Cognitive and Time Predictable Task Scheduling in Edge-cloud Federation

Somayeh Abdi, Mohammad Ashjaei, Saad Mubeen
Mälardalen University, Västerås, Sweden
firstname.lastname@mdu.se

***Abstract*—In this paper, we present a hierarchical model for time predictable task scheduling in edge-cloud computing architecture for industrial cyber-physical systems. Regarding the scheduling problem, we also investigate the common problem-solving approaches and discuss our preliminary plan to realize the proposed architecture. Furthermore, an Integer linear programming (ILP) model is proposed for task scheduling problem in the cloud layer. The model considers timing and security requirements of applications and the objective is to minimize the financial cost of their execution.**

***Index Terms*—Edge-Cloud federation, Time predictable Task Scheduling, Integer linear programming model, scheduling approaches**

I. Introduction

The edge-cloud computing paradigm plays a crucial role in providing computing and services for Cyber-Physical Systems (CPS). Various tasks related to industrial systems, ranging from infrastructure monitoring and smart automation to smart construction equipment, need computing power and storage capacity provided by cloud and edge computing. Although cloud computing provides enhanced storage and computing capacity, it causes high communication latency. Since edge nodes with limited capacity are located closer to end-systems and devices, utilizing edge computing is a promising strategy to ensure the quality of service (QoS) for time-critical tasks. Consequently, deploying a federation of cloud and edge computing is a key step to benefit from their advantages in CPS [1], [2]. This federation provides new opportunities not only for customers but also for providers. From the customer's perspective, the federation provides scalable services for both compute-intensive and time-critical tasks. It also brings this opportunity to utilize the capacity of perishable resources (e.g. CPU cycles), saving energy, and even avoiding extra costs caused by overprovisioning [3].

Cognitive ability, adaptivity, and timing predictability are important factors that must be considered in industrial CPS since most industrial tasks are time-critical and the environment can be highly dynamic [4]. To meet these requirements, this paper proposes a hierarchical architecture that supports cognitive and time predictable task scheduling in the edge and cloud computing layers. In this context, time predictable task scheduling refers to utilizing techniques that detect whether tasks meet their deadline or not. Time predictable task scheduling is supported by a cognitive adaptation mechanism in the edge-cloud architecture. Moreover, the problem of task scheduling in the cloud layer is formulated as an integer linear programming model. In this work, we mainly focus on the timing and security requirements of applications and an application consists of a set of tasks. The novelty of this architecture is to benefit from a hierarchical model to provide scalable and cognitive resource allocation within different computing layers.

II. Envisioned Architecture

The proposed predictable and cognitive edge-cloud architecture is depicted in Fig. 1. In this architecture, the edge-cloud federation consists of several layers of interconnected computing resources. In general, the edge nodes provide services to end-systems or devices, while a cluster of fog nodes aggregate several edge nodes providing enhanced computing capabilities and connectivity. The cloud computing layer can consist of several layers itself including enterprise (private) and public clouds. This computing architecture can provide computing and storage resources to end-users based on application requirements. To meet timing requirements, due to the limited capacity of edge nodes and fog clusters, time-critical applications should be executed on the edge-fog layer, while less time-critical applications should be executed on the cloud layer. To fulfill security requirements, applications with high security-level may only be executed on the private cloud, while applications with lower security-level may be executed on edge nodes, fog, and cloud layers. With such an architecture, enterprises can take advantage of multilayer computing systems for their tasks with different timing and security requirements. To realize this hierarchical scheduling, three main components are considered in the proposed architecture: cognitive task dispatcher, edge-fog scheduler (Sch_E), and cloud scheduler (Sch_C), as depicted in Fig. 1.

A. Cognitive Task Dispatcher

The goal of cognitive technology is to understand the environment abstractly and enable the system services to make decisions or respond to events adaptively [5]. In this work, the cognitive task dispatcher decides to send an application to the edge-fog or cloud layer based on timing and security requirements in addition to considering the workload of edge nodes and fog clusters. This component can be implemented either centralized or distributed.

