A Review of Software Architecture Optimization Approaches for Cloud Applications

Anton Frisch[1], Robin Lichtenthäler[1]

[1]Distributed Systems Group, University of Bamberg, Bamberg, Germany

Abstract

Developing applications and deploying them on cloud platforms is a common approach. With the continuous evolution of cloud computing and the wide range of technological options, however, making architectural decisions while developing cloud applications can become difficult. One possibility to support the development of cloud applications are software architecture optimization approaches that can evaluate and optimize an architecture according to specific goals. This work provides an up-to-date review of currently existing architecture optimization approaches specifically for cloud applications. Based on the review common optimization goals and approaches are identified and the potential for future work is analyzed.

Keywords

cloud application, software architecture optimization, taxonomy

1. Introduction

Cloud computing is a mature concept that is widely used in the software industry [1]. However, since the first cloud offerings were published, cloud computing has evolved [1] and now a wide range of different platforms and services are available. For developing cloud applications it is thus necessary to make architectural choices that, on top of fulfilling functional requirements, also fit the inherent characteristics of cloud computing. This means on the one hand taking advantage of cloud computing benefits and on the other hand preparing for inherent issues of cloud environments. Applications specifically built to fit into cloud environments are also called *cloud-native applications (CNA)* [2]. Architecting CNAs means composing a distributed and scalable system out of (micro)services, using cloud-focused design patterns, and operating it on an elastic cloud platform [2]. Providing support for these challenging tasks thus is desirable. One possibility are so-called architecture optimization approaches [3]. These approaches are able to evaluate the architecture of an application according to certain quality goals, propose potential changes, and even implement a selected change in an architecture. The overall research field of architecture optimization approaches has been reviewed by Aleti et al. in 2013 [3] together with the development of a taxonomy to classify optimization approaches. However, the main impact of cloud computing has occurred after that and a review of architecture optimization approaches with a specific focus on cloud applications is missing in the literature to the best of our knowledge. The aim of this work therefore is to fill this gap. Based on a literature review, the taxonomy of Aleti et al. [3] is adapted to the context of cloud applications, and identified approaches are classified based on it. The guiding research questions thus are:

RQ1: How can the current architecture optimization approaches for cloud applications be classified using a structured taxonomy?

RQ2: What is the current state of software architecture optimization research for cloud applications with respect to this classification?

To answer these research questions, based on the foundations presented in Section 2, we outline our methodology in Section 3. The results of applying this methodology are presented in Section 4 and discussed in Section 5, before a final conclusion in Section 6.

Vienna'25: 17th Central European Workshop on Services and their Composition (ZEUS), February 20–21, 2025, Vienna, Austria

anton-liam.frisch@stud.uni-bamberg.de (A. Frisch); robin.lichtenthaeler@uni-bamberg.de (R. Lichtenthäler)

0000-0002-9608-619X (R. Lichtenthäler)

[1]https://www.oreilly.com/radar/the-cloud-in-2021-adoption-continues/

S. Böhm and M. Vidgof (Eds.): 17th ZEUS Workshop, ZEUS 2025, Vienna, Austria, 20–21 February, 2025 published at http://ceur-ws.org

2. Cloud Application Software Architectures

Software Architecture is a practice for understanding and managing large-scale structures of software systems [4]. A core foundation for it is the observation that "*it isn't enough for a computer program to produce the correct outcome. Other software qualities such as dependability and maintainability are also important and can be achieved by careful structuring.*"[4]. Relevant methods include notations to capture system characteristics or technological decisions, techniques to analyze systems, and tools that facilitate and automate these tasks. The specific methods used by software architects, however, depend on the domain and the quality aspects in focus. Aleti et al. [3] had a broad domain scope and thus relied on a more general definition of architecture. According to the ISO42010 standard, an *architecture* is defined as the "*fundamental concepts or properties of an entity in its environment and governing principles for the realization and evolution of this entity and its related life cycle processes*" [5].

Cloud applications, however, represent a more specific domain for which different aspects may be relevant than for other types of applications. Thus, it is justified to consider cloud application software architectures as a more specific type of architecture for which specific methods can be developed and applied. Important aspects are named by Pahl et al. [6] who define a cloud architecture as "*an abstract model of a distributed cloud system with the appropriate elements to represent not only application components and their interrelationships, but also the resources these components are deployed on and the respective management elements*" [6]. These components, their interrelationships, and the resources on which components are deployed have been described by Kratzke and Peinl [7] in more detail and structured based on a set of layers, from the host layer, over a cluster layer to services and application layers. Implementing so-called cloud-native applications [2] means ensuring certain characteristics for the architecture of an application across the different elements and layers of a cloud system with the goal of ensuring certain quality aspects. Implementing an application in a cloud-native way therefore has the same goal as architecture optimization which is defined by Aleti et al. as "*an automated method that aims to achieve an optimal architecture design with respect to certain quality attributes*" [3].

