Task Scheduling Scheme Based on Cost Optimization in 5G/Hetnets C-RAN

Olfa Chabbouh, Nazim Agoulmine, Sonia Ben Rejeb, Zièd Choukair

Task Scheduling Scheme Based on Cost Optimization in 5G/Hetnets C-RAN

Olfa Chabbouh, Nazim Agoulmine, Sonia Ben Rejeb and Zied Choukair

Mediatron Laboratory, High School of Communication of Tunis (Sup'com), Ariana-Tunisia
IBISC – IBGBI Laboratory, University of Evry-Val - d'Essonne, Evry-France
olfa.chabbouh@supcom.tn
Nazim.Agoulmine@ibisc.univ-evry.fr
Sonia.Benrejeb@supcom.tn
z.choukair@supcom.tn

Abstract. With the increase of data traffic in global mobile network, data computing close to the edge is going more and more memorandum to deal with the resources limitations. This paper, addresses Cloud Radio Access Network (C-RAN) architecture and proposes to provide extra computing and storage resources in the edge in order to allow the offloading of a set of mobile users services from the remote cloud computing infrastructure to a cloud computing infrastructure deployed in the edge next to Remote Radio Heads (RRHs). This approach raises many challenges. One of the challenges is the scheduling strategy of the offloading. Therefore, the main contribution described in this paper is a novel cost based service scheduling (CBSS) mechanism which takes into account deployment cost, deadline and available resources in order to make offloading decisions more efficient and to increase user experience. The solution was implemented in a simulator to highlight the benefit of the approach compared to existing approach.

Keywords: Cloud RAN; offloading; Cloud-RRH; task scheduling.

1 Introduction

The evolution toward global mobile networks is characterized by an exponential growth of traffic. It is estimated that the data traffic will grow at a compound annual growth rate of 47 percent from 2016 to 2021 [1]. This growth is mainly due to the huge success of smart phones and tablet. Nowadays, smartphones and tablets are real computers capable to run a large variety of applications in all areas of backend: entertainment, health care, business, social networking, traveling, news… More and more applications are virtualized and running in the cloud overcoming the limited capacities of the end-user devices. However, this necessitates an end to end communication from the mobile terminal to the application or service deployed in the far end cloud computing infrastructure. With the concept of Mobile Cloud Computing (MCC), the idea is to deploy additional cloud computing resources to allow some parts of the applications/services to run locally and offload the communications from the

backend towards this local cloud to save resources and increase end users experience.

More precisely, cloud-based radio access network has been already proposed in the 5G to decouple the Base Band Units (BBUs) from remote radio heads (RRHs) and to move them into the cloud enabling a centralized processing and management. With this approach, traditional complicated base stations can be simplified to cost-effective and power-efficient radio units (RRHs) by centralizing the processing allowing the efficient management of large-scale small-cell systems. Centralized processing power enables indeed more advanced and efficient network coordination and management.

On the other side, mobile data offloading to external extra resources (such as using wifi) is also an important and popular issue in the cellular network. This consists on offloading the data communication from the mobile network access to another wireless access network (wifi, femto, etc) using therefore additional resources. This offloading can also target the processing using alternative storage and processing capabilities close to the end users. Several state of art proposals exploit therefore cloud computing technology for this purpose [2].

Our work is related to this context. We propose a novel Cloud RAN heterogeneous architecture where we introduce an edge cloud: the Cloud-RRH. It consists on additional computational and storage resources added to High RRHs (macro-cells) close to mobile end users. Using this infrastructure, mobile users will be able to offload their applications/services from the far end cloud computing infrastructure close to them in Cloud-RRH. The technology to support this offloading is containers [3] that provides a higher level of abstraction in terms of virtualization and isolation compared to other virtualization techniques. Therefore, in order to fully profit from this architecture we need to efficiently schedule offloading requests among different containers. That's why we propose a cost based task scheduling scheme. Especially we focus on overload and migration costs. Moreover, load balancing between containers has been taken into consideration.

This paper is organized as follows. After this introduction, section II describes the related works. In section III, we present a model of the system, the formulation of the problem, and the basic idea of the proposed solution. The following section IV presents a simulation of the system and the solution as well as initial results. Finally, section V concludes this paper.

2 Related Work

Scheduling user's computing tasks is a hot challenge in cloud computing environment. Optimal resource allocation or offloading request scheduling helps to guarantee application performance and to reduce operating costs. A set of existing works are discussed in this section.

