Building the Digital Twin of a MEC node: a Data Driven Approach

Riccardo Fedrizzi*, Arturo Bellin[†‡], Cristina Emilia Costa[§], Fabrizio Granelli[†],
* Fondazione Bruno Kessler, Trento, Italy - E-mail: rfedrizzi@fbk.eu
[†] University of Trento, Trento, Italy - E-mail: {arturo.bellin,fabrizio.granelli}@unitn.it
[‡] Research and Innovation Department, Athonet S.r.l. - E-mail: arturo.bellin@intern.athonet.com
[§] CNIT, Italy - E-mail: cristina.costa@cnit.it

***Abstract*—Multi-access edge computing (MEC) represents an emerging solution to improve the performance of mobile networks by bringing computing resources closer to the edge of the network. However, MEC requires the implementation of virtualization and can be deployed using different hardware platforms, including COTS devices. In this highly heterogeneous scenario, the digital twin (DT), assisted by proper AI/ML solutions, is envisioned to play a crucial role in automated network management, operating as an intermediate and collaborative layer enabling the orchestration layer to better understand network behavior before making changes to the physical network. In this paper, we aim to develop a DT model that captures the behavior of a MEC node supporting services with varying workloads. In pursuit of this objective, we adopt a data-driven methodology that effectively learn a model predicting three critical key performance indicators (KPIs): throughput, computational load, and power consumption. To demonstrate the viability and potential of such approach, a measurement campaign is conducted on MEC nodes deployed with different virtualization environments (bare metal, virtual machine, and containerized), and the results are used to build the DT of each node. Furthermore, machine learning models, including k-nearest neighbors (KNN), support vector regression (SVR), and polynomial fitting (PF), are used to understand the amount of actual measurements required to achieve a suitably low KPI prediction error. The results of this study provide a basis for further research in the field of MEC DT models and carbon footprint-aware orchestration.**

***Index Terms*—MEC, 5G, Digital Twin, Green Communications.**

I. Introduction

The integration of multi-access edge computing into the next generation of mobile networks is widely acknowledged as a crucial step to enable innovative services and use cases. The separation of control and user planes has facilitated the integration of edge computing, allowing for the virtualization and management of network functions through the NFV/SDN paradigm. This shift away from dedicated, expensive hardware has opened up new possibilities for the deployment and management of network functions and services.

However, the high complexity and dynamicity of the edge ecosystem bring additional complexity to network management. These aspects have given rise to new orchestration challenges to realize an intent-based networking mechanism where the intents (e.g., service requests) are translated into management actions on the network to ensure a correct trade-off between the requested Quality of Service and network optimization. The performance of services at the edge depends on the resource availability, which might often be limited. Moreover, power consumption is becoming a relevant issue to consider in distributed systems and, in the case of MEC, energy efficiency depends on many factors such as hardware, virtualization technology, and software that might be used for deployment. Trade-offs between performance and power consumption must be considered for orchestration actions, such as edge node selection for service deployment or task offloading [1].

The concept of a digital twin (DT), a virtual representation of a physical system, is emerging as a promising solution to address orchestration and power consumption issues. By creating a DT of the MEC node, it is possible to simulate its behavior, analyze its performance, and optimize its operations in a virtual environment before implementing these changes in the physical system. The existing works on DT mainly focus on analytically deriving the MEC DT model -i.e., the behavior of edge nodes in terms of key performance indicator (KPI)- using mathematical reasoning and assumptions to model it and design optimization algorithms. However, given the high heterogeneity of the edge ecosystem, this approach might not be sufficient to tackle the orchestration challenges effectively. In this work we propose a data-driven approach to gain insights into the behavior of MEC nodes and to accurately predict their performance under varying workloads demand. To the best of our knowledge, a data-driven approach to achieve an online and self-learning DT model is essential to cope with the heterogeneity of the MEC ecosystem, while also handling possible MEC reconfiguration.