The key point in architecture optimization is that an automated method is used. This automated method should be able to take a representation of the architecture of an application as input. On this input, a quality evaluation mechanism has to quantify the current state of an architecture and analyze potentials for optimization. The required changes to optimize an architecture should ideally also be applicable to the application in a structured, automated way. To summarize, the focus of this work is on such automated methods in the specific domain of cloud applications software architectures. That means in this domain a focus is set on the components, typically services, how they communicate, and how they are deployed in the cloud.

3. Methodology

The research method for this study adapts the approach from Aleti et al. [3] and the guidelines for systematic reviews by Kitchenham et al. [8]. This section presents the study selection criteria, followed by the search strategy and the data extraction and synthesis process. To select relevant publications, the inclusion and exclusion criteria by Aleti et al. were slightly adapted to focus on the domain of cloud applications as described in Section 2. Three inclusion criteria must be satisfied by each selected publication. Firstly, a machine-processable representation of the software architecture must be provided which the approach receives as input. Secondly, a quality evaluation mechanism must be defined, either to assess relevant quality attributes or to ensure that quality attributes and constraints are inherently satisfied. Lastly, degrees of freedom must be defined, meaning that it must be described how the approach can modify a given software architecture to achieve the optimization goal. In addition, works are excluded if they: 1) focus on optimizing a single component in an application without integrating context and interactions with other components; 2) discuss topics not directly related to software architecture, e.g., compiler optimization or hardware-specific optimizations; 3) focus on optimizing hardware instead of software.

The inclusion and exclusion criteria were applied in three stages: a title screening, an abstract review, and a full-text evaluation. In each stage, if for a publication the inclusion and exclusion criteria could not be clearly determined it was moved to the next stage.

The applied search strategy is summarized here and additional information can be found online[2]. As shown in Figure 1, a broad search across Google Scholar, IEEE Xplore, and ACM Digital Library was used to identify initial relevant studies and to guide the further process. This identified seven relevant papers and using the publishing sources, the following four databases were selected for a more targeted search: IEEE Xplore, ACM Digital Library, Elsevier ScienceDirect, and Springer Link. This targeted search applied more specific search strings and revealed additional three relevant studies. Together with two works previously known to the authors, a subsequent backward search using these twelve was done, identifying six more. Finally, a forward search revealed one additional publication, leading to a total of 19 relevant publications that satisfy the inclusion criteria, shown in Table 1. It has to be noted that in relation to the amount of search results, not many publications satisfied all inclusion criteria.