Authors in [4] have proposed a selective algorithm that uses standard deviation to decide between the two scheduling algorithms Min-Min and Max-Min in order to minimize the total execution time of tasks. In [5], the improved Max-Min algorithm is modified to define two new algorithms based on the average execution time. Unlike the Max-Min, the task with a just above average run time is selected and assigned to the

resource that gives a minimum run time. The average run time is calculated using the arithmetic mean for independent tasks and the geometric mean for dependent tasks. The main objective is to reduce tasks makespan. Authors in [6] have proposed a task scheduling algorithm based on priority. They have defined three levels of priorities: the scheduling level which represents the objective to be achieved by the planner, the level of resources which represents the attributes available to achieve the desired goal and the level of tasks which represents the available alternatives among which the best task should be scheduled first. Therefore, each task will require resources with a given priority and the priorities of the different tasks are compared with each other in order to be scheduled. In [7], authors have proposed a task scheduling algorithm based on credits. The proposed approach is based on two parameters: the priority of the user and the duration of the task. A credit is assigned to each task according to its duration and priority. The task with the highest credit value is executed first. In [8], an optimized algorithm for task scheduling based PSO (Particle Swarm Optimization) is proposed. PSO is a population-based search algorithm inspired by bird flocking and fish schooling, where each particle learns from its neighbors and itself during the time it travels in space. However, like any other metaheuristic method, this algorithm does not give any guarantees on finding the most optimal solution. Consequently, whenever the search space expands, the chance of finding an optimal solution becomes harder and harder. Authors in [9] have proposed a cost-deadline based task scheduling algorithm (CBD). The cost is calculated according to the task length, deadline and the number of processing elements required. Then, a sorting mechanism is used to decide the order of execution of tasks. Their mapping with virtual machines is given by the Min-Min heuristic algorithm. Therefore, the proposed approach is used to minimize missed deadlines. Authors in [10] have investigated cost based scheduling using linear programming. They have proposed a task scheduling algorithm based on delay bound constraint (SAH-DB) in order to improve the task execution concurrency: when a task is received all the resources (CPU, memory and network) are sorted in descending order based on the resources processing capacity, then the task is dispatched to resources with the minimum execution time.

In this paper, we propose a novel C-RAN architecture and corresponding resource management mechanism, where a Cloud-RRH is introduced in the edge of the mobile network. While most previous works have focused on jobs' completion time, we propose in this work a scheduling optimization mechanism that aims to reduce the cost of tasks scheduling. Unlike previous works, we model the cost of tasks as function of overloading and migration. The scheduling process takes mainly into account the available resources, resource requirements, deadlines and load balancing in Cloud-RRH.

3 OFFLOADING SCHEDULING MECHANISM PROPOSAL

In this section, we will discuss the considered scenario and problem statement before presenting our system model and formulate the optimization problem for the offloading requests scheduling.

3.1 Scenario and problem statement:

The scenario is depicted in figure 1. We consider a C-RAN heterogeneous architecture composed of H-RRHs (High RRHs) which acts as macro cells and L-RRHs (Low RRHs) which acts as small cells. In our scenario, we introduce the Cloud-RRH which represents cloud capacity in the edge network. While in a traditional C-RAN architecture all the RAN functionalities are centralized in BBU pools, we propose to flexibly split of these functionalities between edge and central cloud. We suppose also that additional computation and storage resources are available in the Cloud-RRH for computation offloading. These resources are represented by cloud containers.

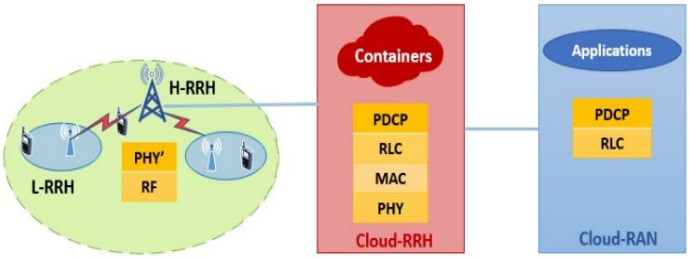


Fig. 1. Proposed C-RAN architecture

We propose to use cloud containers instead of VM because of performance gain. Indeed VM are usually larger than containers since they include the whole operation system and their startup is much slower than containers. A container is essentially a packaged self-contained, ready-to-deploy set of parts of applications, that might even include middleware and business logic in the form of binaries and libraries to run the applications [11], see figure 2. Containers are characterized by: (i) a lightweight portable runtime, (ii) the capability to develop, test and deploy applications to a large number of servers and (iii) the capability to interconnect them.

