SPACE4AI-R: Runtime Management Tool for AI Applications Component Placement and Resource Selection in Computing Continua

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Randeep Singh, 967445

Advisor:
Prof. Danilo Ardagna

Co-advisors:
Federica Filippini
Hamta Sedghani

Academic year:
2021-202

Abstract: The importance and pervasiveness of Artificial Intelligence (AI) are dramatically increasing in these years, together with an accelerated migration towards Internet of Things, determining the rise of the Edge computing paradigm. According to this trend, Edge intelligence is expected to become the foundation of many AI applications use cases, spanning from predictive maintenance to machine vision and healthcare. Edge computing generates a fragmented scenario, where computing and storage power are distributed among multiple devices with highly heterogeneous capacities. In this framework, component placement and resource selection become crucial to arrange in the most convenient way the available resources of the Computing Continuum. In this work, we propose SPACE4AI-R, a tool to effectively address the runtime management of AI applications component placement and resource selection in the Computing Continuum. Through a Random Search combined with a Stochastic Local Search algorithm, SPACE4AI-R copes with runtime workload fluctuations by identifying the cost-optimal reconfiguration of the initial production deployment, while providing performance guarantees across heterogeneous resources including Edge devices and servers, Cloud GPU-based Virtual Machines and Function as a Service solutions. Experimental results show that our tool efficiently finds placement reconfigurations in a real use case of identifying wind turbines blade damage, and can manage large-scale systems providing remarkable cost savings over static placements, while keeping execution time in the order of seconds.

Key-words: Component placement, Edge computing, Local Search, Optimization, Resource selection

1. Introduction

Nowadays, Artificial Intelligence (AI) is becoming increasingly popular in a wide range of sectors and industries. According to Fortune Business Insights (2022), the global AI market size is projected to reach USD 1395 billion in 2029, at a CAGR of 20.1% [1]. Cloud computing has represented one of the main computing paradigm for AI and big data applications, but the recent technological advances in processors, memory and communications initiated the Internet of Things (IoT) era, where a huge amount of data is generated by widespread end devices (smart watches, smart city power grids, connected vehicles, smart homes are some examples) [2]. Since IoT requires mobility support and geo-distribution in addition to location awareness, Cloud computing paradigm

alone is not suited for these latency-sensitive applications [3]. Also, if so many end devices send their data to the Cloud, there may be a network overload, making Cloud prohibitive in this framework.
Edge computing paradigm has emerged as a solution to the Cloud computing limitations. It brings storage, computation and network services near the end devices through the so called fog nodes, which can be switches, cameras, routers, computers and servers, linked through a stable network connection [4]. Thanks to this approach, Edge computing is characterized by low latency, wide geographic distribution, processing power closer to the user and real time interactions, overcoming Cloud computing lacks [5]. Nonetheless, it is important to remark that Edge computing is not a replacement to Cloud computing; if, on the one side, the main advantage of Edge systems is to improve applications performance by reducing the latency, on the other side, Edge resources have usually less computing capacity than the Cloud and can become a bottleneck in the computation. Hence, if an application requires computational power which is not available at the Edge nodes, there should still be the option to connect to the Cloud data centers [4], if delay requirements and network connections allow so. Therefore, the *Computing Continuum* paradigm is the best approach: latency-sensitive tasks can be distributed among Edge nodes while compute-intensive tasks are offloaded to the Cloud layers. At this point it is clear that Edge computing generates a fragmented scenario, where the computational power is distributed among devices with highly heterogeneous capacities and connectivity. In this framework, component placement and resource selection are crucial to orchestrate at best the Computing Continuum resources, minimizing the expected costs while meeting Quality of Service (QoS) requirements [2]. In particular, given an infrastructure $\mathcal{S}$ and an application $\mathcal{A}$, where $\mathcal{S}$ contains information on the resources (Cloud Virtual Machines, Edge nodes, end devices) with their capacity (in terms of CPU, memory), and $\mathcal{A}$ contains information about the components (how they invoke each other, how they exchange data), the goal of the placement problem it to find a mapping of applications components to the infrastructure resources, while fulfilling hardware, network, privacy and Quality of Service requirements [6] (see Figure 1).

