancing microservices architectures using data-driven service discovery and QoS guarantees. In *2020 20th IEEE/ACM International*

Symposium on Cluster, Cloud and Internet Computing (CCGRID), pages 290–299.

Imdoukh, M., Ahmad, I., and Alfailakawi, M. G. (2020). Machine learning-based auto-scaling for containerized applications. *Neural Comput & Applic*, 32(13):9745–9760.

Jiménez, L. L. and Schelén, O. (2019). DOCMA: A Decentralized Orchestrator for Containerized Microservice Applications. In *2019 IEEE Cloud Summit*, pages 45–51.

Junior, A. (2018). Runtime adaptation of microservices.

Kang, P. and Lama, P. (2020). Robust Resource Scaling of Containerized Microservices with Probabilistic Machine learning. In *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pages 122–131.

Keni, N. D. and Kak, A. (2020). Adaptive Containerization for Microservices in Distributed Cloud Systems. In *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6. ISSN: 2331-9860.

Kephart, J. and Chess, D. (2003). The vision of autonomic computing. IEEE.

Khazaei, H., Ghanbari, A., and Litoiu, M. (2018a). Adaptation as a service. In *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*. IBM Corp.

Khazaei, H., Ghanbari, A., and Litoiu, M. (2018b). Adaptation as a service. In *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*, CASCON '18, pages 282–288, USA. IBM Corp.

Kitchenman, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Klinaku, F., Frank, M., and Becker, S. (2018). CAUS: An Elasticity Controller for a Containerized Microservice. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, ICPE '18, pages 93–98, New York, NY, USA. Association for Computing Machinery.

Kumar, J. and Singh, A. K. (2020). Decomposition Based Cloud Resource Demand Prediction Using Extreme Learning Machines. *J Netw Syst Manage*, 28(4):1775–1793.

Lakhan, A. and Li, X. (2020). Transient fault aware application partitioning computational offloading algorithm in microservices based mobile cloudlet networks. *Computing*, 102(1):105–139.

Ma, M., Lin, W., Pan, D., and Wang, P. (2019). MS-Rank: Multi-Metric and Self-Adaptive Root Cause Diagnosis for Microservice Applications. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 60–67.

Magableh, B. (2016). Deep Q Learning for Self Adaptive Distributed Microservices Architecture (in press). 4:17.

Meixner, S., Schall, D., Li, F., Karagiannis, V., Schulte, S., and Plakidas, K. (2019). Automatic Application Placement and Adaptation in Cloud-Edge Environments. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1001–1008. ISSN: 1946-0759.

Mendonça, N., Jamshidi, P., Garlan, D., and Pahl, C. (2019). Developing self-adaptive microservice systems: Challenges and directions. In *IEEE Software 2019*. IEEE.

Mostafa, M. A. A. and Khater, A. M. (2019). Horizontal Offloading Mechanism for IoT Application in Fog Computing Using Microservices Case Study: Traffic Management System. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, pages 640–647.

Muccini, H., Sharaf, M., and Weyns, D. (2016). Self-adaptation for cyber-physical systems: a systematic literature review. In *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. Association for Computing Machinery.

Nabi, S. and Ahmed, M. (2021). OG-RADL: overall performance-based resource-aware dynamic loadbalancer for deadline constrained Cloud tasks. *J Supercomput*.

Neves, F., Vilaça, R., and Pereira, J. (2020). Black-box inter-application traffic monitoring for adaptive container placement. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, pages 259–266, New York, NY, USA. Association for Computing Machinery.

Nguyen, P. and Nahrstedt, K. (2017). MONAD: Self-Adaptive Micro-Service Infrastructure for Heterogeneous Scientific Workflows. In *2017 IEEE International Conference on Autonomic Computing (ICAC)*, pages 187–196. ISSN: 2474-0756.

Orsini, G., Posdorfer, W., and Lamersdorf, W. (2019). Efficient Mobile Clouds: Forecasting the Future Connectivity of Mobile and IoT Devices to Save Energy and Bandwidth. *Procedia Computer Science*, 155:121–128.

Ortiz, G., Caravaca, J. A., García-de Prado, A., O, F. C. d. l., and Boubeta-Puig, J. (2019). Real-Time Context-Aware Microservice Architecture for Predictive Analytics and Smart Decision-Making. *IEEE Access*, 7:183177–183194. Conference Name: IEEE Access.