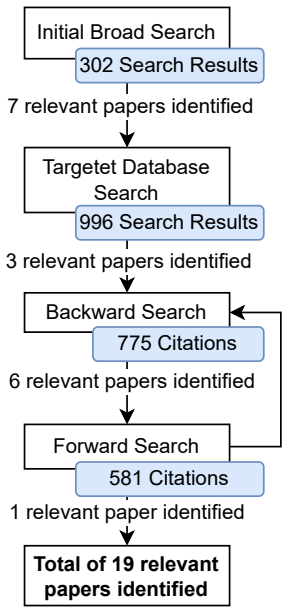

Figure 1: Search Process

Table 1
Identified primary literature for the review

Ref.	Title Authors	Published	Found By	Citing
[9]	**Exploring Sustainable Alternatives for Microservices Deployment in the Cloud** V. Cortellessa, D. Di Pompeo, M. Tucci	2024, IEEE	Database Search (IEEE Xplore)	
[10]	**The μTOSCA Toolchain: Mining, Analyzing, and Refactoring Microservices** J. Soldani, G. Muntoni, D., A. Brogi	2021, Wiley	Known Before	
[11]	**Modeling and Optimization of Performance and Cost of Serverless Applications** C. Lin, H. Khazaei	2020, IEEE	Known Before	
[12]	**Architectural Design of Cloud Applications: A Performance-Aware Cost Minimization Approach** M. Ciavotta, G. P. Gibilisco, D. Ardagna, E. Di Nitto, M. Lattuada	2020, IEEE	Initial Search (Google Scholar)	[13, 14, 15, 16]
[17]	**Performance Optimization of Cloud Applications at Architecture** L. X. Du, Y. Ni, P. Ye, X. Wang, R. Xiao	2019, Springer	Database Search (Springer Link)	[18, 15]
[19]	**Modeling Optimal and Automatized Cloud Application Deployment** S. De Gouw, J. Mauro, G. Zavattaro	2019, Elsevier	Initial Search (Google Scholar)	[20, 21, 22]
[16]	**Multi-Objective Optimization of Deployment Topologies** F. Willnecker, H. Krcmar	2018, ACM	Initial Search (Google Scholar)	[13, 18, 23]
[24]	**ElaClo: A Framework for Optimizing Software Application Topology in the Cloud** N. Tanković, T. G. Grbac, M. Žaga	2017, Elsevier	Initial Search (Google Scholar)	[13, 20, 25, 26, 14]
[15]	**A Mixed Integer Linear Programming Approach for Multi-Cloud Capacity Allocation** M. Ciavotta, D. Ardagna, G. P. Gibilisco	2017, Elsevier	Database Search (ScienceDirect)	[25, 18]
[22]	**Zephyrus2: On the Fly Deployment Optimization Using SMT and CP Technologies** E. Ábrahám, F. Corzilius, E. B. Johnsen	2016, Springer	Backward Search	[25, 21]
[23]	**Optimization of Deployment Topologies for Distributed Applications** F. Willnecker, H. Krcmar	2016, IEEE	Backward Search	[13, 18]
[14]	**Palladio Optimization Suite: QoS Optimization for Component-Based Cloud Apps** M. Ciavotta, M. Ardagna, A. Koziolek	2016, ACM	Backward Search	
[27]	**A Model-Driven DevOps Framework for QoS-Aware Cloud Applications** M. Guerriero, M. Ciavotta, G. P. Gibilisco, D. Ardagna	2015, IEEE	Initial Search (Google Scholar)	[25, 18]
[26]	**A Multi-objective ACS Algorithm for Cost, Performance, and Reliability Optimization** A. Ashraf, B. Byholm, I. Porres	2015, IEEE	Forward Search	
[21]	**Automated Synthesis and Deployment of Cloud Applications** R. Di Cosmo, M. Lienhardt, R. Treinen, S. Zacchiroli, J. Zwolakowski, A. Eiche, A. Agah	2014, ACM	Initial Search (Google Scholar)	
[18]	**A Multi-model Optimization Framework for Cloud Applications** D. Ardagna, G. P. Gibilisco, M. Ciavotta, A. Lavrentev	2014, Springer	Initial Search (Google Scholar)	[25]
[25]	**Search-Based Genetic Optimization for Cloud Software Reconfiguration** S. Frey, F. Fittkau, W. Hasselbring	2013, IEEE	Backward Search	
[20]	**CloudOpt: Multi-Goal Optimization of Application Deployments** J. Z. Li, M. Woodside, J. Chinneck	2011, IEEE	Backward Search	
[13]	**PerOpteryx: Automated Application of Tactics in Multi-Objective Software Architecture Optimization** A. Koziolek, H. Koziolek, R. Reussner	2011, ACM	Backward Search	

[2] https://github.com/AntonFrisch/Methodology_Architecture_Optimization_Cloud_Applications

To answer **RQ1**, the approaches from the selected publications were firstly categorized according to the existing taxonomy by Aleti et al. [3]. This was done in an extensive descriptive manner without predefined values. In a second step, the categorization was reviewed again to refine the classification. Similar concepts were merged together into synonymous terms to create a finite list of categorization values. If necessary, new values were created and unused values were deleted. Several categories that captured the information in the primary studies on an abstract level were extended for them to capture more specific information. In the final step, all papers were re-evaluated with the set of finite categories and values created. Therefore employing a test-retest procedure advised by Kitchenham [8] for single researchers. By synthesizing relevant information for every category in a quantitative and descriptive manner from the resulting categorization, **RQ2** is answered.

4. Results

In the following, the results of applying the methodology described in Section 3 are presented with Section 4.1 focusing on **RQ1** and Section 4.2 focusing on **RQ2**.

4.1. A Taxonomy of Software Architecture Optimization Approaches

The overall structure and most categories of the taxonomy by Aleti et al. [3] are kept also for classifying optimization approaches for cloud applications. With the *Problem*, *Solution*, and *Validation* categories, the main aspects of *what* an approach aims to provide, *how* it does so and whether it can be proven to be *valid*, can be described. For the subcategories, however, some modifications were made, based on the reviewed literature. The result is the taxonomy shown in fig. 2. Firstly, the *Domain* category from the original taxonomy, is replaced with the *Architectural Type* category. While the original taxonomy was intended to be applicable within a broad range of domains, cloud applications are a specific domain. And a category that does not provide differentiation between approaches has no value in a taxonomy. Nevertheless, a differentiation can be done based on which aspects or layers of an architecture are considered. Secondly, subcategories in the *Solution* category have been refined by adding categories that allow for a classification on different layers of abstraction. For example, the *Architecture representation* category enables a classification of the general type of representation. And the *Architecture Representation Method* allows for a classification of which specific language, tool, or technology is used. Thirdly, the largest change is that the specific values assignable in each category are newly defined based on the reviewed approaches. That means values included by Aleti et al. [3] were excluded to enable a clean derivation of relevant values. The resulting values are considered more in detail in the next section.