The contributions of the paper are as follows:

- We propose a data-driven MEC node performance analysis based on actual measurements.
- We define a data-driven DT representation of the performance of MEC nodes.
- We apply machine learning (ML) to improve the accuracy of the DT representation for different populations of training sets.

In the remainder of the paper the related works are presented in section II, while in section III we detail the rationale behind this work and underline the followed methodology. Then, in section IV we evaluate the proposed approach, and in section V we conclude the paper.

II. Related works

A. Power consumption measurements and modelling

In [2] an overview is presented summarizing the challenges, approaches and results of the state-of-the art research concerning the empirical measurements and analytical modelling of virtual entities power consumption in the telco cloud.

The performance of different virtualization technologies under multiple workloads have been analyzed by several studies, showing how each virtualization technology has it own specific advantages as well as a different impact on using of hardware resources. Containers have been proved to be more CPU and memory efficient compared to unikernels and virtual machines (although the lack of isolation might be a concern in specific use-cases)[3]. Container virtualization also registered a lower power consumption especially during the networking workloads [4]. A more extensive comparison can be found in [5], which analyses the power and energy consumption of four of the most adopted hypervisors and container engines across six hardware platforms and multiple workloads.

Other efforts focus on the design of models for the estimation of power consumption without relying on direct measurements. The survey in [6] benchmarks and evaluates the performance of 24 state-of-the-art power consumption models on different server architectures. The results highlight how interpolation and support vector machine (SVM) have the lowest error for single and multi variable models respectively.

B. Digital Twin and MEC

DT has gained a lot of attention recently as an effective tool for modelling simple as well as complex systems. The combination of MEC and DT can be leveraged to improve the network performance by optimizing the task offloading and resource allocation. Authors [1] define the digital twin edge network (DITEN) paradigm and show how it can be used to bridge the physical edge system and the digital space. In their work, they also provide a comprehensive survey.

In [7] a task offloading latency minimization problem while optimizing the transmit power of UEs is proposed. The DT model is derived analytically using mathematical reasoning and assumptions for modelling the processing capacity, latency and energy consumption. In [8], DT-assisted task offloading is modelled as a Markov decision problem. A mathematical optimization model is used to reduce the system delay and power consumption. In [9] a DT architecture to improve task offloading and task caching techniques is proposed. The aim is to minimise the E2E task offloading delay while bounding the energy consumption to a maximum value. Also in this work, the DT model is derived analytically using mathematical reasoning and assumptions to derive the DT predictions.

C. Our contribution

It is worth noticing that the MEC ecosystem is expected to be highly heterogeneous with varying performance from device to device depending on many factors such as, hardware, software, and virtualization technology. Further, a DT model should be adaptive to account for possible HW/SW updates. However, most of the works in literature focus on optimisation strategies assuming an analytically derived model of the DT which, in our opinion, does not meet the needs of a real scenario with high heterogeneity as discussed before. To the best of the knowledge of the authors, data-driven strategies to develop an online self-learning DT model of a network and its performance is still to be investigated. Therefore, the main novelties of the proposed approach include:

- A data-driven MEC performance analysis based on actual measurements.
- A demonstration of the high heterogeneity of performance profiles based on different variables of the system.
- A data-driven DT representation of the performance of MEC nodes.
- The usage of ML to improve the accuracy of the DT representation in sparse training sets.

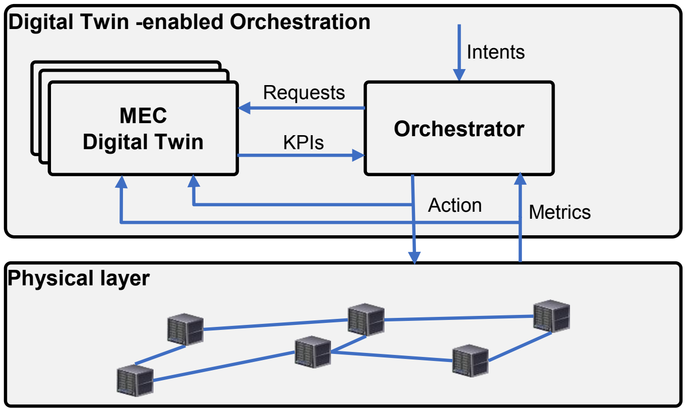


Fig. 1. Conceptual architecture of the MEC Digital Twin -enabled orchestration envisioned in this work.