Pahl, M. and Aubet, F. (2018). All Eyes on You: Distributed Multi-Dimensional IoT Microservice Anomaly Detection. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 72–80. ISSN: 2165-963X.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *EASE'08: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, Swindon, GBR. BCS Learning & Development Ltd.

Podolskiy, V., Jindal, A., Gerndt, M., and Oleynik, Y. (2018). Forecasting Models for Self-Adaptive Cloud Applications: A Comparative Study. In *2018 IEEE 12th International Conference on Self-Adaptive and*

Self-Organizing Systems (SASO), pages 40–49. ISSN: 1949-3681.

Ramirez, Y. M., Podolskiy, V., and Gerndt, M. (2019). Capacity-Driven Scaling Schedules Derivation for Coordinated Elasticity of Containers and Virtual Machines. In *2019 IEEE International Conference on Autonomic Computing (ICAC)*, pages 177–186. ISSN: 2474-0756.

Ravandi, B. and Papapanagiotou, I. (2018). A self-organized resource provisioning for cloud block storage. *Future Generation Computer Systems*, 89:765–776.

Rodríguez-Gracia, D., Piedra-Fernández, J. A., Iribarne, L., Criado, J., Ayala, R., Alonso-Montesinos, J., and Maria de las Mercedes, C.-U. (2019). Microservices and Machine Learning Algorithms for Adaptive Green Buildings. *Sustainability*, 11(16):4320. Publisher: Multidisciplinary Digital Publishing Institute.

Romay, M. P., Fernandez-Sanz, L., and Rodriguez, D. (2011). A systematic review of self-adaptation in service-oriented architectures.

Rossi, F., Cardellini, V., and Presti, F. L. (2020a). Hierarchical Scaling of Microservices in Kubernetes. In *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 28–37.

Rossi, F., Cardellini, V., and Presti, F. L. (2020b). Self-adaptive Threshold-based Policy for Microservices Elasticity. In *2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 1–8. ISSN: 2375-0227.

Rovnyagin, M. M., Guminskaia, A. V., Plyukhin, A. A., Orlov, A. P., Chernilin, F. N., and Hrapov, A. S. (2018). Using the ML-based architecture for adaptive containerized system creation. In *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 358–360.

Rychener, L., Montet, F., and Hennebert, J. (2020). Architecture Proposal for Machine Learning Based Industrial Process Monitoring. *Procedia Computer Science*, 170:648–655.

Sahni, J. and Vidyarthi, D. P. (2017). Heterogeneity-aware adaptive auto-scaling heuristic for improved QoS and resource usage in cloud environments. *Computing*, 99(4):351–381.

Sami, H., Mourad, A., and El-Hajj, W. (2020). Vehicular-OBUs-As-On-Demand-Fogs: Resource and Context Aware Deployment of Containerized Micro-Services. *IEEE/ACM Transactions on Networking*, 28(2):778–790. Conference Name: IEEE/ACM Transactions on Networking.

Sampaio, A. R., Rubin, J., Beschastnikh, I., and Rosa, N. S. (2019). Improving microservice-based applications with runtime placement adaptation. *J Internet Serv Appl*, 10(1):4.

Sanctis, M., Muccini, H., and Vaidhyanathan, K. (2020a). Data-driven adaptation in microservice-based iot architectures. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*.

Sanctis, M. D., Muccini, H., and Vaidhyanathan, K. (2020b). Data-driven Adaptation in Microservice-based IoT Architectures. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 59–62.

Saputri, T. R. D. and Lee, S.-W. (2020). The application of machine learning in self-adaptive systems: A systematic literature review. In *IEEE Access*. IEEE Access.

Siqueira, B. R., Ferrari, F. C., Vogel, T., and Lemos, R. d. (2020). Micro-controllers: Promoting Structurally Flexible Controllers in Self-Aware Computing Systems. In *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pages 188–193.

Štefanič, P., Cigale, M., Jones, A. C., Knight, L., Taylor, I., Istrate, C., Suciu, G., Ulisses, A., Stankovski, V., Taherizadeh, S., Salado, G. F., Koulouzis, S., Martin, P., and Zhao, Z. (2019). SWITCH workbench: A novel approach for the development and deployment of time-critical microservice-based cloud-native applications. *Future Generation Computer Systems*, 99:197–212.