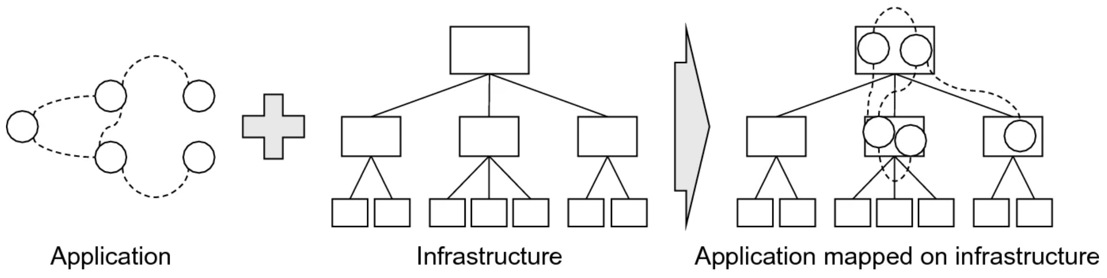

Figure 1: The placement problem: mapping application on the infrastructure [6]

The component placement problem needs to be tackled in two different phases: design-time and runtime [7]. Design-time choices aim to determine the optimal allocation of the application components on the candidate resources, satisfying all the requirements and the devices computational or storage constraints. However, in practical applications, the application workload (requests per second to process) expected at design-time is often subject to fluctuations due, for example, to variations in the generated data volumes [7]. For this reason, the initially designed placement may become unfeasible or oversized depending on the input workload, so it has to be continuously monitored and adapted online.
SPACE4AI-D, developed in [8], is a tool that tackles the component placement problem and resource selection in the Computing Continuum at design-time, dealing with different application requirements. In this thesis, we propose SPACE4AI-R, which, exploiting an efficient Random Search combined with a Stochastic Local Search algorithm, tackles the runtime problem by identifying a suitable reconfiguration of the running placement able to cope with the workload fluctuations. The Computing Continuum consists of heterogeneous resources, including Edge devices and servers, Cloud GPU-based Virtual Machines and Function as a Service solutions
Design-time and runtime approaches have subtle but important differences. The most relevant one lies in their required responsiveness: while a design-time tool is allowed to take as much time as needed (up to several minutes) to find the initial production deployment (i.e., the initial placement), a runtime tool must provide a feasible reconfiguration in few seconds at most, due to the online running application. Therefore, runtime tools must be designed making appropriate cost-reactivity trade-offs while tuning the solver algorithms, as well as implemented in a very efficient way to reduce the execution time.
We validated SPACE4AI-R through an experimental analysis divided in two parts. The first one deals with a real use case application of 2 hours duration, consisting in wind turbines blade damage identification, in three different computing scenarios. We simulate the dynamic workload fluctuations by considering a bi-modal workload profile for the duration of the application, and request a reconfiguration every 5 minutes. Results show that our tool is able to efficiently compute the placement reconfiguration in all scenarios, keeping execution time below half a second. The second part deals with the scalability analysis, where we prove that our tool is able to tackle small (5 components) to large-scale (15 components) systems, keeping the execution time around a

couple of seconds, successfully adapting the design-time tool proposed in [8] to the runtime framework. Finally, we report the cost savings that can be obtained through the dynamic placement and reconfiguration mechanism over a static placement (tuned on the maximum expected workload) that is kept fixed for the entire duration of the application.
The remainder of this thesis is organized as follow. Related works are discussed in Section 2. Section 3 introduces the basic application and resources models. Section 4 describes a real use case that can benefit from our tool. Section 5 formulates the optimization problem, while Section 6 describes the heuristic solution. Experimental results are discussed in Section 7, and final conclusions are drawn in Section 8.

