This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2899679, IEEE Transactions on Industrial Informatics

A Hybrid Computing Solution and Resource Scheduling Strategy for Edge Computing in Smart Manufacturing

Xiaomin Li, Jiafu Wan, Hong-Ning Dai, Muhammad Imran, Min Xia, Antonio Celesti

***Abstract*—At present, smart manufacturing computing framework has faced many challenges such as the lack of an effective framework of fusing computing historical heritages and resource scheduling strategy to guarantee the low latency requirement. In this paper, we propose a hybrid computing framework and design an intelligent resource scheduling strategy to fulfill the real-time requirement in smart manufacturing with edge computing support. First, a four-layer computing system in a smart manufacturing environment is provided to support the artificial intelligence (AI) task operation with the network perspective. Then, a two-phase algorithm for scheduling the computing resources in the edge layer is designed based on greedy and threshold strategies with latency constraints. Finally, a prototype platform was developed. We conducted experiments on the prototype to evaluate the performance of the proposed framework with a comparison of the traditionally-used methods. The proposed strategies have demonstrated the excellent real-time, satisfaction degree and energy consumption performance of computing services in smart manufacturing with edge computing.**

Index Terms—**Industry 4.0, Smart Manufacturing, Edge Computing, Resource Scheduling**

I. Introduction

Recently, thanks for the great progress of information and communication technologies in manufacturing domains, Industrial Internet of Things (IIoT), Cyber-Physical System (CPS) and other smart frameworks and systems have been constructed and implemented to increase the flexibility and enhance economic efficiency [1-5]. In this scenario, an increasing attention from enterprises and academia has been devoted to inventing and extending new computing technology or framework, e.g., Industry 4.0, smart factory and intelligent manufacturing [6-10]. However, traditional manufacturing systems that are lacking of efficiency, in terms of a slow computing speed on complicated task, are not suitable for complicated manufacturing process and dealing with big data analysis, especially for AI task. Hence, an extended framework on top of traditional manufacturing systems by introducing computing resources with different levels of computing capabilities, like cloud and edge computing platforms, can meet fundamental requirement for resolving the aforementioned problems. However, cloud computing platforms generally are far from the industrial devices; it consequently increases the latency, leads to the lag in data transmission, and cannot guarantee the real-time performance.

Real-time feature of data flow of the whole industrial system, directly affecting the production efficiency and normal operation of the system, plays a critical role in a smart factory and Industry 4.0 [11]. Therefore, reducing the overhead of data processing and transmission in computing-extensive tasks (e.g., AI and deep learning tasks) is another aspect to guarantee the real-time performance. The current latency-constraints methods concentrate on network optimizations, data fusion, computing-task simplification, and computing resource decentralization. Edge/fog computing frameworks, close to producing equipment consequently leading to the decrement of the latency for data communications between servers and machines, are good candidate strategies for smart manufacturing to tackle the above problems [12]. In pioneering works [13]-[16], the authors presented the edge computing frameworks/strategies. Furthermore, the studies [17, 18] point out that computing-resource scheduling is an important factor to achieve the real-time performance and realize the fusion among of edge computing, AI and smart manufacturing in a manufacturing environment. Therefore, strategies of scheduling computing resources are a significant perspective for ensuring real-time AI task. The paper will focus on optimizing the solution to this problem and improve the efficiency of resources in a real-time manner. In particular, we will develop a better strategy to improving the efficacy of resources in a real-time manner, reducing the power consumption, and enhancing the reliability of the whole system.

Recent research achievements, such as the fog-assisted manufacturing system (CAMS) [19], and the edge (fog) computing producing system (ECPS) [20] have provided a preliminary basis to enable the design of resource scheduling for smart manufacturing and Industry 4.0. Nevertheless, we should be aware that computing- resource scheduling to ensure the real-time requirement in the context of Industry 4.0 still faces many challenges. These challenges can be summarized as two fundamental problems: 1) how to propose a computing system to handle

Manuscript received November 30, 2018; revised February January 22, 2019; accepted February 11, 2019. Date of publication xx, 2019; date of current version xx, 2019. The corresponding author: Jiafu Wan

X. Li is with the School of Mechanical Engineering, Zhongkai University of Agriculture and Engineering, Guangzhou, China (e-mail: lixiaomin@zhku.edu.cn).

J. Wan is with the School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China (e-mail: mejwan@scut.edu.cn).

H. Dai is with the College of Information Technology, Macau University of Science and Technology, Macau, China (e-mail: hndai@ieee.org).

M. Imran is with the College of Computer and Information Sciences, King Saud University, Saudi Arabia (e-mail: dr.m.imran@ieee.org).