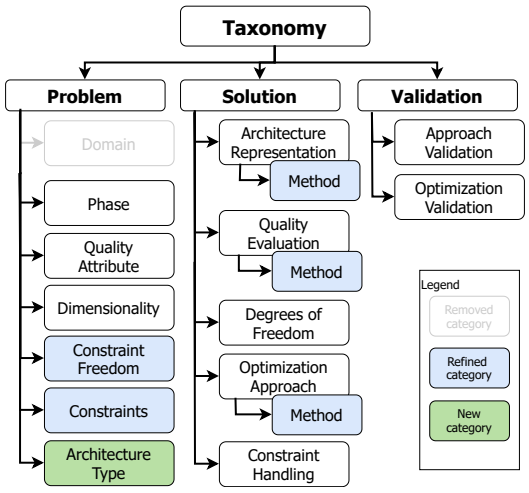

Figure 2: Taxonomy Overview (based on [3])

4.2. Classification of Current Approaches

The result of reviewing the found approaches and their classification based on the taxonomy is shown in table 2. It has to be noted that for the categories *Quality Attribute*, *Constraints*, *Quality Evaluation*, *Degrees of Freedom*, *Optimization Approach*, *Optimization Method*, *Constraint Handling*, and *Optimization Validation* a presented approach could be assigned to multiple values.

Problem Category For the *Phase* in which approaches can be applied, design time approaches are dominant and no approach was identified focusing exclusively on runtime optimization. Thus, these

approaches rely on complete input availability before deployment — either through system modeling or, as in 50% of the approaches, by incorporating runtime data. The most commonly optimized *Quality Attribute* is cost, which is the sole optimization goal in 8 approaches and considered alongside other goals, often performance, in 9 approaches. Other quality attributes appear less frequently. 58% of the reviewed papers use fixed *Constraints*, meaning these approaches adhere to a predetermined unmodifiable set of constraints. In contrast, 37% of the papers allow the user to set the constraints based on the system's needs. Performance is the most frequently used constraint (74%). In most approaches, multiple constraints are applied and combined in various ways. The *Architecture type* category reveals a clear dominance of deployment architecture approaches, accounting for 95% of the approaches.

Solution Category In the *Architecture representation* category, Performance Models are used the most (84%). A performance model focuses on representing a system's components and their connections combined with performance metrics. In contrast, Architecture models capture the structural organization of components and are used only in two approaches. The specific methods that are used cover a broad range. Extended forms of the PCM (Palladio Component model [29]) are the most prevalent (37%). All other modeling approaches are only used once. The *Quality Evaluation* category classifies how the quality attributes are quantified and evaluated. Model-Based (MB) techniques are predominant (47%), followed by Simulation-Based (SB) approaches (26%). Layered Queuing Networks (LQN) and M/G/1 queuing models emerged as core approaches for predicting how quality attributes such as response time, throughput, and system costs behave under load. LQNs are widely adopted due to their capability to model complex, multi-layered application architectures, with each layer representing different components or services [30]. Approaches are categorized as "Inherent" if they achieve optimization through inherent solution satisfaction. These approaches inherently satisfy quality attributes and constraints and therefore don't rely on distinct quality evaluation methods. Regarding *Degrees of Freedom*, Component Allocation stands out as the most frequently applied, (79%). Horizontal Scalability, Component Replication, and Resource selection follow closely, indicating a strong emphasis on scaling strategies, that are essential for enhancing flexibility and performance in cloud environments. The combination of Horizontal Scalability, Component Replication, and Component Allocation is notably prevalent, appearing together in 11 approaches [27, 18, 12, 21, 24, 16, 9, 17, 22, 15, 26]. For the *Optimization strategy* approximate methods dominate in the findings (79%). Exact methods are less common (42%). Within these, Mixed-Integer Linear Programming and Tabu Search are the most prevalent, highlighting their effectiveness in solving specific types of architectural optimization problems. Meanwhile, Genetic Algorithms and Evolutionary Algorithms make up a significant portion of the approximate methods. In the reviewed approaches, Tabu Search is frequently used as a second-phase refinement method following an initial solution generated by Mixed Integer Linear Programming. Genetic Algorithms can be considered as a subfield of Evolutionary Algorithms but since their frequent appearance during the review, they were categorized separately. For *Constraint Handling*, the prohibit method is the most common. It ensures that only feasible configurations are generated by strictly prohibiting any solutions that violate predefined constraints. Repair strategies are the second most common. They allow minor adjustments to solutions to meet constraints.