2. Related work

Component placement problems are continuously gaining a lot of attention from the research community. In [9], authors provided a classification of the literature proposals in terms of the placement purpose (e.g., scheduling, offloading, distribution of physical resources), the computing paradigm (Cloud-Edge, only Cloud, only Edge), and the optimization metrics (latency time, energy consumption, total cost of the placement). In [6], the authors proposed a more specific review on the algorithms used to solve the placement problem, according to different characteristics of the infrastructure $\mathcal{S}$ and the application $\mathcal{A}$. They also highlighted typical bad practices in the field (e.g., evaluation environments not matching real-case settings, insufficient number of simulations biasing the validation of results, missing state-of-the art algorithms as comparison baseline), and provided useful recommendations to foster meaningful evaluation settings that could help both the academic and the industrial world.
In [10], the authors proposed a tool to optimize resource allocation in the context of Deep Learning (DL) tasks (e.g., face recognition, intelligent surveillance system, and so on). Each type of task is characterized by a unique Deep Neural Network (DNN) model, partitionable in independent logical layers. Assuming that the tasks come with a given arrival rate, the tool finds the optimal partitioning of the DNNs models and place the layers at the Edge or at the Cloud, minimizing the long term expected End-to-End delay, under strict energy constraints. They used a reinforcement learning approach to find the best partitioning, while designed a heuristic to compute the optimal resource allocation for each DNN task. Results show that their approach outperforms the current available alternatives, reducing the End-to-End delay from 0.5% to 11% compared to the other tools, especially in tight energy budget scenarios. In [11], the authors coped with the task offloading in a multi-service scenario with Edge-Cloud cooperation. They designed a flexible tool that can address concurrently the service placement at the compatible devices, the computing resource allocation for each service, and the communication rate allocation between the different devices in the Computing Continuum. The optimization aims to minimize the total processing time, while guaranteeing the long term stability of the task queuing on all the Edge-Cloud devices and servers, which is crucial to control the list of pending and backlogged tasks. The problem is transformed into a deterministic Lyapunov-based formulation for each time instant, and a multi-timescale algorithm manages the offloading at subsequent time slots.
Among the discussed proposals, [12, 13] are the closest to our approach. In [12], the authors modeled the computing infrastructure as an Undirected Graph, where each node corresponds to the Edge and Cloud clusters, characterized by specific hardware characteristics (CPU, memory) with the geographic location, and each edge represents the network connection, characterized by bandwidth and latency. The Edge application, instead, is modeled as a Directed Acyclic Graph (DAG), where each node is an application component characterized by the requested hardware resources with the geographic location, and each edge is marked with a requested bandwidth and latency. The provisioning problem is formulated as a single-objective program, aiming to minimize the total provision cost in terms of processing, memory and data transfer costs. The problem is solved through a greedy approach, namely the *Edge-Native Provisioning* (ENP), which first splits the application into subsets of star components sorted according to the number of links, and then greedily selects the best provisioning for each star component. In [13], the authors addressed again the multi-component application placement problem, minimizing the total placement cost. They propose two heuristic algorithms, namely *MATH-MCAPP* and *G-MCAPP*. The first, based on matching and local search techniques, is very efficient when the number of components and devices is relatively limited, while G-MCAPP is a greedy algorithm designed for Mobile Edge Computing (MEC) with a large number of servers and components. Results show that the two algorithms outperforms the CPLEX solver in terms of execution time for small system instances, and they behave well with large scale problems (100 components and 200 available resources).
The proposals discussed so far modeled the placement and resource selection as a single-objective optimization problem (as we do in this work), focusing on delay optmization [10, 11], energy optimization [14, 15], and cost optimization [12, 13]. However, several works frame the placement problem in the multi-objective optimization branch. For instance, in [16] an approach for scheduling and offloading workflow tasks, represented as Directed Acyclic Graph, has been proposed modelling the Fog-Cloud computing infrastructure as a Multi-Agent System

