CloudSkin: ML-based Smart Management for the Cloud-Edge Continuum

Peini Liu*†, Anna Ma. Nestorov*†, Marc Palacín*, Ramon Nou*, Jordi Guitart*†, Josep Ll. Berral*†
*Barcelona Supercomputing Center, Barcelona, Spain
†Universitat Politècnica de Catalunya, Barcelona, Spain
E-mail:{peini.liu, anna.nestorov, marc.palacin, ramon.nou}@bsc.es, {jordi.guitart, josep.ll.berral}@upc.edu

***Abstract*—The CloudSkin project embraces ML-based autonomous management technologies, introducing the Learning Plane (LP), which provides services and methods to embed intelligence in Cloud-Edge Continuum management layers. By leveraging AI, the continuum can achieve user demands for resource, privacy, resilience, and energy efficiency requirements, in an adaptive way. As a first Proof-of-Concept, we envision a smart orchestrator to drive a 5G-powered automotive usecase, focusing on a mobility scenario where the LP optimises resources for AI data and applications, on a transversal Edge environment.**

***Index Terms*—Cloud-Edge Continuum, Orchestration, Machine Learning, Workload Characterization, Edge AI.**

I. Introduction

The emergence of the Cloud-Edge Computing Continuum is driving the development of innovation in various sectors such as smart cities, smart transportation, healthcare, and industry 4.0. To achieve those scenarios, it is critical to develop distributed services that handle large-scale data generation, gathering, storage, and real-time analysis. However, the nature of this distributed paradigm brings complexities to continuum management, including orchestration, performance sustainability, resilience, and scalability. To this end, CloudSkin embraces ML-based autonomous management technologies, introducing the LP, which provides services and methods to embed intelligence in Cloud-Edge Continuum management layers. As shown in Figure 1, the proposed LP covers the management of such distributed modelling, as a layer connected to the Data Plane (DP) retrieving data from the system and environment, and to the Control Plane (CP) providing recommendation and prediction services, oriented to management and orchestration for either centralised and distributed management agents in the Cloud-Edge Continuum.

II. Methodology

The principal research on CloudSkin on smart orchestration focuses on the design of the LP, which focuses on three functionalities, a) System modeling and workload characterization from the telemetry, b) Model storage and federation and c) Model provisioning service for recommendations and predictions.

A. System Modeling and Workload Characterisation

A set of algorithms are explored for system modelling and workload characterisation.

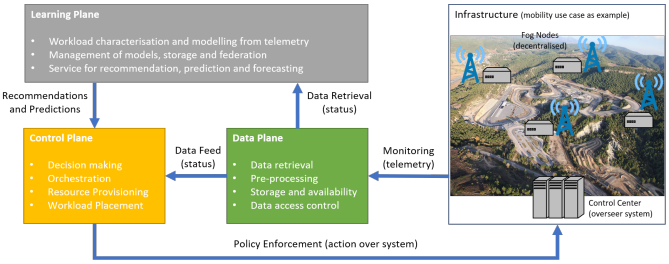

Fig. 1. Learning Plane for System Modelling, Model Management, Storage and Provision through Prediction

*1) **Periodic Behavior Detection - ThetaScan**:* The ThetaScan method [1] uses time-oriented statistical learning to detect statistical properties in telemetry as multi-variate time-series. Computing requests using solely the statistics of the previous time window presents some weaknesses when provisioning periodic behaviours. The Theta-Scan method is intended for detecting periodicity, leveraging the Theta-Model. In this work, we introduce our implementation of the Theta-Model based on the Simplified Exponential Smoothing method (SES), also known as Holt-Winters smoothing [2], with a detrending and deseasonalization of the analyzed time series.

The targeted behaviors to be detected in the CPU/Memory consumption are related to the Trend and Seasonality (i.e., periodic patterns). Modeling the time-series will involve a detrending process, where we extract (and keep) the trend of the time-series, separating it from the potential seasonality and drift. Next, the resulting series are deseasonalized, where we extract (and keep) a potential season (periodic pattern) in the series, separating it from the drift. And finally we can model the drift for future prediction. Forecasting using this model requires to extend the trace from SES, then add the found seasonal pattern and the found trend. Figure 2 shows an execution with hourly and daily periodicity, provisioning results according to our multi-scale policy is presented.