Validation Category In the *Approach Validation* category case studies are often used to demonstrate practical applicability in realistic settings. However, only a few case studies are conducted within a productive industrial setting [24, 19] and most are using available open-source applications as examples [25, 17] or prototypes [27, 10, 9]. Experiments are also widely used, allowing for controlled, theoretical validation in simulated environments [18, 15, 20]. Finally, Benchmark Problems, specifically the SPECjEnterpriseNEXT benchmark, used by Willnecker et al. [23, 16], are less common. For *Optimization Validation*, a significant portion (53%) of studies did not present any explicit comparative validation. Comparison with a baseline is otherwise the predominant method, providing a benchmark against other simpler algorithms or approaches.

Table 2
Resulting classification according to the taxonomy

1.1 Problem Category

Phase	
Design Time (84%)	[18, 12, 21, 24, 28, 16, 10, 15, 9, 17, 22, 20, 23, 14, 13, 26]
Hybrid (16%)	[27, 19, 25]
Quality Attributes	
Cost (89%)	[27, 18, 12, 21, 24, 11, 16, 19, 15, 9, 17, 22, 20, 14, 13, 25, 26]
Performance (47%)	[24, 11, 16, 9, 17, 23, 13, 25, 26]
Resource Utilization (11%)	[16, 23]
Software Quality (5%)	[10]
Power Consumption (11%)	[9, 20]
SLA Violations (5%)	[25]
Reliability (5%)	[26]
Dimensionality	
Single-Objective Optimization (58%)	[27, 18, 12, 21, 11, 19, 10, 15, 22, 23, 14]
Multi-Objective Optimization (37%)	[24, 16, 9, 17, 13, 25, 26]
Hybrid (5%)	[20]
Constraint Freedom	
Fixed (58%)	[18, 24, 11, 9, 17, 20, 23, 14, 13, 25, 26]
Customizable (37%)	[27, 12, 21, 16, 19, 15, 22]
None (5%)	[10]
Constraints	
Performance (74%)	[27, 18, 12, 24, 11, 16, 15, 9, 17, 20, 23, 14, 13, 25, 26]
Resource Limits (47%)	[27, 18, 12, 21, 16, 19, 15, 22, 20]
Deployment Constraints (32%)	[21, 16, 19, 22, 23, 25]
Cost (42%)	[24, 11, 16, 9, 17, 13, 25, 26]
Power Consumption (5%)	[9]
License Availability (5%)	[20]
SLA Violations (5%)	[25]
Reliability (5%)	[26]
None (5%)	[10]
Architecture Type	
Deployment (95%)	[27, 18, 12, 21, 24, 11, 16, 19, 15, 9, 17, 22, 20, 23, 14, 13, 25, 26]
Software Architecture (5%)	[10]

1.3 Validation Category

Approach Validation	
Case study (58%)	[27, 12, 21, 24, 19, 10, 9, 17, 14, 13, 25]
Experiment (32%)	[18, 11, 15, 22, 20, 26]
Benchmark problems (11%)	[16, 23]
Optimization Validation	
Not presented (53%)	[27, 21, 11, 16, 19, 10, 9, 23, 14, 13]
Comparison with baseline heuristic algorithm (42%)	[18, 12, 24, 15, 17, 20, 25, 26]
Comparison with random search (5%)	[24]
Internal comparison (5%)	[22]