III. Motivation and proposed methodology

In this paper we envision a DT-enabled orchestration architecture as highlighted in Fig. 1. Indeed, the DT plays a crucial role in managing the network if used for simulating possible what-if scenarios or analyzing the potential impact of different strategies. Indeed, it allows the orchestration layer to take charge of network intents, i.e., services to be deployed, and to exploit the DT models to understand the network behavior before actually applying changes to the physical network. By introducing the DT in the learning loop of the orchestration, the orchestration layer receives predicted KPIs to evaluate the expected network behavior. After taking actions on the physical network, the orchestration layer informs the MEC DT and forwards relevant metrics to the twin, allowing it to learn a digital representation of the physical MEC performance and predict future requests.

In this paper, our focus is on developing a DT model of a MEC system following a measurement based approach.

We consider the aggregated data-rate and CPU demand as the input requests. Conversely, the performance KPIs of interest are the achieved data-rate, the CPU load, and the MEC power consumption.

In the first stage of our study, we conducted a measurement campaign on MECs with various virtualization technologies under different loads. In order to evaluate the performance of MEC, we generated data traffic with varying CPU loads that imitate the communication/computation demands of applications running on the MEC. This was done by considering different workload profiles (WPs), with each WP representing a specific requested load in the MEC. We simultaneously collected the relevant KPIs such as achieved data-rate, CPU-load, and power consumption of the MEC. It's important to note that, in addition to network KPIs which play a crucial role in providing the required service QoS, the power consumption is also an important KPI to model in order to enable a carbon footprint aware orchestration.

In our measurement campaign we considered three type of MEC deployments where the 5G core network is either directly installed in the bare metal (BM), deployed within a virtual machine (VM), or deployed in a containerized environment (CT). As mentioned already, previous works already investigated the performance of different virtualization technologies in running specific applications. However, in this work the aim is to evaluate the power consumption in each scenarios as well in order to understand and model the trade-off between network KPI and power consumption.

In a next stage of our research, we focused on using ML to model the MEC DT. The goal is to compare the results of the learned model to the real measurements obtained in the measurement campaign, and evaluate its accuracy. To do this, we adopted three different ML algorithms, namely k-nearest neighbors (KNN), support vector regression (SVR), and polynomial fitting (PF).

Regarding the KNN, the parameter specifying the number of neighbors has been set to $k = 5$. This value has been selected since appeared to be a sensible trade-off to prevent high bias and high complexity, happening with high k, and sensitivity to noise, happening with a low value of k.

For the SVR, we used radial basis function (RBF) kernel which is commonly recognised as one of the most well-performing kernels in many cases. Moreover, we set the regularization parameter $C = 100$ since performing cross-validation it appears to be the best compromise to avoid under-fitting (low C) and over-fitting (high C).

The polynomial fitting method is proposed in this paper as an analytical model of the different MEC deployments. With the aim to provide a basis for further research, the collected data-set and the methodology and parameters of the polynomial fitting are provided in the Appendix A of the paper and available online. The goal is to encourage the replication of our results and facilitate the advancement of related research, leveraging the findings and insights obtained in this work.

These models are trained with the collected measurements to understand how they perform, and to determine which algorithm provides the best results.