Summarizing, the Theta-Scan algorithm attempts the automated detection of periodicity towards next period provisioning of resources, with the objective is to leverage the method to detect **periodic behaviors** from the LP, towards provisioning or placing applications and resources in advance.

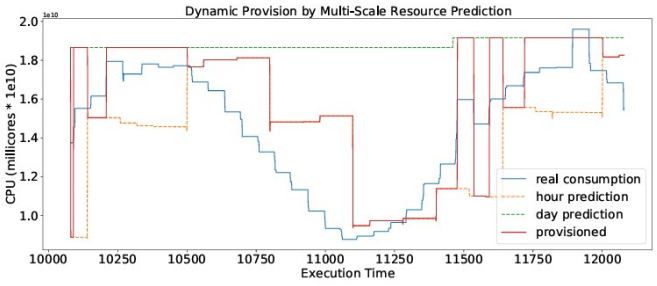

Fig. 2. Example of hourly and daily CPU forecasting, selecting the $t-1$ best forecaster for provisioning

*2) **Behavior Similarity Detection - AI4DL**:* While ThetaS-can is intended to detect periodic patterns in workloads, another approach to detect and discover patterns is to cluster telemetry traces, to be later examined by the system architects (human method) or used for statistical estimation (automatic method). The next technology is AI4DL [3] [4] created in a collaboration between IBM and the Barcelona Supercomputing Center, purposed to characterize Cloud containerization for Deep Learning applications, but we consider its application for any kind of workload to be introduced in the CloudSkin ecosystem for **non-periodic behaviors** on applications.

The proposed methodology collects container resource usage (i.e., CPU and memory, but also network or disk), creates a model encoding these metrics to capture dynamics over the time dimension (behavior), clusters similar behaviors as unique phases, reduces the whole execution to a sequence of phases, and then estimates the resource requests per each phase. Figure 3 shows an example of real trace with differentiated phases detected by AI4DL.

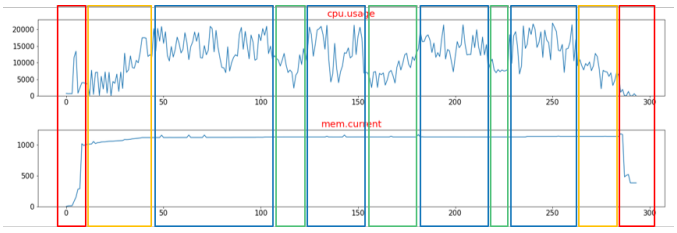

Fig. 3. Phase discovery for multi-variate time series on telemetry using AI4DL

A full life cycle of a containerized application can be encoded as a sequence of phases, representing behavior changes in resource usage. Containers running similar workloads display similar behaviors (e.g., first phases are corresponding to $Memory.load$, next phases to $Intensive.CPU$, last phases to $Memory.unload$). A good representation of phase sequences can indicate what types of behaviors the containers are undergoing, how much resources they need, and in which order they consume resources. The resource usage of different containers may show similar patterns in their phase sequences. E.g., an auto-scaler, resource provisioning or policy enforcement mechanism can leverage behavior recognition, and use the estimated resource usages for a given sequence of phases, to proactively estimate and provision future resource demands.

*3) **Transformer-based Methods (WIP)**:* Finally, we are focusing our research on resource prediction through state-of-art attention neural networks, such as Transformers, to learn specific patterns to be discovered or forecasted. In the CloudSkin project, we are progressing in the following directions:

- The study and research of novel modeling methods based on Transformers for proper prediction and anticipation of sudden behaviors towards resource allocation, putting specific emphasis in the correct prediction of patterns like sudden changes. The objective is to learn and recognize common patterns that can precede a sudden change in the resource consumption, while keeping the model agnostic about an specific workload and its dependencies, only obtaining the required information from past values.
- The creation of new evaluation strategies to assess proper allocation of resources during the execution of a workload, and the correct prediction in time and amplitude of resource consumption changes.