M. Xia is with the Department of Mechanical Engineering, University of British Columbia, Vancouver, Canada. (e-mail: minxia@mech.ubc.ca)

A. Celesti is with the MIFT Department, University of Messina, Messina, Italy. (e-mail: acelesti@unime.it)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2899679, IEEE Transactions on Industrial Informatics

and integrate the historical heritage of computing resources; 2) how to construct some novel and efficient strategies and algorithms to ensure the real-time performance.

This article explores resource scheduling strategy for manufacturing in edge computing to provide real-time computing services, from the perspective of the implementation of Industry 4.0. In summary, there are three main contributions of this paper:

- From the AI task operation perspective, a four-level computing system architecture for smart manufacturing is designed for industrial environments, which contributes to integration and fully utilization of different computing resources.
- A two-phase scheduling strategy for the computing resources strategy in edge computing is provided to meet the performance requirement of different AI tasks with consideration of low latency constraints.
- The proposed scheduling method and the traditional algorithms are compared. The provided algorithm is implemented in a smart manufacturing prototype platform to validate its feasibility and effectiveness.

The remainder of this paper is organized as follows. Section II introduces related work about edge computing, real-time and computing resources scheduling in manufacturing. Section III gives the four-level architecture and the working process for manufacturing computing. Section IV proposes the methods used for computing resources scheduling for edge servers. In Section V, the experiments for the proposed algorithms are undertaken. Section VI concludes the paper.

II. Related Work

The effective of computing resources control is a necessity to guarantee the low latency and continuous production in smart factories, consequently improving production efficiency and bringing economic benefits. Therefore, in this section, we briefly outline existing efforts in aspects of edge computing, real-time methods and resources scheduling strategies in manufacturing.

A. Edge/Fog Computing of Manufacturing

In [21], the authors proposed an architecture of edge computing for IoT-based manufacturing and analyze the function of edge computing in a manufacturing system. In [22], a manufacture inspection system for the smart industry was designed, while it adopted the deep learning models to find out the defects based on fog computing. In [23], the authors proposed a cyber-physical machine tool system based on fog computing-based. Meanwhile, the study gave the definitions and functions for computer numerical control (CNC) machines. In [24], the authors divided the data flows into ordinary and emergent streams according to different latency constraints, then adopted the edge computing the adaptive transmission strategies. In [25], a multi-tier multi-access edge computing (mMEC) framework was provided and its role in the Industry 4.0 was investigated for manufacturing computing performance. In [26], an edge device capable of collecting, processing, storing and analyzing data is constructed by using a single-board computer and a sensor. All these studies contribute to edge/fog computing in manufacturing. However, these literatures cannot consider the resource unbalance and differences among edge computing servers.

B. Real-time Schemes in Manufacturing

In [27], the authors discussed the importance of real-time in the industrial system and pointed out that real-time is the most significant evaluation indicator for industrial automation applications. In [28], the wireless transmission characteristics of wireless networks were obtained and analyzed, then according to these characteristics, a real-time big data gathering (RTBDG) algorithm for wireless networks is proposed for industrial operations. In [29], an industrial cyber-physical system based on the emerging fog computing paradigm was provided. In the system, machine learning models can be installed to support factory operation. In [30], authors propose an innovative multi-microprogrammed control unit (MCU) system framework combining a field-programmable-gate-array-based hardware bridge and multiple scalable MCUs to realize an edge computing gateway to get low latency performance of in industrial IoT. In [31], a ship inspection system based on fog computing was introduced. The system offers identifying and tracking of the pipe tasks, consequently decreasing latency. These literatures mostly focused on strategies of industry networks or adopted edge computing to ensuring real-time performance. However, there are few papers investigating resources scheduling to support real-time in intelligent manufacturing.

C. Resource Scheduling Strategies for Edge Computing

In [32], a novel model for allocating computing resources in an edge computing platform was proposed to allow service providers to establish resource sharing contracts with edge infrastructure. In [33], an intelligent agent at the edge computing was designed to develop a real-time adaptive policy for computational resource allocation for offloaded tasks of multiple users in order to improve the system reliability. In [34], the authors formulated the computation offloading problem for mobile edge computing into the system cost minimization problem and present a distributed algorithm consisting of offloading strategy selection by taking into account the completion time and energy. In [35], the authors design a new optimization framework by using an extended Lyapunov technique. In [36], a resource allocation strategy for fog computing based on priced timed Petri nets (PTPNs), by in which the user can choose the satisfying resources autonomously. These studies above-mentioned were not suitable for manufacturing, as the real-time constraint was not considered in these algorithms. Moreover, the literature seldom solves the problem by considering the cooperation of multiple computing servers.

III. System Architecture

This section presents the hybrid computing architecture in manufacturing and the working process of manufacturing computing.