B. Cloud and Edge-fog Schedulers

Task scheduling in the edge-cloud federation is challenging due to diversity in applications requirements (e.g., timing and

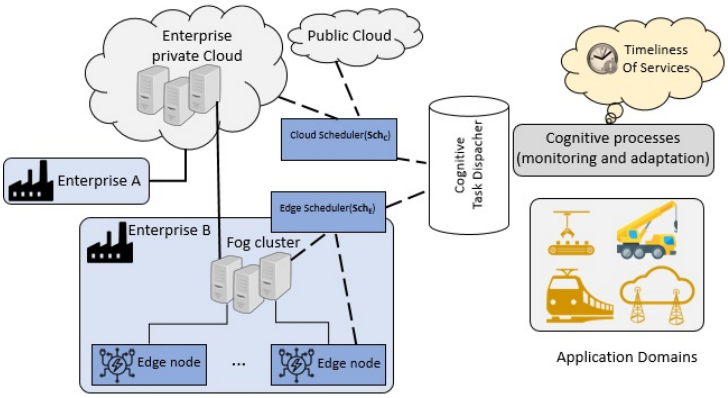

Fig. 1: Proposed multi-layer edge-cloud federation architecture.

security requirements) and diversity in the characteristics of available resources (e.g., performance and etc.). Moreover, the essence of scheduling in edge-fog and cloud layers can be different. In other words, the objective function and constraints of task scheduling in the edge-fog and cloud layers can be greatly different. Because of these reasons, in this architecture, a cloud scheduler and an edge-fog scheduler are considered, as shown in Fig. 1. In the proposed multilayer architecture, the cloud scheduler minimizes the financial cost of executing applications in the cloud computing layer and considers security-aware constraints and the edge-fog scheduler minimizes the waiting time of tasks to meet deadlines of time-critical applications. Although offloading tasks to the closest edge nodes reduces communication latency, the response time of the tasks may increase due to overloading the edge nodes. To meet timing requirements in the edge-fog layer, communication delay, computing delay, and tasks run-time must be considered to detect whether the scheduling of a time-critical application is feasible or not. A task scheduling problem is feasible when all requirements of the application can be fulfilled, such as its deadline, age and reaction constraints [6].

III. Investigation of Problem-solving Approaches

Regarding the scheduling problem, the three most common problem-solving approaches[1] are meta-heuristic algorithms, mathematical models, and machine learning techniques. Even though these approaches aim to find optimal or near-optimal solutions, they are significantly different in the aspect of fundamental techniques, and each of them has its pros and cons. In this section, these approaches are investigated in the aspects of solution quality, time complexity, detecting an infeasible problem, and cognitive ability.

A. Meta-heuristics Approaches

These approaches (e.g. genetic algorithm) may find an admissible approximation of an optimal solution with a low volume of computations, based on designed partial search algorithms. Since these approaches can be easily adapted for a wide range of problems, they are common in computer science. However, they do not guarantee to find optimal solutions. In addition, it is very difficult to detect that a problem is infeasible, using a meta-heuristic algorithm.

B. Machine Learning (ML) Techniques

Utilizing ML techniques enhances the cognitive ability of a system since ML's essence is to learn a model based on a dataset following a specified target. Therefore, accessing an appropriate dataset is crucial for learning a model with high precision. Not only the accuracy of data but also the scale and variety of data are vitally important in improving the model that an ML algorithm learns. Regarding the scheduling problem, on the one hand, ML techniques are useful for online scheduling because they can make decisions in a reasonable time. On the other hand, appropriate datasets are not available because of the diversity in task requirements and resource characteristics [7]. Moreover, ML techniques do not find optimal solutions and it is very difficult to detect that a solution is infeasible.

C. Mathematical Models

With the improvements in parallelization and computing power, applying mathematical optimization has gained more and more attention in research and business domains. Proposing a mathematical model is useful for formulating the objective and constraints of a problem using exact mathematical functions and expressions. Not only does it play a crucial role in obtaining a deep insight into the problem but it also generates globally optimal solutions, which can be used to make optimal decisions. Moreover, a mathematical model detects whether solving a problem is infeasible or not. Nevertheless, some real-world problems are so complex that using exact methods is time-consuming. In these cases, even if the models are too difficult to solve for large instances, it is useful to run the models for small instances and use the optimal solutions to create datasets for utilizing ML techniques or even evaluating the quality of solutions of other approaches, such as meta-heuristics approaches.