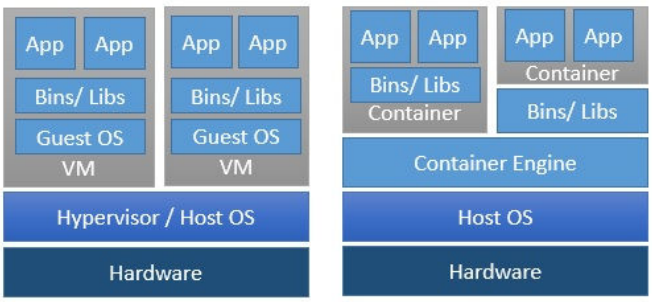


Fig. 2. VM vs Container Virtualization Architecture

In current data centers the control of virtual machines (VM) requires a Virtual Infrastructure Manager (VIM), which is the entity in charge of VM lifecycle management.

In our approach and as part of the cloud management, we propose to add a new functional entity called Cloudlet Manager (CM). The main functionalities of the CM are the following:

- Containers placement / deployment
- Containers monitoring
- Applications scheduling

Mobile users can access their services directly in the edge cloud. The CM could instantiate containers in the edge and offload (part of) the service logic computation in these containers. Containers are not always active, rather they are activated or deactivated accordingly. Different interactions schema are represented in figure 3.

Mobile users' application tasks can be offloaded in the Cloud-RRH to achieve better performances. The cloudlet manager is responsible to decide in which container application tasks will be executed. A container is characterized by a triplet of allocated resources (CPU, RAM, and Network Bandwidth). Each offloading request is considered as a set of tasks to instantiate in the Cloud-RRH. Each task has a delay constraint and resource requirements in terms of CPU, RAM and Network Bandwidth.

However, it is necessary to well design the scheduler of tasks and the offloading decision based on the available resources and the concurrent requests. The research questions that we are trying to respond are the following:

1. How to find the most suitable container for application tasks offloading that minimizes the total cost, comprising overloading cost and migration cost?
2. How to schedule offloading requests while respecting load balancing between containers in Cloud-RRH?

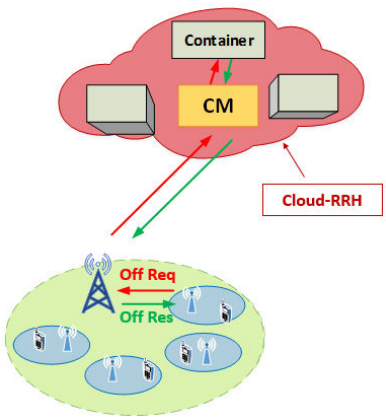


Fig. 3. Cloudlet Manager Interactions

3.2 System model:

We assume that each Cloud-RRH infrastructure is able to run N predefined containers. Each container is characterized by its available capacity resources CPUi, RAMi and Neti, $i \in N$. An offloading request is specified as a set of M tasks to execute with a deadline D. Each task is characterized by its CPUj, RAMj and Netj requirements, and has an expected execution time Tex_j , $j \in M$ (time execution if all resources are satisfied). We consider a binary variable $t_{(i,j)}$ to indicate if a task is allocated to a container or not:

$$t_{(i,j)} = \begin{cases} 1 & if\ task\ j\ is\ allocated\ to\ container\ i \\ 0 & otherwise \end{cases}$$

We associate to each pair container-task allocation a cost C which value depends on whether the container is overloaded after the execution of the task or not and also whether a task migration was necessary due to user mobility. In this work we did not consider the energy consumption cost. The details of the considered costs are presented in the following:

Overload cost.

Let us denote by C_cap_i the computational capacity of container i at time t:

$$C\_cap_i = \begin{pmatrix} C\_cap_i^{CPU} \\ C\_cap_i^{RAM} \\ C\_cap_i^{Net} \end{pmatrix} \quad (1)$$

$C\_ut_{j,i}$ the average resource utilization of task j on container i:

$$C\_ut_{j,i} = \begin{pmatrix} C\_ut_{j,i}^{CPU} \\ C\_ut_{j,i}^{RAM} \\ C\_ut_{j,i}^{Net} \end{pmatrix} \quad (2)$$