A. Hybrid Computing Architecture in Manufacturing

In the traditional framework, there are two layers to complete the computing tasks: fog/edge and cloud. All the tasks are transmitted into a cloud or edge, on top of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2899679, IEEE Transactions on Industrial Informatics

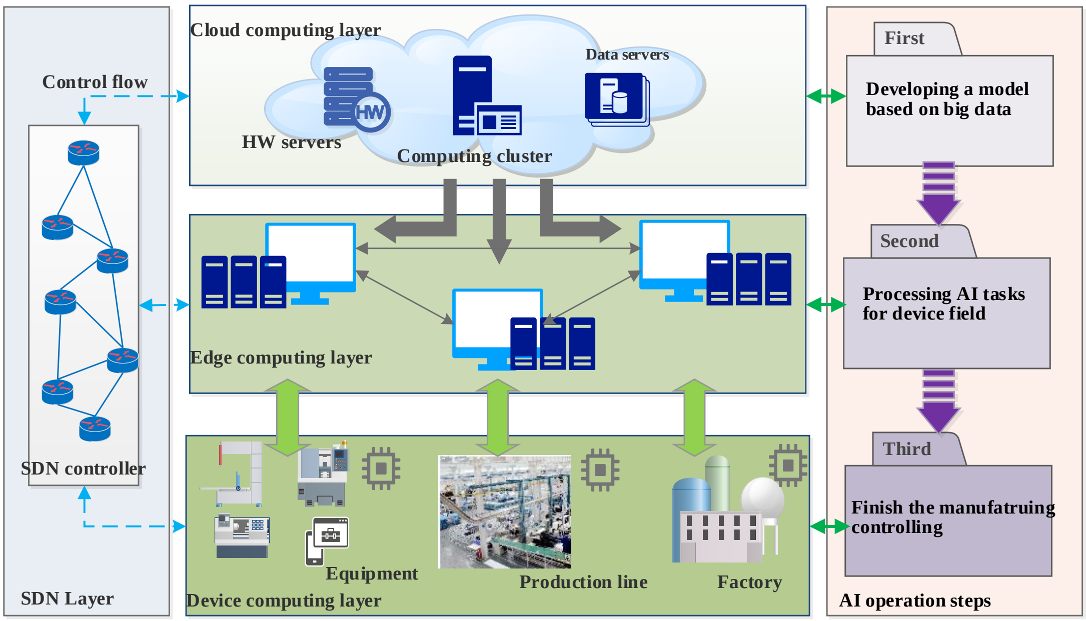


Figure 1. Architecture for hybrid computing system

traditional environment, the main drawback of this architecture lies in failing to fulfill the real-time requirement, especially many tasks queuing in edge servers (ES). After introducing smart networks nodes, agent devices with limited computing capability, cloud-computing servers and fog-computer servers, the traditional intelligent manufacturing system can be transferred to the hybrid computing architecture. Figure 1 shows the system architecture for manufacturing computing using different computing resources for the computational task.

Obviously, cloud servers have the strengths of data storage and computing power; edge servers are close to the industrial devices and equipment, thereby having benefits of real-time performance; device computing units can directly drive the mechanical structure; Software defined networking (SDN) can simply provide the cooperation of different network devices. Hence, from the network perspective, all computing resources are integrated into the hybrid computing architecture to meet the latency requirement. This hybrid architecture essentially contains four parts from the task-node perspective (such as manufacturing devices): 1) Device computing layer, 2) Edge computing layer, 3) Cloud server, and 4) SDN layer. All these elements are collected by the industrial networks (i.e., wired/wireless network). In the cloud layer, the servers are mainly used to resolve the computing-extensive tasks, in which AI model are developed based on different information and big data. In edge computing layer, the edge computing servers are explored to finish the real-time, AI works. Additionally, in the devices computing layer, the devices are mainly responsible for finishing the sensing and controlling works. Besides, the SDN layer is used to control and coordinate different computing layer. There are differences between traditional maintenance and hybrid manufacturing computing architecture.

B. Working Process of Manufacturing Computing

In this paper, we mainly focus on computing resource allocation in the proposed framework. The working process of manufacturing computing is briefly introduced. For the hybrid computing system, all computing tasks are created on the field devices, including producing machines, wireless network nodes and mobile elements. Tasks are random events which should usually be processed in real-time manner. For scheduling a task, there are three factors to be considered: compute capability, queueing time and data communication latency. While according to the three factors, the latency of the task during the special time window can be computed, then in the hybrid computing system, in accordance with the real-time requirement, the AI task can be located at different computing layers.

For edge computing layer, the computing capability and queueing time are the significant factors to determine the task completion time. It is obvious that there are differences for one edge computing layer from computing power, storage power. Since the different servers may deal the task with different complexity, they may have different values of queueing time.