Tefera, G., She, K., and Deeba, F. (2019). Decentralized Adaptive Latency-Aware Cloud-Edge-Dew Architecture for Unreliable Network. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, ICMLC '19, pages 142–146, New York, NY, USA. Association for Computing Machinery.

Wang, R., Ying, S., Li, M., and Jia, S. (2020). HSACMA: a hierarchical scalable adaptive cloud monitoring architecture. *Software Qual J*, 28(3):1379–1410.

Wang, W., Fan, L., Huang, P., and Li, H. (2019). A New Data Processing Architecture for Multi-Scenario Applications in Aviation Manufacturing. *IEEE Access*, 7:83637–83650. Conference Name: IEEE Access.

Wang, X., Feng, Z., and Huang, K. (2018). D3L-Based Service Runtime Self-Adaptation Using Replanning. *IEEE Access*, 6:14974–14995. Conference Name: IEEE Access.

Wang, Y. (2019). Towards service discovery and autonomic version management in self-healing microservices architecture. In *Proceedings of the 13th European Conference on Software Architecture - Volume 2*, ECSA '19, pages 63–66, New York, NY, USA. Association for Computing Machinery.

Wanigasekara, N. (2015a). A semi lazy bandit approach for intelligent service discovery in iot applications. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*. Association for Computing Machinery.

Wanigasekara, N. (2015b). A semi lazy bandit approach for intelligent service discovery in IoT applications. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, UbiComp/ISWC'15 Adjunct, pages 503–508, New York, NY, USA. Association for Computing Machinery.

Wieringa, R., Maiden, N., Mead, N., and Rolland, C. (2006). Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. In *Requir. Eng.*

Wu, L., Tordsson, J., Acker, A., and Kao, O. (2020). MicroRAS: Automatic Recovery in the Absence of Historical Failure Data for Microservice Systems. In *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pages 227–236.

Xu, M., Toosi, A. N., and Buyya, R. (2020). A Self-adaptive Approach for Managing Applications and Harnessing Renewable Energy for Sustainable Cloud Computing. *IEEE Transactions on Sustainable Computing*, pages 1–1. Conference Name: IEEE Transactions on Sustainable Computing.

Yang, Z., Nguyen, P., Jin, H., and Nahrstedt, K. (2019). MIRAS: Model-based Reinforcement Learning for Microservice Resource Allocation over Scientific Workflows. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 122–132. ISSN: 2575-8411.

Yudong, L., Yuqing, Z., and Zhangbin, Z. (2020). Service Availability Guarantee with Adaptive Automatic Flow Control. In *2020 IEEE World Congress on Services (SERVICES)*, pages 101–105. ISSN: 2642-939X.

Zang, X., Chen, W., Zou, J., Zhou, S., Lisong, H., and Ruigang, L. (2018). A Fault Diagnosis Method for Microservices Based on Multi-Factor Self-Adaptive Heartbeat Detection Algorithm. In *2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pages 1–6.

Zhang, S., Zhang, M., Ni, L., and Liu, P. (2019). A Multi-Level Self-Adaptation Approach For Microservice Systems. In *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 498–502.

Zheng, T., Zheng, X., Zhang, Y., Deng, Y., Dong, E., Zhang, R., and Liu, X. (2019). SmartVM: a SLA-aware microservice deployment framework. *World Wide Web*, 22(1):275–293.

Zhou, A., Wang, S., Wan, S., and Qi, L. (2020). LMM: latency-aware micro-service mashup in mobile edge computing environment. *Neural Comput & Applic*, 32(19):15411–15425.

Zhou, X., Jin, Y., Zhang, H., Li, S., and Huang, X. (2016). A map of threats to validity of systematic literature reviews in software engineering. In *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. IEEE.

APPENDIX

As mentioned in section 2.5, table 3 represents the detailed scheme we used to conduct our SMR.

Table 3: The Systematic Mapping Review's detailed scheme.