The utilization rate μ_i of container i corresponding to the actual system configuration is given by the following formulation:

$$\mu_i = \begin{pmatrix} \mu_i^{CPU} = \dfrac{\sum t_{(i,j)} \cdot C\_ut_{j,i}^{CPU}}{C\_cap_i^{CPU}} \\ \mu_i^{RAM} = \dfrac{\sum t_{(i,j)} \cdot C\_ut_{j,i}^{RAM}}{C\_cap_i^{RAM}} \\ \mu_i^{Net} = \dfrac{\sum t_{(i,j)} \cdot C\_ut_{j,i}^{Net}}{C\_cap_i^{Net}} \end{pmatrix} \quad (3)$$

Therefore, container i is considered as overloaded when $Max(\mu_i^{CPU}, \mu_i^{RAM}, \mu_i^{Net}) > 1$. When a task j is allocated to an overloaded container, we associate a penalty which we also assume to be positively proportional to the level of overloading. We define the overload cost $ov\_\cos t_i$ a metric as follows:

$$ov\_\cos t_i = \begin{cases} (\mu_i - 1)^{\lambda} & if \quad Max(\mu_i^{CPU}, \mu_i^{RAM}, \mu_i^{Net}) > 1 \\ 0 & otherwise \end{cases} \quad (4)$$

Indeed, λ allows to accentuate the overload cost when approaching the saturation. Therefore, the closer we go to the maximum capacity and the more the cost will increase and the choice will go for another container in order to avoid saturation.

The overall overload cost for the Cloud-RRH system to execute all the tasks can be calculated as follows:

$$ov\_\cos t = \sum_{i=1}^{N} ov\_\cos t_i \quad \text{, N: set of containers} \quad (5)$$

Migration cost.

When a mobile user is moving from one cell to another one, the corresponding tasks may be migrated. We associate a penalty r_j when a user task j is migrated, from one container to another one, to capture the service downtime incurred by the migration. The overall migration cost is defined as:

$$mig\_\cos t = \sum_i \sum_j t_{(i,j)} \cdot r_j \quad (6)$$

In this paper, we only consider a migration of the tasks in the same Cloud-RRH and that migration penalty only depends on the type of task. (Migration in the whole network will be considered in future works)

Intuitively the two variables, overload cost and migration cost, are correlated. For example, if we completely optimize the overload cost, task will be distributed over all

available containers which will increase the migration cost. Therefore, we need to get a trade-off between the two variables.

Optimization model.

Therefore, the goal of the scheduler is to minimize the total cost of overloading and migration in the entire system when executing all the submitted requests. We considered two parameters α and β that represents the importance of weight given to each cost.

Objective function

$$Minimize\ \ Cost = \sum \alpha\ ov\_\cos t + \beta\ mig\_\cos t \tag{7}$$

Subject to

$$\sum_j t_{(i,j)} \cdot Tex_j \le D \tag{8}$$

$$\begin{cases} \sum_j t_{(i,j)} \cdot C\_ut_{j,i}^{CPU} \le CPU_i \\ \sum_j t_{(i,j)} \cdot C\_ut_{j,i}^{RAM} \le RAM_i \\ \sum_j t_{(i,j)} \cdot C\_ut_{j,i}^{Net} \le Net_i \end{cases} \tag{9}$$

$$\frac{\left| \mu_i - \frac{\sum_i \mu_i}{N} \right|}{\mu_i} \le \varepsilon \tag{10}$$

$$\sum_j t_{(i,j)} = 1 \tag{11}$$

The optimization is subject to constraints given by (8) through (11). Constraint (8) guarantees that each offloading request is executed before the application's deadline. Constraint (9) enforces that all tasks' requirements including number of CPU, amount of memory and network bandwidth are lower than container resources. Constraint (10) guarantees load balancing between containers in the same Cloud-RRH where ε denotes for the maximum tolerance of load balancing. Finally, constraint (11) ensures that each task is scheduled on only one container.

First we set $\alpha = \beta = 0.5$ which means that equal weight is given for the different types of resources. We also consider that all tasks are executed in parallel and the deadline D constraint is therefore fixed for the worst case when all tasks are executed in serial.

This problem is a MIP and can therefore be solved as a linear program since the objective function is linear to all variables.