B. Model Federation and Storage

For the CloudSkin project, the Learning Plane will start with a centralized approach, where all control and management operations are in the Cloud, while the aim during the project is to explore ways to off-load such management to the Edge, allowing scalability and autonomy to the different Edge regions. The basic scenario corresponds to a centralized system where management and orchestration relies in a single location (single- or multi-node). As shown in Figure 4 (left), a centralized system must have visibility and control of all the infrastructure, concentrating all decision making on Edge placement and provisioning. The Planner and Learning applications are in a central Cloud, while the storage can still be distributed for the sake of collecting telemetry from the Edge nodes without pushing all data to the central Cloud. For the early designs of CloudSkin, we are pushing for this approach for its initial simplicity regarding to the proposed use cases.

However, an alternative is a distributed system, where management and decisions are distributed across Edge nodes (or intermediate Fog nodes), as shown in Figure 4 (right). In case of large-scale systems that must maintain certain autonomy, the Control and Learning Plane can be distributed (along with the already distributed Data Plane). For this, each Edge node or group of Edge nodes can coordinate locally and later report or coordinate with the Central Cloud. Each Edge node/group would perform orchestration and forecasting of applications autonomously, leveraging the distributed repository to share models, applications, learning applications, meta-data and telemetry from other nodes, being able to create larger federated schemes on sharing data and models. In this early stage of CloudSkin, we are focusing on a centralized architecture, but without losing sight of distributed architectures, as they

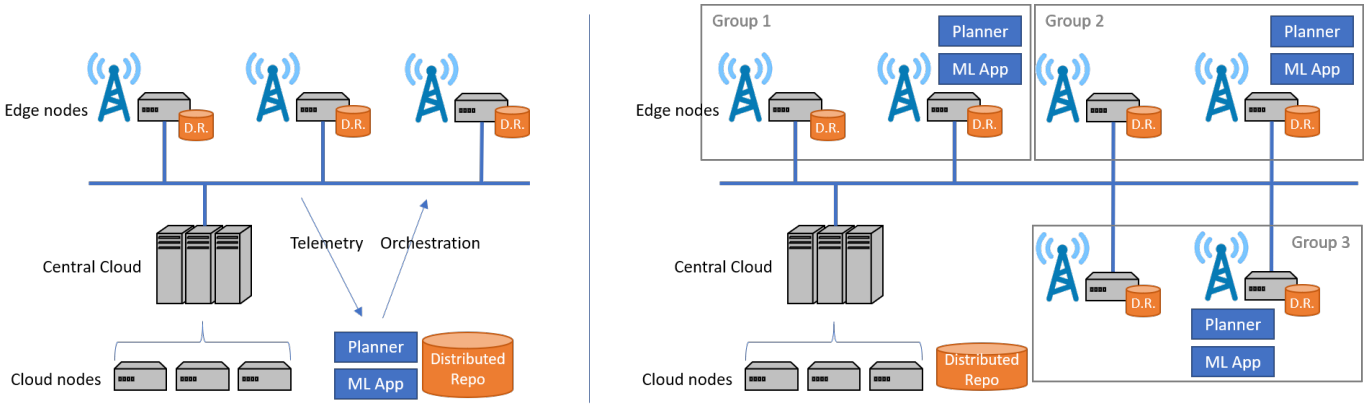

Fig. 4. Example of centralized (left) vs. decentralized (right) schemes where the control and learning is placed in the Cloud nodes or in the Edge nodes

will allow off-loading part of the management load from the Cloud to the Edge.

*1) **Model Federation**:* Model federation functionality involves the distribution of the models towards sharing and aggregation (what is "learned"); also the meta-data related to the models, datasets and execution environments for the algorithms fitting and inferring data using the models (the indications on "how to use the model"); and the objects containing the implementation of those algorithms (the model's "engine" application).

*2) **Storage**:* Different candidates were considered as distributed file-systems and object storage platforms, as an example for the first ones Hadoop HDFS for distributed data with replication [5], [6] and GekkoFS for distributed in-memory volatile data [7] were considered. Such systems are a baseline option without the optimizations required for the current case. Therefore, our preferences are object storage systems, like GEDS [8], providing a distributed storage system with replication and resilience, developed by the IBM partners of CloudSkin.

C. Model Provisioning and Management

This functionality involves the execution of prediction and recommendation, and provisioning algorithms with the retrieved data and learned models.

Current technologies involve containerization of the fitting and inference engines (e.g. PyTorch, TensorFlow, ScikitLearn, Mlflow, Seldon etc.) through Kubernetes and/or K3S on the one hand, and micro-function serverless computing frameworks like Lithops [9] or WebAssembly (WASM) [10] on the other. Actions to be performed include model and meta-data management (e.g., training, prediction and forecasting, model aggregation, model updating), that must be executed near the data, either source or storage.