8. Conclusions

The online placement orchestration requires fast and reactive solutions able to reconfigure the system in real time, i.e., while the AI applications are running, in order to cope with workload fluctuations. This work proposes SPACE4AI-R, a tool to support application components placement and resource selection in the Computing Continuum at runtime. The original design-time approach [8], not suited for the runtime framework due to inadequate solving time, has been first adapted to the runtime problem by implementing the Random Search exploiting the features of the faster `C++`, and then enhanced by adding an efficient Stochastic Local Search downstream, designed to further reduce the placement cost. Experimental results show that our tool effectively manages a real use case exploiting three different computing infrastructures, and can cope with large-scale systems while providing important cost savings through the dynamic placement. SPACE4AI-R execution time is of the order of seconds, being at least two orders of magnitude faster than the design-time approach, and yet the costs of the computed solutions never exceed the ones of the design-time placements.
Future works will extend the Stochastic Local Search algorithm by investigating other neighborhood exploration techniques possibly better than the first improving, as well as design new approaches based on different heuristics, such as Tabu Search. Moreover, new performance models based on machine learning models, trained with real use cases data, will be introduced to improve the resources response time estimation accuracy.

References

[1] Fortune Business Insights Pvt. Ltd. Artificial Intelligence (AI) Market Size Forecast, 2022. URL `https://www.fortunebusinessinsights.com/industry-reports/artificial-intelligence-market-100114`.

[2] Breno Costa, Joao Bachiega, Leonardo Rebouças de Carvalho, and Aleteia P. F. Araujo. Orchestration in Fog Computing: A Comprehensive Survey. *ACM Comput. Surv.*, 55(2), 2022. ISSN 0360-0300. doi: 10.1145/3486221. URL `https://doi.org/10.1145/3486221`.

[3] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450315197. doi: 10.1145/2342509.2342513. URL `https://doi.org/10.1145/2342509.2342513`.

[4] Sabireen H. and Neelanarayanan V. A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges. *ICT Express*, 7(2):162–176, 2021. ISSN 2405-9595. doi: https://doi.org/10.1016/j.icte.2021.05.004. URL `https://www.sciencedirect.com/science/article/pii/S2405959521000606`.

[5] Michaela Iorga, Larry Feldman, Robert Barton, Michael Martin, Nedim Goren, and Charif Mahmoudi. Fog Computing Conceptual Model, 2018-03-14 2018.

[6] Sven Smolka and Zolt Mann. Evaluation of Fog Application Placement Algorithms: A Survey. *Computing*, 104(6):1397–1423, jun 2022. ISSN 0010-485X. doi: 10.1007/s00607-021-01031-8. URL `https://doi.org/10.1007/s00607-021-01031-8`.

[7] Hamta Sedghani, Federica Filippini, and Danilo Ardagna. A Randomized Greedy Method for AI Applications Component Placement and Resource Selection in Computing Continua. In *2021 IEEE International Conference on Joint Cloud Computing (JCC)*, pages 65–70, 2021. doi: 10.1109/JCC53141.2021.00022.

[8] Hamta Sedghani, Federica Filippini, and Danilo Ardagna. A Random Greedy based Design Time Tool for AI Applications Component Placement and Resource Selection in Computing Continua. In *2021 IEEE International Conference on Edge Computing (EDGE)*, pages 32–40, 2021. doi: 10.1109/EDGE53862.2021.00014.

[9] Julian Bellendorf and Zoltán Ádám Mann. Classification of optimization problems in fog computing. *Future Generation Computer Systems*, 107:158–176, 2020. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2020.01.036. URL `https://www.sciencedirect.com/science/article/pii/S0167739X19323568`.

[10] Yi Su, Wenhao Fan, Li Gao, Lei Qiao, Yuan'an Liu, and Fan Wu. Joint DNN Partition and Resource Allocation Optimization for Energy-Constrained Hierarchical Edge-Cloud Systems. *IEEE Transactions on Vehicular Technology*, pages 1–15, 2022. doi: 10.1109/TVT.2022.3219058.