1.2 Solution Category

Architecture Representation			
Performance Model (48%)	[27, 18, 12, 21, 11, 16, 19, 15, 9, 17, 22, 23, 14, 13, 25, 26]		
Architecture Model (11%)	[24, 10]		
Evaluation Model (5%)	[20]		
Architecture Representation Method			
PCM [extended] (42%)	[27, 18, 12, 16, 15, 23, 14, 13]		
ACM [extended](11%)	[21, 22]	ATG (5%)	[24]
Directed Graph (5%)	[11]	ABS extended (5%)	[19]
TOSCA (5%)	[10]	UML extended (5%)	[9]
CAPOM (5%)	[17]	LQM (5%)	[20]
KDM (5%)	[25]	None (5%)	[26]
Quality Evaluation			
Model-Based (47%)	[27, 18, 12, 11, 15, 9, 20, 14, 13]		
Simulation-Based (26%)	[24, 16, 19, 23, 25]		
Nonlinear Math. Function (21%)	[27, 18, 12, 15]		
Inherent (26%)	[21, 10, 17, 22, 26]		
Quality Evaluation Method			
Layered Queueing Network (42%)	[27, 18, 12, 15, 9, 20, 14, 13]		
M/G/1 (21%)	[27, 18, 12, 15]		
Palladio-bench (11%)	[16, 23]		
MC-OQN (5%)	[24]	Analytical Model (5%)	[11]
ABS Simulator (5%)	[19]	CDOSim (5%)	[25]
Inherent (26%)	[21, 10, 17, 22, 26]		
Degrees of Freedom			
Component Allocation (79%)	[27, 18, 12, 21, 21, 24, 16, 19, 9, 17, 22, 20, 23, 14, 13, 25, 26]		
Horizontal Scalability (74%)	[27, 18, 12, 21, 24, 16, 19, 15, 9, 17, 22, 23, 14, 13, 25, 26]		
Component Replication (74%)	[27, 18, 12, 21, 21, 24, 16, 19, 15, 9, 17, 22, 20, 23, 13, 26]		
Resource Selection (58%)	[27, 18, 12, 21, 24, 19, 15, 17, 22, 14, 25, 26]		
Vertical Scalability (26%)	[11, 19, 23, 13, 25]		
Component Selection (16%)	[19, 14, 13]		
Provider Service Selection (11%)	[15, 25]		
Software Pattern (5%)	[10]		
Workflow Orchestration (5%)	[11]		
Optimization Strategy			
Approximate (79%)	[27, 18, 12, 24, 11, 16, 10, 15, 9, 17, 23, 14, 13, 25, 26]		
Exact (42%)	[27, 18, 12, 21, 19, 15, 22, 20]		
Optimization Method			
Mixed Linear Int. Prog. (26%)	[27, 18, 12, 15, 20]		
Tabu Search (26%)	[27, 18, 12, 15, 14]		
Genetic Algorithm (26%)	[9, 14, 13, 25]		
Constraint Programming (16%)	[21, 19, 22]		
Evolutionary Algorithm(16%)	[24, 16, 17, 23]		
Greedy Algorithm (5%)	[11]		
Refactoring Rules (5%)	[10]		
Ant Colony Optimization (5%)	[26]		
Constraint Handling			
Prohibit (79%)	[27, 18, 12, 21, 24, 11, 19, 15, 9, 22, 23, 14, 13, 25, 26]		
Repair (37%)	[27, 18, 12, 16, 15, 17, 14]		
None (5%)	[10]	Penalty (5%)	[20]

5. Discussion

Architecture Optimization Approaches for Cloud Applications, as considered here, remain a niche topic, reflected by the relatively few papers found in the literature search. Although this observation may in part be the result of the more strictly formulated inclusion criteria in Section 3 that consider only approaches presenting a structured and automated optimization method. More approaches are available which however include manual steps or do not provide as concrete optimization suggestions as the approaches considered in this work.

To answer **RQ1**, we can state that Aleti et al.'s taxonomy already enables an effective categorization of cloud application optimization approaches. We adapted it by removing one category, adding new categories with different abstraction levels, and defining a new set of values as described in 4.1.

For **RQ2** our findings in 4.2 represent the basis from which the following conclusions can be drawn:
Dominance of Design-Time Approaches Most reviewed approaches focus on optimization before deployment, allowing early performance and cost predictions without incurring additional test infrastructure expenses. This design-time emphasis supports more complex solution derivation, as runtime constraints (e.g., adaptation speed) are less critical. However, questions remain about the actual benefits of runtime optimization versus predefined scaling policies.
Prevalence of Cost and Performance as Key Quality Attributes: Cost and performance dominate in the reviewed approaches. Cost optimization, in particular, appears in nearly all single-objective approaches and all multi-objective ones, reflecting the economic focus of using cloud services. Furthermore, performance and cost are more straightforward to observe at runtime in order to validate optimization approaches. Optimization approaches for other quality aspects, like maintainability or portability require more effort in their validation since case studies or experiments need to be done in a longer time span. While this focus on performance and cost is practical, it highlights a research gap concerning other quality attributes, for example also reliability.
Focus on Performance Models and Palladio Component Model Many approaches rely on performance models, with PCM being the most widely used thanks to its maturity and tool support. This strong focus indicates PCM's effectiveness in predicting and evaluating QoS attributes. Although alternatives such as Aeolus or customized models exist and are employed the PCM is the only model with widespread adoption in the reviewed approaches.
Dominance of Approximate Optimization Strategies 79% of the approaches apply heuristic or approximate approaches to tackle the NP-hard [31] nature of cloud architecture optimization. Exact methods appear less frequently and often in combination with approximations, usually by limiting problem scope or mixing modeling techniques to ensure feasibility.
Validation Gap in Cloud Architecture Optimization. Few standardized benchmarks exist, making comparisons across approaches difficult. Most validations rely on case studies rather than real-world industrial settings and lack detailed demonstrations of economic benefits. Future research could focus on robust benchmarking frameworks and well-documented success stories to strengthen the field's credibility and practical impact.