III. Use Case

As a Proof-of-Concept, the smart orchestrator will be oriented to drive a 5G automotive use case, collaborated among Barcelona Supercomputing Center, Nearby Computing (NBC) and Cellnex Telecom (CNX). The proposed LP can use AI as a driver to orchestrate resources across the Cloud-Edge for AI applications (e.g., video analytics application), where optimising resource provisioning, data movement and application placement are cornerstone elements.

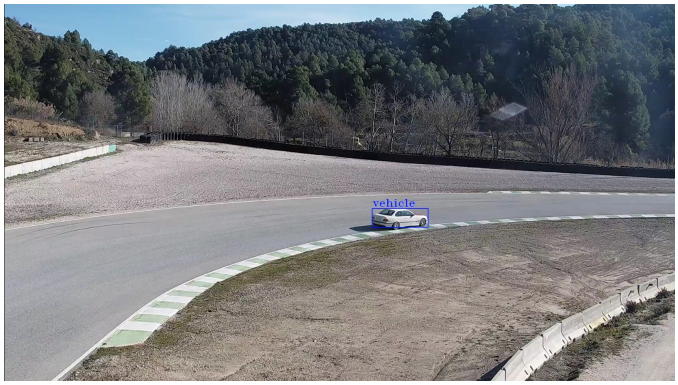

Fig. 5. Vehicle detection from camera streams

This use case considers two scenarios: a network-bound setting, where the workload and data have to follow a user across the Cloud-edge infrastructure (horizontal service migration across edge servers), and a computing-bound situation, where heavy workload needs to be placed closer to the user as much as possible (vertical service migration across the cloud continuum). Therefore, this use case requires a smart orchestrator to automatically deal with the placement and migration of tasks across the Cloud-edge infrastructure and also a universal virtualization abstraction (i.e., containerization) platform that enables the seamless execution of tasks on a wide array of cloud and embedded devices. With this holistic architecture, applications are migratable across different servers and devices in the continuum to effectively support both horizontal and vertical service migration.

A. Usecase 1: Application Placement (WIP)

The basis to advice applications moving among edges or between edge and cloud is to enable the placement of the application in a containerized way in an edge or cloud.

Currently, we are working with NBC and CNX to prepare the infrastructure, building Kubernetes platforms and corresponding toolbox to run our applications (i.e., video analytics application see figure 5). On the other hand, on the integration of the characterization models (AI-based modules) with the schedulers (recommenders), also on a first design and implementation of the LP (i.e., data connectors) upon the CP (i.e., NearbyONE orchestrator).

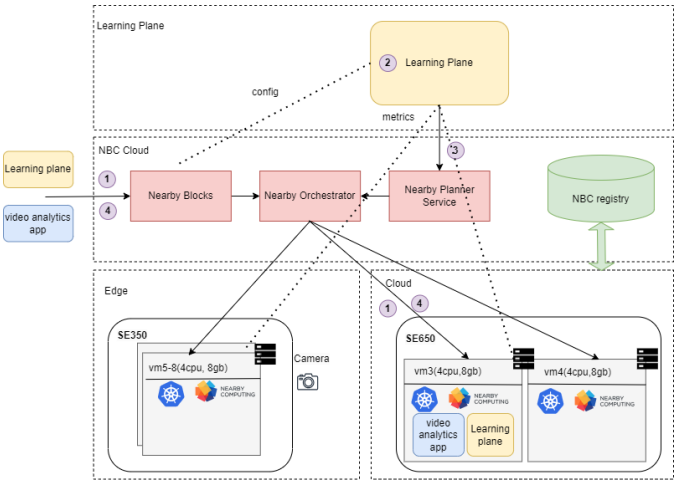

Fig. 6. Workflow of learning Plane on an usecase of application placement and provisioning.