[1] https://www.gurobi.com/resource/4-key-advantages-of-using-mathematical-optimization-instead-of-heuristics/

IV. Preliminary Plan To Realize The Architecture

Each of the mentioned problem-solving approaches has its merits and drawbacks. Therefore, utilizing an appropriate combination of them is required to realize cognitive and time predictable task scheduling within the presented multilayer architecture.

To implement cognitive task dispatcher, we aim to utilize ML techniques to increase the cognitive ability of the proposed system. Indeed, the task dispatching problem will be formulated as a classification algorithm, and well-known algorithms such as decision tree, SVM, and NN algorithm will be used to deploy a suitable algorithm with the highest rate of precision. This component decides to send an application to the edge-fog or cloud layer based on its timing and security requirements.

To support time predictable task scheduling in Edge-fog layer, we aim at utilizing approaches that can prove or demonstrate whether the scheduling problem is infeasible or not. As discussed, a mathematical model detects that a problem is infeasible, conversely, other approaches cannot detect it. Moreover, to obtain a deep insight into the scheduling problem and generate optimal solutions, we will formulate the task scheduling problem in the edge-fog layer as a linear model to minimize the waiting time of tasks and fulfill constraints related to task requirements and the limited capacity of resources in this computing layer. Furthermore, we will propose a meta-heuristic approach to find near-optimal solutions in a reasonable time. The results of the meta-heuristic algorithm will be compared with the optimal solutions of the mathematical model.

The novel essence of this architecture is its support for timing predictability in all layers. Indeed, each application has a deadline that must be met; time-critical applications have short deadlines and execute on the edge-fog layer while less time-critical applications have longer deadlines and execute on cloud layers. Therefore, to support time predictable task scheduling in the cloud layer, we formulate the problem as an integer linear programming model to minimize the financial cost of executing applications.

V. Proposed Integer linear programming model for task scheduling in the cloud layer

An integer linear programming is a mathematical optimization in which objective function and constraints are linear. This paper proposes an ILP model for the task scheduling problem is the cloud layer. Indeed, an optimization algorithm is executed in the cloud scheduler, shown in 1. In this computing layer, the private and public clouds provide virtual machines (VMs) with different costs and performance levels. The scheduler selects virtual machine types that are appropriate to minimize financial cost of executing applications.

A. Application and computing model

To solve a scheduling problem, structure of the application must be specified in advance. Indeed, some constraints of a mathematical model are related to characteristics of the application. In this work, we consider bag of tasks (BoT) applications; a BoT application consists of a set of independent tasks $A = \{t_1, t_2, ..., t_n\}$ which $\delta = \mid A \mid$ represents the number of tasks of the application and DS_A indicates the input data size of one task in MB. Moreover, to demonstrate other requirements of an application, we can consider $< App, TST_A, D, RAM_A, Storage_A >$ which $TST_A \in \{private, public\}$ represents security requirement and D indicates deadline of the application. RAM_A, $Storage_A$ indicates minimum required RAM and storage for executing the application. Our proposed mathematical model fulfills constraints related to the required computing and storage resources for executing tasks, deadline of the application, and security requirements. Since in cloud computing layer the private and public clouds provide virtual machines with different performance levels, we use CCU metric as the assessment metric [8]. This metric was developed by CloudHarmony [2] to provide a uniform metric for the performance evaluation of virtual machine types belonging to different IaaS providers. A value of 1 CCU, which is approximately equal to 1 ECU, indicates a CPU capacity of 1.0–1.2 GHz 2007 Opteron or 2007 Xeon processor. Table I lists the characteristics of the virtual machine types provided by the participating clouds in the cloud layer.

TABLE I. Model parameters and their description.