Actually, the historical legacy of computing resources and the low system latency are considered in the presented architecture. In particular, the computing framework can integrate the different level computing resources in a smart factory, such as device computing, edge servers and cloud servers. Therefore, our framework has outstanding performance in term of low latency with the comprehensive utilization of various computing resources, especially for device layer.

IV. Resources Scheduling in Edge Layer

This section mainly describes the resource scheduling in the edge layer. We first give the architecture of the edge layer, then present an algorithm for selecting edge computing server and a cooperation strategy for multiple edge servers.

A. Architectures of the Edge Layer

After the above analysis, the manufacturing edge layer (MEL), cloud, device computing resources of devices (LCRD) are constructed and connected to the manufacturing computing system via SDN wired /wireless networks. However, cloud servers are typically far from the devices and the system has to spend more time in transmitting the task data between devices and the cloud server. Meanwhile, LCRD is limited by computing capability and responsible for dealing with the necessary tasks with the supporting of the local system normal operation. MEL that is close to the manufacturing equipment, plays the most important role in processing the real-time tasks. As shown in Figure 2, the MEL consists of multiple edge server clusters (ESCs). Every edge server (ES) is heterogeneous in the capacity of computing, storage and task loads. In MEL, the ESs are connected via the high-bandwidth networks, such as wired links, optical fiber, etc. Therefore, ESs can form an edge server cluster network with low delay. Therefore, every ES is deployed collocating with the devices to fulfill real-time computing tasks.

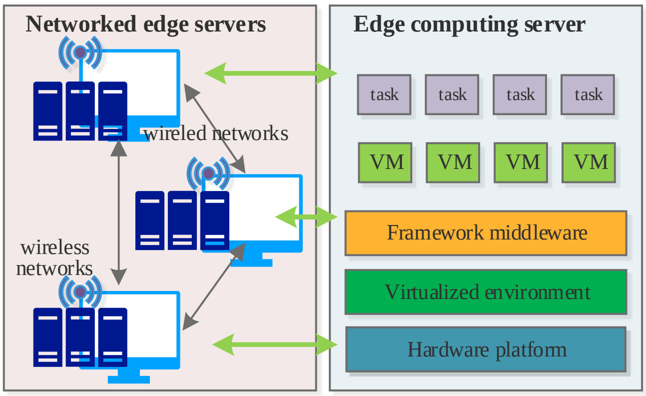


Figure 2. Mechanism of tasks scheduling for edge computing layer

In order to achieve an efficient task process, ESs are placed in approximation to the devices. The tasks randomly are generated by the manufacturing equipment. They are then arranged and transmitted to the near and suitable ESs to ensure the real-time constraint. Obviously, there are two cases: 1) Single ES can be qualified for the task; 2) Single ES cannot be qualified. Therefore, there are two strategies for computing resources scheduling: Selecting algorithms for ES (SAE) and cooperation of edge computing cluster (CEC) to fulfill real-time requirement. The former way can be used to meet the low real-time requirement of computing tasks. SAE scheduling algorithm undertakes to choose the suitable ES from the edge server set (ESS) according to the task load, communication time and computing power. Moreover, CEC is adopted for low latency requirement, in which one ES cannot qualify to guarantee the low latency.

B. Algorithm for Selecting Edge Computing Server

MEL has the direct impact on the computing performances of the manufacturing task. It is indispensable to propose the scheduling algorithms for MEL and ESS. Scheduling for MEL contains two aspects: Selecting algorithms for ES (SAE) and cooperation of edge computing for low latency task (CEC). Based on the requirement of specific application for manufacturing, the latency requirement of getting computing results depends on communication, computing and queuing time. In particular, the task x processing time in single edge server es T_{task} can be formulated as (1)

$$T_{task}(x,esc) = T_{trans}(x,es) + T_{que}(x,es) + T_{process}(x,es) + T_{re}(x,es) \tag{1}$$

where T_{trans}, T_{que}, $T_{process}$, T_{re} are the times of transmitting task to edge server, queuing, processing and receiving, respectively. Furthermore, assume that the data size of task and results are $\wp$, $\Im$ and the data rate is v, so T_{trans}, T_{re} can be described as (2), (3):

$$T_{trans}(x,es) = k_{x,esc}\frac{\wp(x)}{v(es)} \tag{2}$$

$$T_{re}(x,es) = k_{x,esc}\frac{\Im(x)}{v(es)}. \tag{3}$$

Meanwhile, let X be set of tasks in edge server, namely $\mathrm{X} = \{x_1, x_2, \cdots, x_{|\mathrm{X}|}\}$. The set of computer instructions is denoted by $XN = \{xn_1, xn_2, \cdots, xn_{|\mathrm{X}|}\}$, which is used to deal with X. According to the [23], for the new tasks, the queuing time can be formulated as (4)