4 SIMULATION AND RESULTS

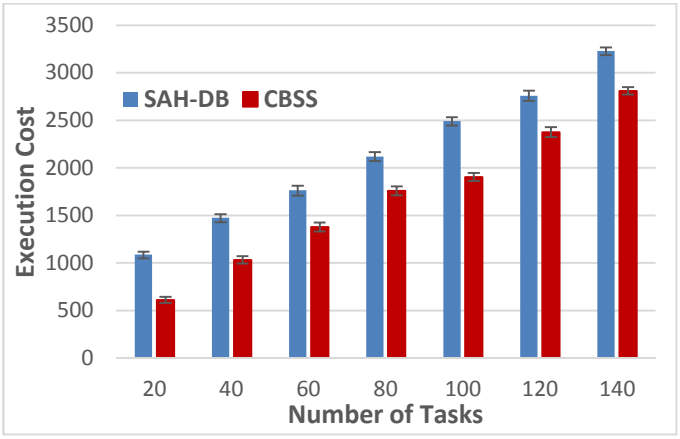
In order to evaluate the scheduling performance on tasks' execution cost for the proposed cost based scheduling scheme (CBSS), we have compared its results with SAH-DB scheduling mechanism. SAH-DB is a task scheduling algorithm based linear programming. It aims to schedule tasks while reducing the total execution cost within the user-expected delay bound. When a task t is utilizing a resource k, the execution cost is expressed as the cost of the resource k executing the task t.

We considered a Cloud-RRH with N= {25, 50, 75, 100} containers having heterogeneous resources. The computing capacity of containers varies from 1 to 10 CPUs. The memory is set from 128 Mbytes to 512 Mbytes and the network bandwidth is set from 100 Kbps to 200 Kbps. The number of tasks is set as M= {20, 40, 60, 80, 100, 120, 140}. Tasks have heterogeneous requirements: CPU varies from 1 to 4, memory is between 128 and 1024 Kbytes and network bandwidth is varying between 1 and 20 Kbps. Offloading requests are embedded sequentially and their requirements are generated randomly. Simulation parameters are summarized in Table I. As we have mentioned before, we set $\alpha = \beta = 0.5$ and $\lambda=2$.

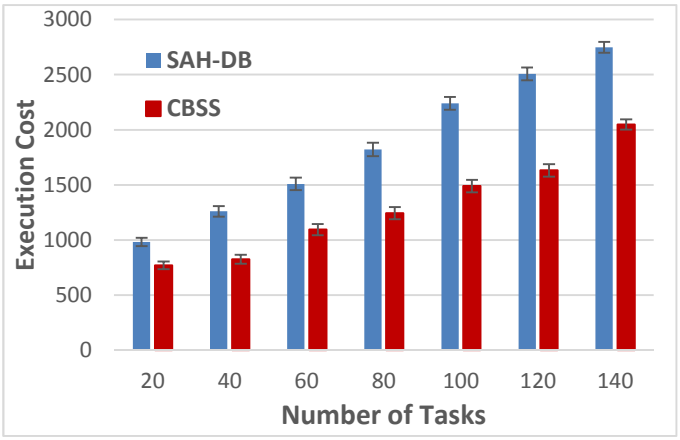
Table 1. PARAMETERS SETTING

Entity	Parameter	Value
Container	Number of containers	25 - 100
	CPU	1 - 10
	Memory (RAM)	128 - 512 Mbytes
	Network Bandwidth	100 – 200 Kbps
Task	Total number of tasks	20 - 140
	CPU	1 - 4
	Memory (RAM)	128 – 1024 Kbytes
	Network Bandwidth	1 – 20 Kbps

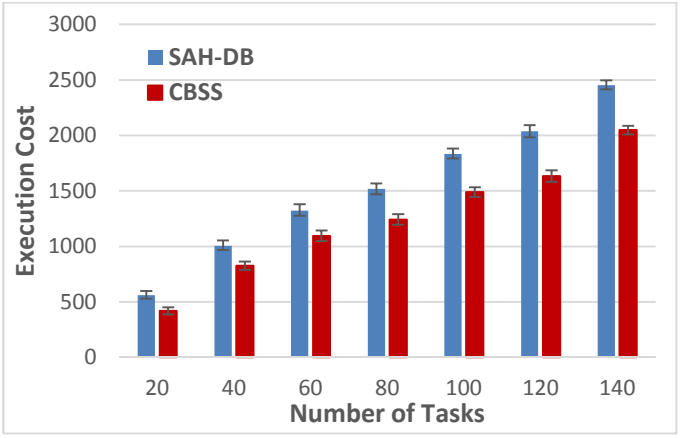
We have used IBM's linear programming solver CPLEX [12], and solved the problem with multiple data inputs.