Figure 6 shows the workflow of LP on a usecase of application placement and provisioning. In the example, an application arrives and needs to be deployed. The NBC block service needs to inform the LP of the application information. Then the LP should get the application and the system resource usage in order to forecast the quality of service (QoS) of the application running on different systems, and based on the prediction results the heuristic algorithm will be enabled to select the system for the application placement. The first-time placement will follow a naïve approach meaning the LP will use the specifications of the application from the user or the application placement records from a previous execution to place the application, then after having some data, a "smart" approach will be attempted to forecast some *a-priori* information from the application, in order to place and provision it with less resource waste or more quality of service. Note that the Planner will continuously monitor the application and systems and make decisions on whether to re-provision or migrate the application among systems.

In this example, after the arrival of the application, the Planner will first place the application based on the application specifications, step 1. Then in step 2, the NBC block gets some information from the application, e.g. a profile, and configures it to the LP, as well as the block should provide the list of systems that can place the application, and then deploy the learning plane aligned with the application. The learning plane will run continuously to predict or forecast the QoS of the application on different systems based on the online monitoring data from the application execution and the systems, and calculating the best QoS. Therefore, then the LP could return to the Planner with the system that could achieve the best application QoS (i.e., step 3). Finally, the step 4, the Planner then call the orchestrator to enable the placement/migration of the application in different systems (e.g., edge or cloud).

The "Learning Application" request the Planner through a service, API or remote procedure call. This Learning Application is a machine learning engine, in charge of loading an already trained model capable of performing such predictions or forecasts, performing inference with the application information, the model and additional data such as meta-data or telemetry from the system.

IV. Envisioning Next Steps

To achieve such goals, we plan to deepen on the knowledge learned from system modeling and characterisation, expanding scenarios from specialised telemetry towards holistic monitoring, considering distributed learning and inference, from both application and infrastructure layers. We envision that the continuum can achieve user demands for resource efficiency, privacy, resilience, and energy efficiency requirements by using such emerging smart orchestrators, evolving with new ML/Deep Learning capabilities for day-by-day more complex behavior patterns and problem solving.

Acknowledgment

This work is financed by the EU-HORIZON programme under grant agreements EU-HORIZON GA.101092646, EU-HORIZON MSCA GA.101086248, EU-HORIZON GA.101092644, by Generalitat de Catalunya (AGAUR) GA.2021-SGR-00478, and the Spanish Ministry of Science (MICINN), the Research State Agency (AEI) and European Regional Development Funds (ERDF/FEDER) PID2021-126248OB-I00, MCIN/AEI/10.13039/ 501100011033/FEDER, UE.

References

[1] J. L. Berral, D. Buchaca, C. Herron, C. Wang, and A. Youssef, "Theta-scan: Leveraging behavior-driven forecasting for vertical auto-scaling in container cloud," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, 2021, pp. 404–409.

[2] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207003001134

[3] J. L. Berral, C. Wang, and A. Youssef, "Ai4dl: Mining behaviors of deep learning workloads for resource management," in *Proceedings of the 12th USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'20. USA: USENIX Association, 2020.

[4] D. B. Prats, J. L. Berral, and D. Carrera, "Automatic generation of workload profiles using unsupervised learning pipelines," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 142–155, 2018.

[5] T. White, *Hadoop: The Definitive Guide*, 1st ed. O'Reilly Media, Inc., 2009.
[6] Y. Zhai, J. Tchaye-Kondi, K.-J. Lin, L. Zhu, W. Tao, X. Du, and M. Guizani, "Hadoop perfect file: A fast and memory-efficient metadata access archive file to face small files problem in HDFS," *Journal of Parallel and Distributed Computing*, vol. 156, pp. 119–130, oct 2021. [Online]. Available: https://doi.org/10.1016/j.jpdc.2021.05.011
[7] M.-A. Vef, N. Moti, T. Süß, M. Tacke, T. Tocci, R. Nou, A. Miranda, T. Cortes, and A. Brinkmann, "Gekkofs — a temporary burst buffer file system for hpc applications," *J. Comput. Sci. Technol.*, vol. 35, no. 1, p. 72–91, jan 2020. [Online]. Available: https://doi.org/10.1007/s11390-020-9797-6
[8] IBM, "GEDS Distributed Ephemeral Data Store," https://github.com/IBM/GEDS.
[9] J. Sampé, M. Sánchez-Artigas, G. Vernik, I. Yehekzel, and P. García-López, "Outsourcing data processing jobs with lithops," *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 1026–1037, 2023.
[10] "WebAssembly," https://webassembly.org.