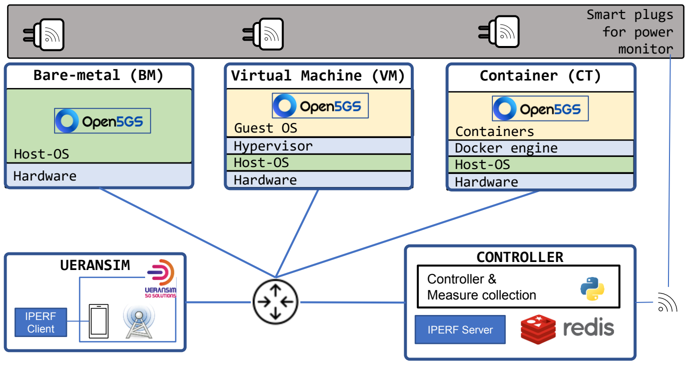


Fig. 2. Testbed set-up for the proposed investigation.

IV. Experimental Results

This section provides the details about the experimentation proposed study. We first provide the details on the environment used for the performance evaluation, followed by a description on the experiments. Later, we provide details on the measured KPIs and the MEC performances under different virtualization technology. Finally, we provide the details on the prediction performance of the MEC behaviour using the three types of ML algorithm to model the MEC system.

A. Testbed setup

For our experimentation we built a testbed composed by 5 Intel next unit of computing (NUC) connected via a fast switch, as shown in Fig. 2. The rationale behind the setup is to demonstrate 5G-enabled MECs with different types of virtualization technologies. In each MEC we leverage the Open5GS[1] software to realise the 5G Core Network, which has been deployed with three configurations: BM, VM and CT. The three types of MECs, are realised with the same type of NUC which is equipped with a $i5 - 7260U$ CPU @2.20GHz. This is essential in order to have comparable results between the three types of MEC deployments. Conversely, the nodes to emulate the radio access network (RAN) and the controller are realised with two NUCs equipped with a $i5 - 1145G7$ CPU @2.60GHz. All the NUCs are connected toghether via a TL-SG105E five-ports Gigabit switch.

In the VM case, we utilize the quick emulator (QEMU) hypervisor with kernel-based virtual machine (KVM) acceleration as it is widely accepted for its cost-effectiveness and high performance. On the other hand, for the CT deployment, we opt for Docker to create container hosting the 5G core (5GC) functions. This decision was made to simplify the process and avoid unnecessary overhead that comes with other solutions like Kubernetes, since we only target a single MEC.

The RAN is established with the use of UERANSIM[2] which is an emulator designed to simplify the deployment of an E2E 5G network. Upon initiation, the user equipment (UE) process effectively establishes a data-plane connection between the UE

[1] https://open5gs.org
[2] https://github.com/aligungr/UERANSIM

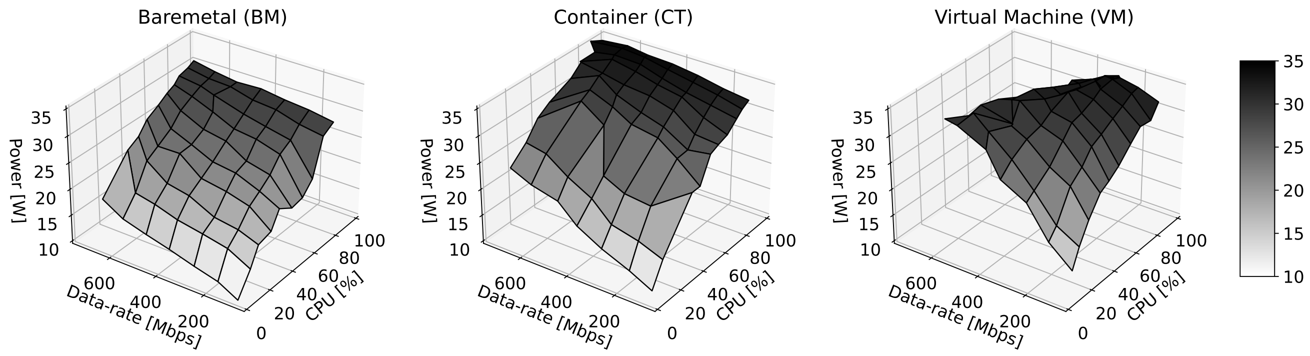


Fig. 3. Relations between the measured KPIs for each type of MEC.

and the selected Open5GS deployment. Although UERANSIM does not include the protocol stack below the radio resource control (RRC), making it unsuitable for scenarios where lower layers are needed, this limitation is deemed beyond the scope of this paper. We thus resort to this solution for its simplicity.