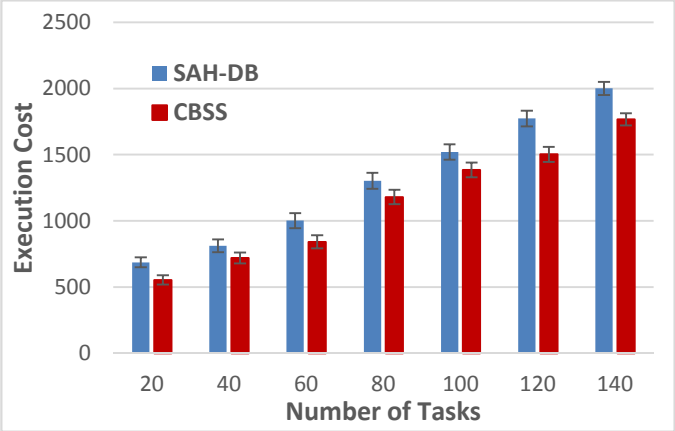
(a) Resources = 25



(b) Resources = 50



(c) Resources = 75



(d) Resources = 100

Fig. 4. The execution cost with different resources

We have evaluated the scheduling efficiency in terms of execution cost under a varying number of associated tasks. Figure 4 represents the execution cost and its standard deviation by applying the proposed cost based scheduling scheme and SAH-DB scheduling algorithm with 25 to 100 cloud containers respectively. The proposed scheduling algorithm can reduce total execution cost compared with SAH-DB algorithm in the different number of associated tasks. Moreover, the total scheduling cost decreases with the increase of the number of resources and increases with the number of associated tasks. Therefore, the more resources are available the more the scheduling process is efficient.

5 CONCLUSION AND PERSPECTIVES

This paper proposes a cost based scheduling scheme (CBSS) that aims to minimize scheduling cost while considering available resources, resource requirements, deadline and load balancing in Cloud-RRH. We consider a scenario where users can offload tasks to Cloud-RRH. We focus on scheduling tasks that requests several resources such as CPU, memory and disk. We formulate the problem as cost optimization problem which takes into account user performance in terms of system overload and migration cost. Simulation results show that the proposed scheme is able to schedule offloading requests while minimizing the total execution cost.

As future works, we will try consider mobility between different Cloud-RRHs while scheduling offloading request. . Furthermore, we will try to better investigate and evaluate the network performances by handling the interference and mobility management in C-RAN.

REFERENCES

1. Cisco, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper,” *Cisco*. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. [Accessed: 29-Apr-2017].
2. K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, “A Survey of Computation Offloading for Mobile Systems,” *Mob. Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.
3. R. Dua, A. R. Raja, and D. Kakadia, “Virtualization vs Containerization to Support PaaS,” in *2014 IEEE International Conference on Cloud Engineering*, 2014, pp. 610–614.
4. M. Katyal and A. Mishra, “Application of Selective Algorithm for Effective Resource Provisioning in Cloud Computing Environment,” *Int. J. Cloud Comput. Serv. Archit.*, vol. 4, no. 1, pp. 1–10, Feb. 2014.
5. S. B and D. M. D.h, “An Improved Task Scheduling Algorithm based on Max-min for Cloud Computing,” *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 2, no. 2, pp. 84–88, Apr. 2015.
6. S. J. Patel and U. R. Bhoi, “Improved Priority Based Job Scheduling Algorithm in Cloud Computing Using Iterative Method,” in *2014 Fourth International Conference on Advances in Computing and Communications*, 2014, pp. 199–202.
7. A. Thomas, G. Krishnalal, and V. P. Jagathy Raj, “Credit Based Scheduling Algorithm in Cloud Computing Environment,” *Procedia Comput. Sci.*, vol. 46, pp. 913–920, Jan. 2015.
8. A. Khalili and S. M. Babamir, “Makespan improvement of PSO-based dynamic scheduling in cloud environment,” in *2015 23rd Iranian Conference on Electrical Engineering*, 2015, pp. 613–618.
9. T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, “A Cooperative Scheduling Scheme of Local Cloud and Internet Cloud for Delay-Aware Mobile Cloud Computing,” in *2015 IEEE Globecom Workshops (GC Wkshps)*, 2015, pp. 1–6.
10. M. Yingchi, X. Ziyang, P. Ping, and W. Longbao, “Delay-Aware Associate Tasks Scheduling in the Cloud Computing,” in *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*, 2015, pp. 104–109.
11. S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, “Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors,” in *Proceedings of the 2Nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, New York, NY, USA, 2007, pp. 275–287.
12. “IBM ILOG CPLEX Optimization Studio,” 01-Jan-2016. [Online]. Available: https://www.ibm.com/software/products/fr/ibmilogcpleoptistud. [Accessed: 29-Apr-2017].