$$T_{que}(x,es) = \sum_{i=1}^{|\mathrm{X}|}\sum_{j=1}^{|xn_i|}\frac{IN_j}{V_{process}} \tag{4}$$

where IN_j and $V_{process}$ are the j-th instruction of the i-th task and the process speed of the edge server, respectively. In a similar way, we can get the equation for processing this task as follows,

$$T_{process}(x,es) = \sum_{j=1}^{|x|}\frac{IN_x}{V_{process}}. \tag{5}$$

According to formulations (1) to (5), the processing time of task x demoted by T_{task} can also be formulated as

$$T_{task}(x,es) = k_{x,esc}\frac{\wp + \Im}{v} + \sum_{i=1}^{|\mathrm{X}|}\frac{IN \cdot xn_i}{V_{process}} + \frac{IN \cdot xn}{V_{process}}. \tag{6}$$

Algorithm1: pseudocode of Selecting single ES

Initialization: Input task x, xn, v, v_{process}, X, XN, E, $t_{requrement}$, $Es = \varnothing$

Begin: $Es \leftarrow E$
for $i \leftarrow 1$ to $|E|$
  $T_{com}(e_i) \leftarrow k_{e_i} \cdot \frac{\wp + \Im}{v}$
  // computing the communication time;
  $t_{process} \leftarrow \frac{IN \cdot xn}{V_{process}}$
  // computing the process time of task x;
  for $j \leftarrow 1$ to $|X_{e_i}|$
    // computing the queuing and process time;
    $t_{que}(ij) \leftarrow \frac{xn_{ij} IN}{V_{process}}$
    if ($t_{que}(ij) \geq t_{requrement}$)
      $Es \leftarrow Es \setminus e_i$
      //Selecting ES according with t_{que};
    else // Selecting ES with the total time of task x;
      for $f \leftarrow 1$ to $|Es|$
        //computing the total time
        $T_{task}(x, e_f) \leftarrow T_{com}(e_f) + T_{que}(e_f) + T_{process}(e_f)$
        if ($T_{task}(x, e_f) > t_{requrement}$)
          $Es \leftarrow Es \setminus e_f$
          //Selecting ES according with t_{que};
        else
          Break;
      End for
    End if
  End for
End for
Return Es

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2899679, IEEE Transactions on Industrial Informatics

To ensure the real-time requirement of processing task *x*, the edge computing server must be subject to the flowing inequation (7):

$$T_{task}(x,es) \le T_{req}(x). \tag{7}$$

Assume that there are multiple edge computing servers close to the device which contains the task *x*. For easily understanding, let *E* be the set of ESs, namely $E = \{e_1, e_2, \cdots e_{|E|}\}$.

So, we propose Algorithm 1 to fulfill the strategy of SAE. In this algorithm, the selecting a single ES strategy can mainly divide into three steps. Firstly, the system searches the all ESs, and construct the set of *E*. Then, according to the equation of (2)-(4), we can get the communication time T_{com} and queuing time $T_{process}$ of every ES in the set of *E*. Thirdly, we evaluate the queuing time to determine whether it is larger than the deadline time of task *x*. Moreover, we update the candidate set *Es* of ES to processing the task. Fourthly, in the light of the total time for resolving the task, we update *Es*. Finally, the device randomly selects the ES from *Es* for the task *x*.

C. Cooperation Strategy of Networked Edge Computing to Achieve Low Latency

In Section 4-B, the low real-time requirement of processing algorithm is given. It is obvious that Algorithm 1 may not deal with the computing-extensive task as there is only one ES assigned for this task. Therefore, we propose a method to cooperate multiple edge computing servers to create ESC to fulfill the latency constraints of single ES.

Algorithm2: pseudocode of CES

Initialization
for $k \leftarrow 1$ to $|E|$
  $IN(e_k) \leftarrow (T_{req}(x) - T_{com}(ec_0, device_x) - T_{que}(e_k))V_{process}(e_k)$
  //getting the subtask instruction number in the constraints of $T_{req}(x)$
  $IN(E) \leftarrow IN(e_k)$
  $Es' \leftarrow sort(IN(E))$
  //*sort* () is the function for sorting the E according with the $IN(e_k)$;
  $main\_ESC \leftarrow Max(IN(E))$
  //selecting the main ES
End for
for $i \leftarrow 1$ to $|ES'|$
  Temp_sum= $IN(es_i^{'})$ +*Temp_sum*
  if ($Temp\_sum < xn$)
    $Es \leftarrow Es / es_i^{'}$
  else
    break;
  End if
End for
divided_task (***x***, *Es*) // divide the task x according with Es
processing_subtask () // processing subtask in selecting edge server
Return_subtask_result ()
//returning the subtask result from different edge server
$RES \leftarrow merge\_subresult$ ()
//the main edge server merging the results
Return *RES*

Indeed, once the edge servers are placed into smart factory, they are connected via industrial networks. Then, in the industrial system, the edge servers are clustered with cloud servers via SND controllers according to network distance between edge servers and cloud servers. Hence, to achieve low latency, the latter is adopted in the novel framework. The main idea of the method is explained as follows: 1) Selecting an edge server as the main server for dividing task and merging the results; 2) Choosing other edge servers to cooperate to finish the task according to the latency.