In our experiments, the MEC nodes were subjected to various combinations of CPU and data-rate loads. The CPU load is generated using *stress-ng* allowing to set an overall CPU load, in percentage, within a NUC. Conversely, the data-rate is generated by deploying an *Iperf* client at the UE side and connecting it to the server in the controller node.

The Controller node oversees the generation of experiments and the collection of measurements. It hosts an Iperf server for terminating the data traffic generated by an UE and flowing through the desired MEC. In each experiment we gather the relevant KPIs related to a MEC such as power consumption, overall data-rate, and the CPU usage. To measure the CPU usage and data-rate KPIs in the MEC nodes, we utilize the *psutil* Python package. After proper instruction from the controller, the MEC nodes report the collected measurement to the Redis[3] database hosted in the Controller. On the other hands, to measure the power consumption KPI each NUC hosting a MEC node is connected to a Meross MSS310 smart plug that monitors its power consumption. To gather those values, the Controller nodes queries the smart plugs via a WiFi connection and send them to Redis. All the measures are collected every second and stored in the Redis database, allowing us to have a centralized repository of all the gathered data. This makes it easier to analyze and interpret the results of the experiments.

B. MEC performance and measured KPIs

We first analyze the power consumption and performance trade-offs in the evaluated MEC deployments, namely BM, VM, and CT. Each MEC was tested by setting WPs with CPU loads ranging from 0% to 90% and data-rates between 100 and 700 Mbps. For each WP, we gathered measurements every second within a 30-second time frame. The experiments are depicted in Figure 3, where each plot displays the relationship between the measured KPIs (power consumption, CPU load, and data-rate) for each type of MEC. The aim of this measurement campaign extends beyond investigating the impact of virtualization on MEC performance; it also aims to create a data-set to study the prediction performance of the MEC DT which is detailed in the next sections.

By examining the results under high data-rate and low CPU usage, we observe that the VM is the one consuming the highest amount of power, while the BM is the most efficient, as expected. The CT falls in between the two, with results that are close to those of the BM. Although it may not be clearly visible from the figure, it is worth mentioning that the higher power consumption in the VM can be attributed to its greater CPU usage in handling high data-rates. On the other hand, the BM MEC is the most efficient in traffic management, and the CT case represents a compromise between the two.

When the set WP demands higher CPU loads and high traffic, we see that the CT's power consumption is always greater than the BM. When the imposed demand is the highest which corresponds to 90% of CPU load and 700Mbps of traffic, the CT consumes about 5W more than the BM. However, it is still able to achieve the same BM data-rate without impairments. On the other hand, the power consumption for the VM appears to decrease significantly under high data-rate demand. This decrease is actually the result of impairments in the data-rate, which drops from the requested 700Mbps to an actual 250Mbps when the CPU and data-rate demand is the highest.

The lower data-rate achieved with the VM can be attributed to several factors. It is worth mentioning that the 5GC virtual machine is run using QEMU, a type-2 hypervisor, with KVM acceleration for improved performance. Despite the KVM acceleration, the QEMU VM adds an extra layer of virtualization, which can result in overhead and reduced performance. Furthermore, Docker utilizes a different networking approach compared to virtual machines, which can provide better speed and reduced latency due to leveraging the host's network stack, unlike virtual machines that use virtual network interface cards and incur additional overhead.