[11] Wenhao Fan, Liang Zhao, Xun Liu, Yi Su, Shenmeng Li, Fan Wu, and Yuan'an Liu. Collaborative Service Placement, Task Scheduling, and Resource Allocation for Task Offloading with Edge-Cloud Cooperation. *IEEE Transactions on Mobile Computing*, pages 1–18, 2022. doi: 10.1109/TMC.2022.3219261.

[12] Baudouin Herlicq, Abderaouf Khichane, and Ilhem Fajjari. NextGenEMO: an Efficient Provisioning of Edge-Native Applications. In *ICC 2022 - IEEE International Conference on Communications*, pages 1924–1929, 2022. doi: 10.1109/ICC45855.2022.9839012.

[13] Tayebeh Bahreini and Daniel Grosu. Efficient Algorithms for Multi-Component Application Placement in Mobile Edge Computing. *IEEE Transactions on Cloud Computing*, pages 1–1, 2020. doi: 10.1109/TCC.2020.3038626.

[14] Jing Bi, Kaiyi Zhang, Haitao Yuan, and Jia Zhang. Energy-Efficient Computation Offloading for Static and Dynamic Applications in Hybrid Mobile Edge Cloud System. *IEEE Transactions on Sustainable Computing*, pages 1–13, 2022. doi: 10.1109/TSUSC.2022.3216461.

[15] Ying Chen Shaoxuan Yun. Intelligent Traffic Scheduling for Mobile Edge Computing in IoT via Deep Learning. *Computer Modeling in Engineering & Sciences*, 134(3):1815–1835, 2023. ISSN 1526-1506. doi: 10.32604/cmes.2022.022797. URL `http://www.techscience.com/CMES/v134n3/49749`.

[16] Marwa Mokni, Sonia Yassa, Jalel Eddine Hajlaoui, Mohamed Nazih Omri, and Rachid Chelouah. Multi-objective fuzzy approach to scheduling and offloading workflow tasks in Fog-Cloud computing. *Simulation Modelling Practice and Theory*, page 102687, 2022. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2022.102687.

[17] Ahmad Almadhor, Abdullah Alharbi, Ahmad M. Alshamrani, Wael Alosaimi, and Hashem Alyami. A new offloading method in the green mobile cloud computing based on a hybrid meta-heuristic algorithm. *Sustainable Computing: Informatics and Systems*, 36:100812, 2022. ISSN 2210-5379. doi: https://doi.org/10.1016/j.suscom.2022.100812. URL `https://www.sciencedirect.com/science/article/pii/S2210537922001433`.

[18] Yeting Guo, Fang Liu, Nong Xiao, Zhaogeng Li, Zhiping Cai, Guoming Tang, and Ning Liu. PARA: Performability-aware resource allocation on the edges for cloud-native services. *International Journal of Intelligent Systems*, 37(11):8523–8547, 2022. doi: https://doi.org/10.1002/int.22954. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22954`.

[19] Xun Shao, Go Hasegawa, Mianxiong Dong, Zhi Liu, Hiroshi Masui, and Yusheng Ji. An Online Orchestration Mechanism for General-Purpose Edge Computing. *IEEE Transactions on Services Computing*, pages 1–1, 2022. doi: 10.1109/TSC.2022.3164149.

[20] Danilo Ardagna and Barbara Pernici. Adaptive Service Composition in Flexible Processes. *IEEE Transactions on Software Engineering*, 33(6):369–384, 2007. doi: 10.1109/TSE.2007.1011.

[21] Maryam Ebrahimi, Alexandre da Silva Veith, Moshe Gabel, and Eyal de Lara. Combining DNN Partitioning and Early Exit. In *Proceedings of the 5th International Workshop on Edge Systems, Analytics and Networking*, EdgeSys '22, page 25–30, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392532. doi: 10.1145/3517206.3526270. URL `https://doi.org/10.1145/3517206.3526270`.

[22] Lizheng Jiang, Yunman Pei, and Jiantao Zhao. Overview Of Serverless Architecture Research. *Journal of Physics: Conference Series*, 1453(1):012119, jan 2020. doi: 10.1088/1742-6596/1453/1/012119. URL `https://dx.doi.org/10.1088/1742-6596/1453/1/012119`.