Assume that the task *x* is be divided into N ($1 \le N \le |E|$) subtasks, which are executed in parallel at an ESC to ensure the real-time demands. We denote the set of sub-task by $x = \{sx_0, sx_1, sx_2, \cdots, sx_{N-1}\}$.

Let $Ec = \{ec_0, ec_1, ec_2, \cdots, ec_{N-1}\}$ be the set for cooperating to process the task *x*. For the subtask $sx_i \in x$ $(0 \le i \le N-1)$, the communication time can be given as (8)

$$T_{com}(ec_0, ec_i) = \begin{cases} \dfrac{D_{rough}(sx_i) + D_{result}(sx_i)}{V_{ec_0, ec_i}} & if\ i \ne 0 \\ 0 & otherwise \end{cases} \tag{8}$$

where $D_{rough}(sx_i)$ and $D_{result}(sx_i)$ are subtask rough and data size of results, respectively. The term of V_{ec_0,ec_i} is the average data rate between ec_0 and ec_i. Furthermore, in light of the literature [23], we can get the subtask processing time in (9)

$$T_{process}(sx_i, ec_i) = \frac{IN_{sx_i}}{V_{process}(ec_i)} \tag{9}$$

where IN_{sx_i} is the subtask instruction number, and $V_{process}(ec_i)$ is the processing speed of the *i*-th ES of *Ec*. It is worth mentioning that $|Ec| = N$, while we can get the formulation of $T_{sub\_task}(sx_i, ec_i)$ $(0 \le i \le N-1)$ as shown in equation (10), according to formulations (1) and (6):

$$\begin{aligned} T_{sub\_task}(sx_i, ec_i) &= T_{com}(ec_0, ec_i) + T_{que}(sx_i, ec_i) \\ &+ T_{process}(sx_i, ec_i) \end{aligned} \tag{10}$$

where $T_{com}(ec_0, ec_i)$ is the communication time between ec_0 and ec_i, $T_{que}(sx_i, ec_i)$ and $T_{process}(sx_i, ec_i)$ are queuing time and the process time for subtask sx_i in edge computing server ec_i, respectively.

Recall the fact that the main ES is responsible for dividing task and merging the results. Therefore, the running task time in main ES is formulated as (11)

$$\begin{aligned} T_{main}(x, sx_0, Ec) &= T_{divide}(x) + \sum_{i=1}^{N-1} T_{com}(ec_0, ec_i) \\ &+ Max(T_{sub\_task}(sx_i, ec_i)) + T_{merge}(x, Ec) \end{aligned} \tag{11}$$

where $T_{divide}(x)$ and $T_{merge}(x, Ec)$ are the dividing time and the merging-result time for task *x* in edge computing servers set *Es*, respectively. Therefore, the total time for task *x* running at *Ec* is decided by the communication time between main edge computing server and the device of task *x* and the running time $T_{main}(x, sx_0, Es)$ (as given in equation (11)). It is formulated as (12):

$$T_{task}(x, Es) = T_{main}(x, sx_0, Ec) + T_{com}(ec_0, device_x) \tag{12}$$

It is worth noting that T_{divide}, T_{merge}, $T_{com} \ll T_{process}$, so

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2899679, IEEE Transactions on Industrial Informatics

according to equation of (11), the equation can be simplified into (13):

$$T_{task}(x,Es) = Max(T_{sub\_task}(sx_i, ec_i)) + T_{com}(ec_0, device_x) \quad (13)$$

Equation (7) gives the time constraints for processing the task. Therefore, we can get the inequation (14):

$$\begin{aligned} &Max(T_{sub\_task}(sx_i, ec_i)) + T_{com}(ec_0, device_x) \le T_{req}(x) \\ &Max(T_{sub\_task}(sx_i, ec_i)) \le T_{req}(x) - T_{com}(ec_0, device_x) \quad . \\ &T_{sub\_task}(sx_i, ec_i) \le T_{req}(x) - T_{com}(ec_0, device_x) \end{aligned} \quad (14)$$

According to the (9) and (14), it is easy to get the task instruction number of the i-th ES. It is described as (15)

$$IN_{sx_i} \le (T_{req}(x) - T_{com}(ec_0, device_x) - T_{que}(sx_i, ec_i))V_{process}(ec_i). \quad (15)$$

Furthermore, task time $T_{task}(x,Es)$ is determined by the maximum $T_{process}$. Therefore, in light of the above discussion, we propose the strategy of cooperating edge computing servers for the extensive task as shown in Algorithm 2.