Some limitations of this review must be acknowledged. Only complete optimization approaches presented in academia were included which are typically toolchains comprised of different subfields like software modeling and the formulation of optimization problems. Therefore advancements in individual subfields and industry-driven solutions are not part of this study. Additionally, threats to completeness arise from the selection of databases and search terms. Also, the review was mainly conducted by a single researcher. Therefore the risk of some bias can't be completely excluded but was mitigated through a clear methodology and taxonomy-based extraction.

As stated before, we are not aware of another review for software architecture optimization approaches specifically targeting cloud applications. Therefore, as related work mainly the reviewed primary studies would have to be considered and the review by Aleti et al. [3] upon which this study is built.

6. Conclusion

This work presents a systematic review of 19 papers on architecture optimization approaches for cloud applications. A taxonomy for categorizing approaches was derived and current research trends in the field were identified. Key findings include the dominance of design-time optimization, cost-focused goals, reliance on performance models like PCM, and the prevalence of approximate optimization methods. Gaps in validation strategies and a lack of standardized benchmarks were identified, highlighting areas for improvement. These insights aim to guide future research and development for new approaches, advancing the field with more robust and versatile solutions.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Taherkordi, F. Zahid, Y. Verginadis, G. Horn, Future cloud systems design: Challenges and research directions, IEEE Access 6 (2018) 74120–74150. doi:10.1109/access.2018.2883149.

[2] N. Kratzke, P.-C. Quint, Understanding Cloud-native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study, Journal of Systems and Software 126 (2017) 1–16. doi:10.1016/j.jss.2017.01.001.

[3] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, I. Meedeniya, Software architecture optimization methods: A systematic literature review, IEEE TSE 39 (2013) 658–683. doi:10.1109/tse.2012.64.

[4] M. Shaw, P. Clements, The golden age of software architecture, IEEE Software 23 (2006) 31–39. doi:10.1109/ms.2006.58.

[5] ISO/IEC/IEEE, ISO/IEC/IEEE 42010:2022 - Software, systems and enterprise — Architecture description, 2022. URL: https://www.iso.org/standard/74393.html.

[6] C. Pahl, P. Jamshidi, O. Zimmermann, Architectural Principles for Cloud Software, ACM Transactions on Internet Technology 18 (2018) 1–23. doi:10.1145/3104028.

[7] N. Kratzke, R. Peinl, ClouNS - a Cloud-Native Application Reference Model for Enterprise Architects, in: IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW), IEEE, 2016, pp. 1–10. doi:10.1109/edocw.2016.7584353.

[8] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, techreport EBSE-2007-01, Keele University and Durham University, 2007. URL: https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf.

[9] V. Cortellessa, D. Di Pompeo, M. Tucci, Exploring Sustainable Alternatives for the Deployment of Microservices Architectures in the Cloud, in: 2024 IEEE 21st International Conference on Software Architecture (ICSA), 2024, pp. 34–45. doi:10.1109/ICSA59870.2024.00012.

[10] J. Soldani, G. Muntoni, D. Neri, A. Brogi, The μTOSCA toolchain: Mining, analyzing, and refactoring microservice-based architectures, Software: Practice and Experience 51 (2021) 1591–1621. doi:10.1002/spe.2974.

[11] C. Lin, H. Khazaei, Modeling and Optimization of Performance and Cost of Serverless Applications, IEEE TPDS 32 (2021) 615–632. doi:10.1109/TPDS.2020.3028841.

[12] M. Ciavotta, G. P. Gibilisco, D. Ardagna, E. D. Nitto, M. Lattuada, M. A. A. Da Silva, Architectural Design of Cloud Applications: A Performance-Aware Cost Minimization Approach, IEEE Transactions on Cloud Computing 10 (2022) 1571–1591. doi:10.1109/TCC.2020.3015703.

[13] A. Koziolek, H. Koziolek, R. Reussner, PerOpteryx: Automated application of tactics in multi-objective software architecture optimization, in: Joint ACM SIGSOFT Conference – QoSA and ACM

SIGSOFT Symposium – ISARCS on Quality of Software Architectures, ACM, Boulder Colorado USA, 2011, pp. 33–42. doi:10.1145/2000259.2000267.

[14] M. Ciavotta, M. Ardagna, A. Koziolek, Palladio Optimization Suite: QoS optimization for component-based Cloud applications, in: 9th EAI International Conference on Performance Evaluation Methodologies and Tools, ACM, Berlin, Germany, 2016. doi:10.4108/eai.14-12-2015.2262562.

[15] M. Ciavotta, D. Ardagna, G. P. Gibilisco, A mixed integer linear programming optimization approach for multi-cloud capacity allocation, Journal of Systems and Software 123 (2017) 64–78. doi:10.1016/j.jss.2016.10.001.