C. Digital Twin prediction under complete training data-set

The aim of this analysis is to determine the accuracy of the DT in predicting the KPIs of the MEC deployments under

[3] https://hub.docker.com/r/redislabs/redistimeseries

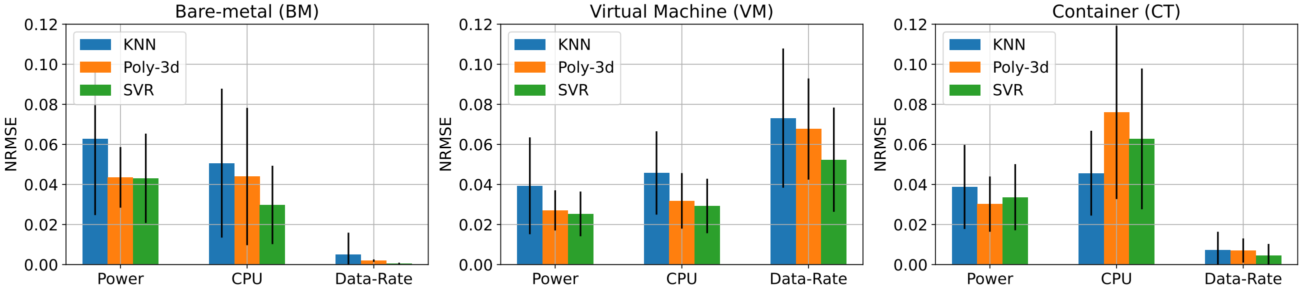
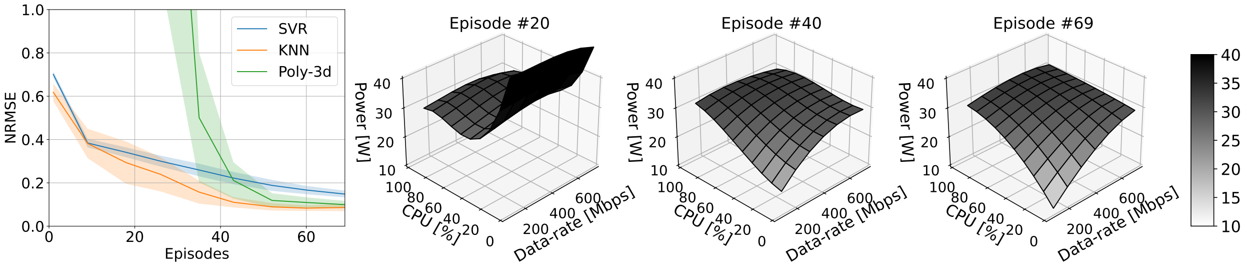


Fig. 4. DT KPI prediction error in the different MEC deployments. The error bars represent the NRMSE standard deviation across the different WPs.



(a) (b)

Fig. 5. Prediction error with incomplete training-set for the VM MEC. In (a) we show the NRMSE vs. the number of episodes, in each episode a new randomly picked WP is added to the training set. Considering one single run, in (b) we plot the predicted power consumption for three episodes.

different WPs. Building upon the findings discussed in the prior section, we delve deeper to evaluate the forecasting accuracy of a DT using the data-set obtained from aforementioned measurement campaign. As explained in section III, we evaluate three type of ML algorithms, namely KNN, RBF and PF. To do so, we use a subset of the collected data to train the DT by considering all the WPs but using $2/3$ of the samples. To determine the effectiveness of the DT in accurately predicting the behavior of the MEC deployments, we use the remaining samples as a validation set.

The results of the DT prediction performance are depicted in Fig. 4. The figure displays the normalized root mean square error (NRMSE) for each predicted KPI for each MEC deployment. The NRMSE metric was chosen as it provides a percentage representation of the error relative to the absolute value of each KPI.