Notation	Description
m	The number of participating clouds in the cloud layer.
CP_k	kth cloud in the cloud layer.
J_k	The set of VM types provided by CP_k in the federation.
VM_{kj}	jth VM type provided by CP_k.
λ	Ready time of VMs
P_{kj}	The fee of running VM_{kj} in dollar per hour.
ccu_{kj}	The performance of VM_{kj} in Cloud Harmony Compute Unit (CCU).
CPU_{kj}	The CPU capacity of VM_{kj}.
RAM_{kj}	The memory capacity of VM_{kj}.
$Stor_{kj}$	The Storage capacity of VM_{kj}.
$CPUC_k$	The CPU capacity of kth cloud.
$RAMC_k$	The memory capacity of kth cloud.
$Stor_k$	The Storage capacity of kth cloud.
CST_k	Security tag of cloud CP_k.
DTR_k	The data transfer rate in MB per second between an end-device and and CP_k
DTC_k	The price of data transfer in $ per MB from an end-device to CP_k

Required auxiliary parameters that are inputs of the proposed mathematical model are as follows:

- $\tau_{kj} = \frac{BT}{ccu_{kj}}$ indicates the execution time of one task of on a VM_{kj}.It is supposed that run time of tasks is known beforehand [3] and BT indicates the run time of one task of the application on a VM with 1 CCU.
- $DTT_k^{in} = \frac{DS_A}{DTR_k \cdot 3600}$ indicates the data transfer time of one task from the end-device to cloud provider CP_k. To calculate data transfer time, the average bandwidth

[2]http://blog.cloudharmony.com/2010/09/benchmarking-of-ec2s-new-cluster.html

[3]https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator

between the end-device and the clouds can be measured using a bandwidth measurement tool iperf[4].

- $\eta_{kj} = \min(\delta, \lfloor \frac{D - DTT\ ^{in}_{k}}{\tau_{kj}} \rfloor)$ determines the maximum number of tasks which can be executed on VM_{kj} provided that the deadline D is met. It should be noted that if the maximum number of tasks can be executed on VM_{kj} is more than δ then η_{kj} attains δ.
- Since cost of running a VM is charged in dollar per hour, it is necessary to round up required time of running each VM. $T_{kj} = \begin{cases} \infty & \eta_{kj} = 0 \\ \lceil \tau_{kj} \cdot \eta_{kj} \rceil & \eta_{kj} > 0 \end{cases}$ denotes rounded up time of running each VM_{kj} in hours for executing tasks. Moreover, T_{kj} attains ∞ for infeasible assignment.

B. Mathematical model

In this model, the objective function is to minimize the financial cost of executing applications. This cost includes the cost of running VMs in dollar per hour and data transfer cost. The model fulfills constraints on timing and security requirements, and resource limits of the private cloud. The objective function and model constraints are as follows:

$$\text{Cost}= \min \sum_{k=1}^{m} \sum_{j \in J_k} P_{kj} \cdot N_{kj} \cdot T_{kj} + \sum_{k=1}^{m} y_k \cdot DS_A \cdot DTC_k$$

subject to:

$$y_k \in \{0,1\} \qquad \forall k \in \{1,..,m\} \tag{1}$$

$$R_{kj} \in \{0,1\} \qquad \forall k \in \{1,..,m\},\ \forall j \in J_k \tag{2}$$

$$N_{kj} \in \mathbb{Z} \quad \forall k \in \{1,..,m\},\ \forall j \in J_k \tag{3}$$

$$\sum_{k=1}^{m} y_k = 1 \tag{4}$$

$$\sum_{j \in J_k} N_{kj} \cdot \eta_{kj} >= \delta \cdot y_k \qquad \forall k \in \{1,..,m\} \tag{5}$$

$$N_{kj} \geq R_{kj} \qquad \forall k \in \{1,..,m\},\ \forall j \in J_k \tag{6}$$

$$N_{kj} \leq N_{kj} \cdot R_{kj} \qquad \forall k \in \{1,..,m\},\ \forall j \in J_k \tag{7}$$