[16] F. Willnecker, H. Krcmar, Multi-Objective Optimization of Deployment Topologies for Distributed Applications, ACM Transactions on Internet Technology 18 (2018) 1–21. doi:10.1145/3106158.

[17] X. Du, Y. Ni, P. Ye, X. Wang, R. Xiao, Performance Optimization of Cloud Application at Software Architecture Level, in: K. Li, W. Li, H. Wang, Y. Liu (Eds.), Artificial Intelligence Algorithms and Applications, volume 1205, Springer Singapore, 2020, pp. 724–738. doi:10.1007/978-981-15-5577-0_58.

[18] D. Ardagna, G. P. Gibilisco, M. Ciavotta, A. Lavrentev, A Multi-model Optimization Framework for the Model Driven Design of Cloud Applications, in: C. Le Goues, S. Yoo (Eds.), Search-Based Software Engineering, volume 8636, Springer International Publishing, Cham, 2014, pp. 61–76. doi:10.1007/978-3-319-09940-8_5.

[19] S. De Gouw, J. Mauro, G. Zavattaro, On the modeling of optimal and automatized cloud application deployment, Journal of Logical and Algebraic Methods in Programming 107 (2019) 108–135. doi:10.1016/j.jlamp.2019.06.001.

[20] J. Z. Li, M. Woodside, J. Chinneck, M. Litoiu, CloudOpt: Multi-goal optimization of application deployments across a cloud, in: 7th International Conference on Network and Service Management, 2011, pp. 1–9. URL: https://ieeexplore.ieee.org/abstract/document/6103947.

[21] R. Di Cosmo, M. Lienhardt, R. Treinen, S. Zacchiroli, J. Zwolakowski, A. Eiche, A. Agahi, Automated synthesis and deployment of cloud applications, in: 29th ACM/IEEE International Conference on Automated Software Engineering, ACM, Vasteras Sweden, 2014, pp. 211–222. doi:10.1145/2642937.2642980.

[22] E. Ábrahám, F. Corzilius, E. B. Johnsen, G. Kremer, J. Mauro, Zephyrus2: On the Fly Deployment Optimization Using SMT and CP Technologies, in: M. Fränzle, D. Kapur, N. Zhan (Eds.), Dependable Software Engineering: Theories, Tools, and Applications, Springer International Publishing, Cham, 2016, pp. 229–245. doi:10.1007/978-3-319-47677-3_15.

[23] F. Willnecker, H. Krcmar, Optimization of Deployment Topologies for Distributed Enterprise Applications, in: 12th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA), IEEE, Venice, 2016, pp. 106–115. doi:10.1109/QoSA.2016.11.

[24] N. Tanković, T. Galinac Grbac, M. Žagar, ElaClo: A framework for optimizing software application topology in the cloud environment, Expert Systems with Applications 90 (2017) 62–86. doi:10.1016/j.eswa.2017.07.001.

[25] S. Frey, F. Fittkau, W. Hasselbring, Search-based genetic optimization for deployment and reconfiguration of software in the cloud, in: 35th International Conference on Software Engineering (ICSE), IEEE, San Francisco, CA, USA, 2013, pp. 512–521. doi:10.1109/ICSE.2013.6606597.

[26] A. Ashraf, B. Byholm, I. Porres, A Multi-objective ACS Algorithm to Optimize Cost, Performance, and Reliability in the Cloud, in: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), IEEE, Limassol, Cyprus, 2015, pp. 341–347. doi:10.1109/UCC.2015.54.

[27] M. Guerriero, M. Ciavotta, G. P. Gibilisco, D. Ardagna, A Model-Driven DevOps Framework for QoS-Aware Cloud Applications, in: 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), IEEE, Timisoara, Romania, 2015, pp. 345–351. doi:10.1109/SYNASC.2015.60.

[28] M. Loukides, The Cloud in 2021: Adoption Continues, O'Reilly Media, Inc., 2021. URL: https://www.oreilly.com/radar/the-cloud-in-2021-adoption-continues/.

[29] S. Becker, H. Koziolek, R. Reussner, The Palladio component model for model-driven performance

prediction, Journal of Systems and Software 82 (2009) 3–22. doi:10.1016/j.jss.2008.03.066.

[30] G. Franks, T. Al-Omari, M. Woodside, O. Das, S. Derisavi, Enhanced Modeling and Solution of Layered Queueing Networks, IEEE TSE 35 (2009) 148–161. doi:10.1109/TSE.2008.74.

[31] G. Canfora, M. Di Penta, R. Esposito, M. L. Villani, An approach for qos-aware service composition based on genetic algorithms, in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO '05, Association for Computing Machinery, New York, NY, USA, 2005, p. 1069–1075. doi:10.1145/1068009.1068189.