The low prediction error observed in all the cases highlights a good accuracy of all the three methods employed demonstrating their applicability to realise the DT prediction. However, our findings indicate that the SVR model performs better than KNN and PF models in the majority of cases. Nevertheless, the PF still displayed noteworthy performance, proving the viability of utilizing analytical models for further research in this area. In the CT scenario, the error on the CPU is more significant because of the observed variability in CPU usage, which may be due to scheduling problems with Docker. Interestingly, compared to the other deployment scenarios, the prediction error for the data-rate in the VM scenario is higher. This is because of the data-rate degradation explained in the previous section and shown in Fig. 3 which makes more challenging in the VM case to forecast the data-rate accurately.

D. Model prediction under incomplete data-set

In this section, we analyze how accurate the DT forecast is with sparse data, as opposed to having a complete set of WPs available for training as was analyzed in the previous section. Training is conducted in several episodes, with measurements for a randomly selected WP being added for each episode, representing the variable measurements that might be acquired by the networking infrastructure. To evaluate the performance of the different ML algorithms, the results in terms of NRMSE were averaged over 20 trials, where in each trial a different WP was randomly selected as the test set, while the remaining WP were used for training. The goal of this approach is to understand the extent to which the different ML approaches are able to generalize the prediction when working with incomplete data, and to provide insight on how much the WP domain shall be explored before to achieve an acceptable prediction error.

In Fig. 5(a), we report the results showing the NRMSE against the number of episodes. These results pertain to the experiments carried out on VM MEC. As expected, the error in prediction decreases as the number of episodes increases. Interestingly, while the SVR method outperformed the others techniques under complete training sets, the KNN algorithm achieved a better prediction accuracy when the training data is incomplete. This is because of KNN only relies on the proximity of the data points in the feature space, conversely the SVR is a model-based algorithm which potentially leads to biased or incorrect model fit under missing data.

The evaluation of the PF method reveals a substantial prediction error until episode 40. This can be attributed to the polynomial fitting's tendency to produce extreme spikes when evaluating it outside the range covered by the training set. This is demonstrated in the plot of predicted power consumption for different episodes in Fig. 5(b). If the training set is sparse, the polynomial fitting may be quite ineffective, as highlighted in episode 20. However, the prediction is significantly better as in episode 40 and eventually stabilizes later on. This underlines that the polynomial fitting method can be used to improve the accuracy of the DT representation, even if it requires a suitable number of samples to properly capture the characteristics of the physical twin. To encourage the usage of this concept to model networks and MEC in particular, the polynomial fitting model is shared online in the Appendix A.

V. Conclusions

In this work we proposed a data driven approach to realise a MEC DT. We first performed a measurement campaign analysing the impact of the virtualization on the MEC performance and power consumption using a lab testbed. Using the collected data-set, we explored how a DT can be effectively realised using three types of ML techniques to predict the MEC behaviour. The results demonstrate that the approach can effectively be used in order to predict the MEC behaviours. Such prediction capabilities potentially enables an orchestration layer to learn off-line the best action to take before to act on the physical network. This will be subject of further work on the topic.

Appendix

A. Polynomial fitting

The process for developing the MEC analytical model using polynomial fitting is as follows. Let c and t be the requested WP for the MEC expressing the CPU and throughput demand, respectively. Let $\{\hat{c}, \hat{t}, \hat{p}\}$ be the predicted KPIs for the overall CPU load, throughput, and power consumption. To model the MEC behaviour with respect to each KPI, we define the polynomial equation in (1), where $p_{kpi}^{i,k}$ are the polynomial parameters as shown in Table I.

$$F_{kpi}(c,t) = \sum_{i} \sum_{k} p_{kpi}^{i,k} c^i t^i, \quad \forall KPI \in \{\hat{c}, \hat{t}, \hat{p}\} \qquad (1)$$

TABLE I
Polynomial fittings.

	k=0	k=1	k=2	k=3
i=0	$p_{kpi}^{0,0}$	$p_{kpi}^{0,1}$	$p_{kpi}^{0,2}$	$p_{kpi}^{0,3}$
i=1	$p_{kpi}^{1,0}$	$p_{kpi}^{1,1}$	$p_{kpi}^{1,2}$	0
i=2	$p_{kpi}^{2,0}$	$p_{kpi}^{2,1}$	0	0
i=3	$p_{kpi}^{3,0}$	0	0	0

For each KPI, we use the non-linear least squares method to fit the set of observations with the non-linear equation (1). The derived analytical models based and the measurement are available online[4]. As an exemplification, in Table II we report the PF parameters for the BM case. It is worth to highlight that this model is valid within the considered range of WPs.