Number	Paper	Year	Target	Technique	Evidence	Use of ML	Architecture level	Application level
1	(Zang et al., 2018)	2018	Fault detection	Multi-factor self-adaptive heartbeat detection algorithm	Solution proposal	No	No	Yes
2	(Chegini and Mahanti, 2019)	2019	Context, orchestration and collaboration, task workflow and scheduling, data transfer and filtering	Knowledge or rule-based component, Data classification algorithms (ML) and data science and feature engineering	Evaluation paper	Yes	No	Yes
3	(Zhang et al., 2019)	2019	Service and instance numbers	Microservice self-adaptation description language and adaptive K8 (Extending Kubernetes)	Solution proposal	No	No	Yes
4	(Wang et al., 2019)	2019	Data acquisition	Big Data processing	Solution proposal	No	Yes	No
5	(Xu et al., 2020)	2020	Resource scheduling	Resource scheduling algorithms, SVM to predict solar irradiation or PV power output for the availability of renewable energy	Solution proposal	Yes	No	Yes
6	(Ravandi and Papapanagiotou, 2018)	2018	Resource utilization	Bayesian network, decision tree, Naive bayes and Support Vector Machine	Solution proposal	Yes	No	Yes
7	(Wanigasekara, 2015b)	2015	Context	Contextual Bandit Reinforcement Learning algorithms	Solution proposal	Yes	No	Yes
8	(Casalicchio, 2019)	2019	Resource utilization, scaling	KHPA-A algorithm (Extending Kubernetes's autoscaling algorithm based on CPU usage)	Solution proposal	No	No	Yes
9	(Khazaei et al., 2018b)	2018	Security, configuration, fault detection and resource optimization	Adaptation as external services	Solution proposal	Yes	Yes	No
10	(Keni and Kak, 2020)	2020	SLA based resource allocation	Optimal containerization framework using mathematical representations	Solution proposal	No	No	Yes
11	(Coulson et al., 2020)	2020	Resource allocation and auto-scaling	Stacked Long Short-Term Memory Network	Solution proposal	Yes	No	Yes
12	(Pahl and Aubet, 2018)	2018	Anomaly detection and fire-walling	Grid-based algorithm, k-mea,s	Solution proposal	Yes	No	Yes
13	(Rychener et al., 2020)	2020	Anomaly detetion	Machine learning	Opinion paper	Yes	Yes	Yes
14	(Bellur et al., 2017)	2017	Context, application requirement	Service-oriented middleware	Solution paper	No	Yes	Yes
15	(Meixner et al., 2019)	2019	Resource and placement	NFRs based on user defined rules for placement and data-driven automatic deployment	Solution proposal	Yes	Yes	No
16	(Donca et al., 2020)	2020	Resource and auto-scaling	Kubernetes auto-scaler and Raspberry Pi	Solution proposal	No	Yes	No
17	(Neves et al., 2020)	2020	Adaptive placement	Extended Berkeley Packet Filter	Solution proposal	No	No	Yes
18	(Ramirez et al., 2019)	2019	Auto-scaling	Scaling Policy Derivation Tool	Solution proposal	No	No	Yes
19	(Klinaku et al., 2018)	2018	Resource and auto-scaling	Heuristics	Solution proposal	No	No	Yes
20	(Wang et al., 2018)	2018	Exception handling	AI Planning algorithms	Solution proposal	Yes	No	Yes
21	(Sanctis et al., 2020b)	2020	Resource	Machine Learning, Q-learning, AI Planning	Validation research	Yes	Yes	Yes
22	(Tefera et al., 2019)	2019	Latency	Cloud-Edge-Dew Architecture	Validation Research	No	Yes	No
23	(Kumar and Singh, 2020)	2020	Resource	Extreme Learning Machine	Solution proposal	Yes	No	Yes
24	(Magableh, 2016)	2016	Anomaly detection	Deep Q Learning, Reinforcement Learning	Solution proposal	Yes	Yes	Yes
25	(Barna et al., 2017)	2017	Auto-scaling	Self-tuning performance model and custom autonomic management system	Solution proposal	No	Yes	No
26	(Abdullah et al., 2020)	2020	Resource	Deep Learning	Solution proposal	Yes	No	Yes
27	(Jiménez and Schelén, 2019)	2019	Resource	Custom orchestrator; DOCMA	Solution proposal	No	Yes	Yes
28	(De Sanctis et al., 2020)	2020	Resource	Architecture	Solution proposal	Yes	Yes	No
29	(Orsini et al., 2019)	2019	Context	Machine Learning	Solution proposal	Yes	No	Yes