$$y_k \cdot CST_k \leq TST_A \qquad \forall k \in \{1,..,m\} \tag{8}$$

$$R_{kj} \cdot T_{kj} + DTT\ ^{in}_{k} + \lambda \leq D \quad \forall k \in \{1,..,m\}, \forall j \in J_k \tag{9}$$

$$R_{kj} \cdot Stor_{kj} \geq Stor_A \quad \forall k \in \{1,..,m\}, \forall j \in J_k \tag{10}$$

$$R_{kj} \cdot RAM_{kj} \geq RAM_A \quad \forall k \in \{1,..,m\}, \forall j \in J_k \tag{11}$$

$$N_{kj} \cdot Cpu_{kj} \leq CPUC_k\ \forall k \in \{Private\ cloud\}, \forall j \in J_k \tag{12}$$

$$N_{kj} \cdot RAM_{kj} \leq RAMC_k\ \forall k \in \{Private\ cloud\}, \forall j \in J_k \tag{13}$$

$$N_{kj} \cdot StorC_{kj} \leq StorC_k\ \forall k \in \{Private\ cloud\}, \forall j \in J_k \tag{14}$$

The constraints of the proposed model are defined in Eqs. (1)-(14). These constraints are explained in details as follows:

- Eqs. (4)-(7) are related to scheduling policy. Indeed, these equations guarantee that an application is submitted to a cloud and VM types of that cloud are selected for executing its tasks.
- Eqs. (8) insures security requirements of the application. Indeed, we consider security tags for applications and clouds.
- Eqs. (9) fulfills deadline of application A. To fulfill the deadline, the completion time of tasks must be less than or equal the deadline. In this constraint, time of executing tasks, data transfer time and ready time for VMs are considered.
- Eq. (10)-Eq. (11) meet the minimum required storage and RAM for executing tasks.
- Eqs. (12) and (14) fulfill resource limits related to the private cloud.

[4]http://iperf.sourceforge.net/

VI. Conclusion and Ongoing Work

In this paper, we presented the work in progress on the development of a cognitive and time-predictable architecture for task scheduling in the edge-cloud federation. The proposed architecture supports task scheduling for any task domain with strict and non-strict timing and security requirements that employs edge-cloud computing, such as construction vehicles, railways, telecommunication, and many more. What is worth highlighting here is that the proposed architecture benefits from a hierarchical model that can be applied to implement task scheduling among various computing layers. Moreover, this paper proposed an ILP model for task scheduling in the cloud layer with cost minimization objective. The proposed model considers timing and security requirements of tasks. The concrete implementation of various components in the proposed architecture comprises the ongoing work.

Acknowledgement: The work in this paper is supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) through the DESTINE, PROVIDENT and INTERCONNECT projects and KKS foundation through the projects DPAC, HERO and FIESTA.

References

[1] C.-H. Hong, B. Varghese, Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms, ACM Computing Surveys (CSUR) 52 (5) (2019) 1–37.

[2] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, A. Ahmed, Edge computing: A survey, Future Generation Computer Systems 97 (2019) 219–235.

[3] B. Kar, W. Yahya, Y.-D. Lin, A. Ali, A survey on offloading in federated cloud-edge-fog systems with traditional optimization and machine learning, arXiv preprint arXiv:2202.10628 (2022).

[4] M. Ashjaei, S. Mubeen, M. Daneshtalab, V. Casamayor, G. Nelissen, Towards a predictable and cognitive edge-cloud architecture for industrial systems, in: Real-time And intelliGent Edge computing workshop, 2022.

[5] M. Chen, W. Li, Y. Hao, Y. Qian, I. Humar, Edge cognitive computing based smart healthcare system, Future Generation Computer Systems 86 (2018) 403–411.

[6] S. Mubeen, T. Nolte, M. Sjödin, J. Lundbäck, K.-L. Lundbäck, Supporting timing analysis of vehicular embedded systems through the refinement of timing constraints, Software & Systems Modeling 18 (1) (2019) 39–69.

[7] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al., Deep q-learning from demonstrations, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

[8] S. Abdi, L. PourKarimi, M. Ahmadi, F. Zargari, Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds, Future Generation Computer Systems 71 (2017) 113–128.