TABLE II
Fitting parameters for Baremetal MEC.

KPI	$\hat{c}$	$\hat{t}$	$\hat{p}$
$p_{kpi}^{0,0}$	5.425	3.208	7565.895
$p_{kpi}^{0,1}$	0.689	-7.570e-2	308.480
$p_{kpi}^{0,2}$	2.88e-3	1.204e-3	-4.116
$p_{kpi}^{0,3}$	-8.433e-06	-8.066e-06	3.087e-2
$p_{kpi}^{1,0}$	5.265e-2	1.004	22.686
$p_{kpi}^{1,1}$	9.978e-4	3.888e-4	0.470
$p_{kpi}^{1,2}$	-8.702e-06	-1.712e-06	-3.167e-3
$p_{kpi}^{2,0}$	-1.753e-4	9.015e-05	-4.634e-2
$p_{kpi}^{2,1}$	-3.886e-07	-2.214e-07	-3.232e-4
$p_{kpi}^{3,0}$	1.862e-07	-7.497e-08	4.503e-05

Acknowledgment

The work leading to this publication was partially funded by the European Union's Horizon Europe under Grant Agreement no. 101096342 (HORSE project) and Grant Agreement no. 101096925 (6Green project).

References

[1] F. Tang, X. Chen, T. K. Rodrigues, M. Zhao, and N. Kato, "Survey on digital twin edge networks (diten) toward 6g," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 1360–1381, 2022.

[2] E.-V. Depasquale, F. Davoli, and H. Rajput, "Dynamics of research into modeling the power consumption of virtual entities used in the telco cloud," *Sensors*, vol. 23, no. 1, p. 255, Dec 2022. [Online]. Available: http://dx.doi.org/10.3390/s23010255

[3] R. Behravesh, E. Coronado, and R. Riggio, "Performance evaluation on virtualization technologies for nfv deployment in 5g networks," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 24–29.

[4] R. Morabito, "Power consumption of virtualization technologies: An empirical investigation," in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, 2015, pp. 522–527.

[5] C. Jiang, Y. Wang, D. Ou, Y. Li, J. Zhang, J. Wan, B. Luo, and W. Shi, "Energy efficiency comparison of hypervisors," *Sustainable Computing: Informatics and Systems*, vol. 22, pp. 311–321, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210537917300963

[6] L. Ismail and H. Materwala, "Computing server power modeling in a data center: Survey, taxonomy, and performance evaluation," *ACM Comput. Surv.*, vol. 53, no. 3, jun 2020. [Online]. Available: https://doi.org/10.1145/3390605

[7] T. Do-Duy, D. Van Huynh, O. A. Dobre, B. Canberk, and T. Q. Duong, "Digital twin-aided intelligent offloading with edge selection in mobile edge computing," *IEEE Wireless Communications Letters*, vol. 11, no. 4, pp. 806–810, 2022.

[8] T. Liu, L. Tang, W. Wang, Q. Chen, and X. Zeng, "Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1427–1444, 2022.

[9] D. Van Huynh, S. R. Khosravirad, A. Masaracchia, O. A. Dobre, and T. Q. Duong, "Edge intelligence-based ultra-reliable and low-latency communications for digital twin-enabled metaverse," *IEEE Wireless Communications Letters*, vol. 11, no. 8, pp. 1733–1737, 2022.

[4] https://github.com/RiccardoFedrizzi/MEC_DT_model