In Algorithm 2, the steps can be mainly described as followings: Firstly, according to the referenced equations, we compute subtask instruction number $IN(E)$ in the constraints of $T_{req}(x)$ for every ES in the set of E. Then, we sort $IN(E)$ in the descending order (*i.e.*, from largest to smallest), and create the sorted ES set *Es'*. Thirdly, we sum the subtask instruction number, *temp_sum* and evaluate whether the *temp_sum* meets the requirement of task x. Finally, the main ES divides the task x and finishes the processing task x by *Es*, and returns the result of the task (*RES*).

V. Analysis and Experiment

This section mainly evaluates the performance of the proposed framework based on an implementation of a prototype in a realistic manufacturing platform.

A. Prototyping Platform and Experiment Installment

For analyzing our proposed strategies of scheduling edge computing resources, a prototyping platform with edge computing servers and industrial internet of things is constructed. As demonstrated in Fig.3, the prototyping platform is composed of four parts: device field, edge computing layer, industrial private cloud servers, and industrial networks (with SDN).

The device field contains multiple types of equipment mobile robots, products processing machines, conveyor belts, and manipulators. To construct the edge computing layer, multiple single-board computers connected with wireless/wired local area network (LANs) and Bluetooth (*i.e.*, Raspberry Pi 3 Model B) are adopted to sever as edge servers. Each single-board computer is equipped with a quad-core 1.2GHz Broadcom BCM2837 64bit central processing unit (CPU) and 1GB random-access memory (RAM). For linking with other edge servers, we exploit the Ethernet with 100Mbps bandwidth and adopt servers switch hubs. In addition, wireless communications (Wi-Fi or Bluetooth) are responsible for connection to the devices. From the software aspects, every edge server is installed with operating systems (Linux) and OpenCV framework to support image processing tasks for mobile robots or machines. Based on the XenServer and Hadoop cloud ecosystem, we construct a private cloud platform supporting big data processing platform. Meanwhile, the wired ethernet and wireless communication system are used to data interaction between the different layers of smart manufacturing. In a word, an industrial manufacturing prototype platform with edge servers has been constructed.

Then based on the prototype platform, an experiment is constructed to evaluate the performance of the proposed. In the experiment, a mobile robot equipped with an industrial camera moved along with a fixed trajectory to monitor the operation of equipment of the constructed industry 4.0 platform. The robot periodically captured pictures and then to calculate the images boundary for the next work process. Then, the running state of the equipment is judged by a neural network. So, to assess the performance of different methods, these pictures are sent to different computing layer via industrial networks. Then, the related parameters are measured during the experiment.

In this experiment, a task for processing an image with the average size of 10Mb~20Mb was executed at the edge computing server (in which typical OpenCV algorithms were executed). The different edge servers communicate with each other via Ethernet with bandwidth 100Mbps. An edge server connects with the mobile robot with bandwidth 1~54Mbps. Moreover, the data is transmitted to the cloud server in four hops via industrial networks. Assume that the communication energy consumption is 1J/s and the computing energy consumption rate is 0.8J/s.

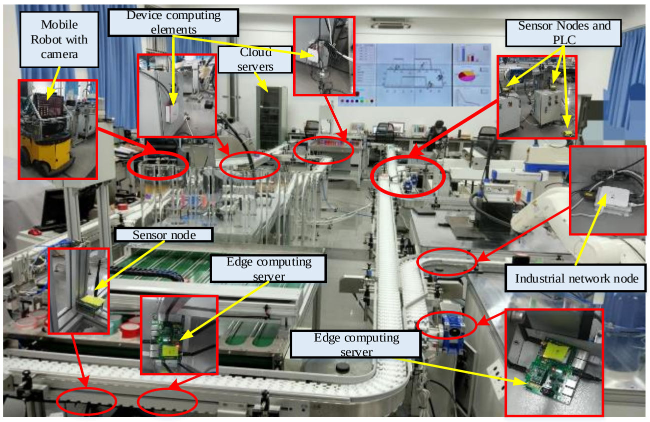


Figure 3. Prototyping Platform for smart manufacturing with edge computing

Meanwhile, we use the computing latency, satisfaction degree and the energy consumption as the evaluation metrics to assess the provided strategies via selecting algorithms for ES (SAE) and cooperating edge computing servers (CEC). We evaluate the performance of the proposed strategies with the comparison with traditional strategies such as cloud server computing (CSC), ordinary edge computing without scheduling (OEC).