30	(Houmani et al., 2020)	2020	Resource	Custom auto-scaler, scale-up/down and load shedding algorithm	Solution proposal	No	Yes	Yes
31	(Podolskiy et al., 2018)	2018	Auto-scaling, resource	Machine Learning, ARIMA, GARCH SSA, SVR	Solution proposal	Yes	No	Yes
32	(Guerrero et al., 2018a)	2018	Resource, container allocation and elasticity management	Non-dominated Sorting Genetic Algorithm II	Solution proposal	No	No	Yes
33	(Florio and Nitto, 2016)	2016	Auto-scaling, resource	Autonomic manager, MAPE-K	Solution proposal	No	No	Yes
34	(Sahni and Vidyarthi, 2017)	2017	Auto-scaling	Heuristics	Solution proposal	No	No	Yes
35	(Rossi et al., 2020a)	2020	Elasticity, Auto-scaling	Custom Auto-scaler, Kubernetes extension, RL based scaling	Solution proposal	Yes	No	Yes
36	(Mostafa and Khater, 2019)	2019	Context	Horizontal offloading mechanism	Solution proposal	No	Yes	Yes
37	(Wang et al., 2020)	2020	Resource	HSACMA (Hierarchical Scalable Adaptive Cloud Monitoring Architecture)	Solution proposal	No	Yes	Yes
38	(Sampaio et al., 2019)	2019	Resource, placement	REMap (custom adaptation mechanism)	Solution proposal	No	No	Yes
39	(Zhou et al., 2020)	2020	Network resource consumption	Latency aware microservice mashup algorithm	Solution proposal	No	No	Yes
40	(Imdoukh et al., 2020)	2020	Auto-scaling, resource	LSTM on different data types	Solution proposal	Yes	No	Yes
41	(Siqueira et al., 2020)	2020	Context	Micro-controllers	Solution proposal	No	No	Yes
42	(Wu et al., 2020)	2020	Recovery	Model based approach	Solution proposal	No	No	Yes
43	(Hassan et al., 2017)	2017	Granularity of services	MS Ambients using Aspects	Solution proposal	No	Yes	No
44	(Rodríguez-Gracia et al., 2019)	2019	Context, energy consumption	ML for decision making, Fuzzy Lattice Reasoning (FLR)	Solution proposal	Yes	Yes	Yes
45	(Hassan and Bahsoon, 2016)	2016	Granularity of services	Modified MAPE-K	Opinion paper	No	No	Yes
46	(Yang et al., 2019)	2019	Resource	Reinforcement Learning	Solution proposal	Yes	No	Yes
47	(Hassan, 2019)	2019	Granularity of services	Architecture and planning engine for insight at runtime	Solution proposal	No	Yes	Yes
48	(Nguyen and Nahrstedt, 2017)	2017	Resource, scheduling	Artificial Neural Network for system identification	Solution proposal	Yes	Yes	No
49	(Ma et al., 2019)	2019	Anomaly detection	Custom framework, algorithm	Solution proposal	No	No	Yes
50	(Nabi and Ahmed, 2021)	2021	Resource and Task scheduling	Heuristics	Solution proposal	No	No	Yes
51	(Ortiz et al., 2019)	2019	Context	Event processing techniques	Solution proposal	No	Yes	No
52	(Guerrero et al., 2018b)	2018	Resource	Non-dominated Sorting Genetic Algorithm	Evaluation paper	No	No	Yes
53	(Kang and Lama, 2020)	2020	Resource	Probabilistic Machine Learning based models; Gaussian Process Regression	Solution proposal	Yes	No	Yes
54	(Rossi et al., 2020b)	2020	Auto-scaling, resource	Reinforcement Learning to calculate threshold	Solution proposal	Yes	No	Yes
55	(Yudong et al., 2020)	2020	Resource	Heuristics	Solution proposal	No	No	Yes
56	(Zheng et al., 2019)	2019	Resource, auto-scaling	Heuristics	Solution proposal	No	Yes	Yes
57	(Štefanič et al., 2019)	2019	Resource, auto-scaling	Architecture	Solution proposal	No	Yes	Yes
58	(Herrera and Moltó, 2020)	2020	Resource, auto-scaling	Bio inspired algorithms	Solution proposal	No	No	Yes
59	(Wang, 2019)	2019	Resource, auto-scaling	Architecture	Opinion paper	No	Yes	Yes
60	(Lakhan and Li, 2020)	2020	Resource, auto-scaling	Custom Algorithm; application partitioning task offloading algorithm	Solution proposal	No	No	Yes
61	(Rovnyagin et al., 2018)	2018	Container orchestration	ML Engine for metrics analysis	Opinion paper	Yes	Yes	Yes
62	(Sami et al., 2020)	2020	Resource, Context	Architecture, custom algorithm	Solution proposal	No	Yes	Yes