[23] Amazon. AWS Lambda: Run code without thinking about servers or clusters, 2022. URL `https://aws.amazon.com/lambda/`.

[24] Amazon. AWS Lambda Pricing, 2022. URL `https://aws.amazon.com/lambda/pricing/`.

[25] Microsoft. Azure Functions pricing, 2022. URL `https://azure.microsoft.com/en-us/pricing/details/functions/`.

[26] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative system performance: computer system analysis using queueing network models.* Prentice-Hall, Inc., 1984. ISBN 978-01374697580.

[27] Nima Mahmoudi and Hamzeh Khazaei. Performance Modeling of Serverless Computing Platforms. *IEEE Transactions on Cloud Computing*, pages 1–15, 2020. doi: 10.1109/TCC.2020.3033373.

[28] Uma Tadakamalla and Daniel A Menasce. Autonomic Resource Management for Fog Computing. *IEEE Transactions on Cloud Computing*, pages 1–1, 2021. doi: 10.1109/TCC.2021.3064629.

[29] Amazon. AWS Step Function pricing, 2022. URL `https://aws.amazon.com/step-functions/pricing/`.

[30] Microsoft. Azure Logic Apps, 2022. URL `https://azure.microsoft.com/en-us/products/logic-apps/`.

[31] Sebastián Risco, Germán Moltó, Diana M Naranjo, and Ignacio Blanquer. Serverless Workflows for Containerised Applications in the Cloud Continuum. *Journal of Grid Computing*, 19, 2021.

[32] Sean Luke. *Essentials of Metaheuristics* . Lulu, second edition, 2013. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[33] Randeep Singh. SPACE4AI-R GitHub repository, 2022. URL `https://github.com/randosrandom/Space4AI/tree/space4ai-v1.1`.

[34] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search Algorithms: An Overview*, pages 1085–1105. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. ISBN 978-3-662-43505-2. doi: 10.1007/978-3-662-43505-2_54. URL `https://doi.org/10.1007/978-3-662-43505-2_54`.

[35] Heber F. Amaral, Sebastián Urrutia, and Lars M. Hvattum. Delayed Improvement Local Search. *Journal of Heuristics*, 27(5):923–950, 2021. ISSN 1381-1231. doi: 10.1007/s10732-021-09479-9. URL `https://doi.org/10.1007/s10732-021-09479-9`.

[36] Eugeniusz Nowicki and Czeslaw Smutnicki. A Fast Taboo Search Algorithm for the Job Shop Problem. *Management Science*, 42(6):797–813, 1996. ISSN 00251909, 15265501. URL `http://www.jstor.org/stable/2634595`.

[37] Mauricio G. C. Resende and Celso C. Ribeiro. *Greedy Randomized Adaptive Search Procedures: Advances and Extensions*, pages 169–220. Springer International Publishing, Cham, 2019. ISBN 978-3-319-91086-4. doi: 10.1007/978-3-319-91086-4_6. URL `https://doi.org/10.1007/978-3-319-91086-4_6`.

[38] Pierre Hansen and Nenad Mladenović. First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, 154(5):802–817, 2006. ISSN 0166-218X. doi: https://doi.org/10.1016/j.dam.2005.05.020. URL `https://www.sciencedirect.com/science/article/pii/S0166218X05003070`.

[39] Danilo Ardagna Hamta Sedghani, Federica Filippini. SPACE4AI-D GitLab, 2022. URL `https://gitlab.polimi.it/ai-sprint/space4ai-d`.

[40] Danilo Ardagna, Michele Ciavotta, Riccardo Lancellotti, and Michele Guerriero. A hierarchical receding horizon algorithm for qos-driven control of multi-iaas applications. *IEEE Transactions on Cloud Computing*, 9(2):418–434, 2021. doi: 10.1109/TCC.2018.2875443.

[41] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.