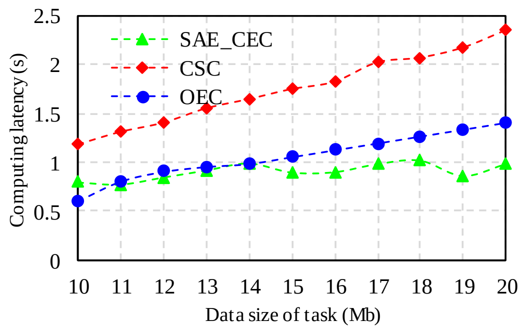
In particular, computing latency (CL) is the time between devices transmission data task from devices to the cloud server and receiving results at device. Satisfaction degree (SD) is the QoS lever performance metric. Satisfaction degree is expressed as the ratio between requirement computing service time and the computing latency. The higher SD implies the better performance.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2899679, IEEE Transactions on Industrial Informatics

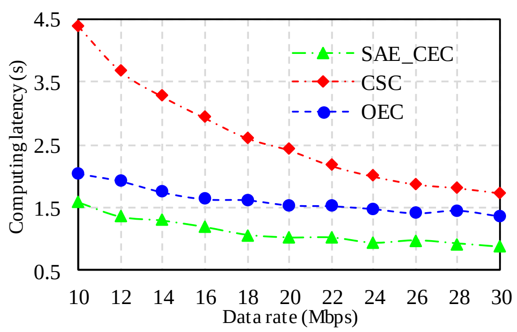
B. Analysis and Results

Fig. 4 show the average of computing latency of different methods with their best performance. In particular, Fig.4(a) shows that with an increment of the task data size, the computing latency increases for all these methods. It is obvious that our proposal (SAE_CEC) outperforms other strategies, as SAE_CEC will select the better edge server to complete the task. Moreover, with the increase the data size, the single edge server cannot meet the deadline, SAE_CEC will call Algorithm 2 and exploit multiple edge servers together to process the computing-extensive task to ensure the real-time constraints. It is worth mentioning that OEC gets better performance than CSC, as OEC adopts edge computing servers to finish the tasks. Meanwhile, the computing servers far away the device and system have to spend more time in data transmission. Hence, the CSC gets the worst computing latency performance.

Fig. 4(b) demonstrates the results the latency of different methods with the same task data size 10Mb in varied data rate between devices and server. Similarly, SAE_CEC shows a better performance than other algorithms. This because SAE_CEC method can dynamically select the better edge servers according to the deadline of the task. Furthermore, with the increase of data rate the latency of CSC decreases dramatically, as data rate has a great impact on CSC.

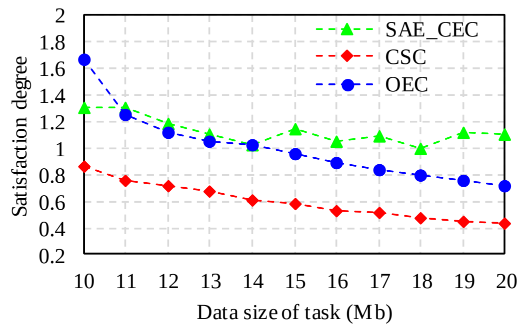


(a) Computing latency in different data size of task

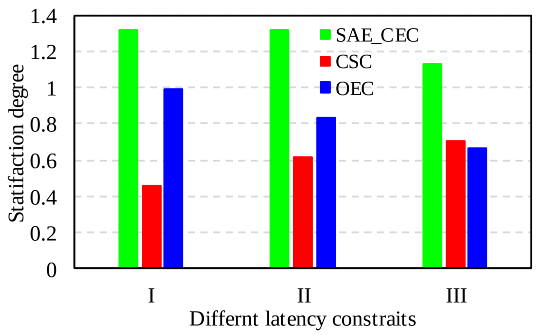


(b) Computing latency in different data rate
Figure 4. Comparison of computing latency

In addition, the satisfaction degree is useful to evaluate the computing services with the consideration of the latency constraints. In particular, in Fig.5(a) gives the results of satisfaction degree with different deadline requirements and the same data rate and the data size of the task. It is shown that when the data size of the task is less than 14Mb, satisfaction degrees of three methods are more than 100%. However, when the data size of the task is larger than 14Mb, only satisfaction degree of SAE_CEC can get 100%. That is to say, our proposal can meet the real-time requirement for processing the task. Fig. 5 (b) shows the satisfaction degree in different communication bandwidths and data amount of the task with different deadlines (I: 2s, II: 3s, III: 3s). Fig. 5(b) shows the similar results to Fig. 5(a). In summary, the proposals can adapt to different environments and different real-time constraints. The traditional methods cannot be adopted in the industrial maneuvering system due to the failure of fulfilling the real-time requirement.

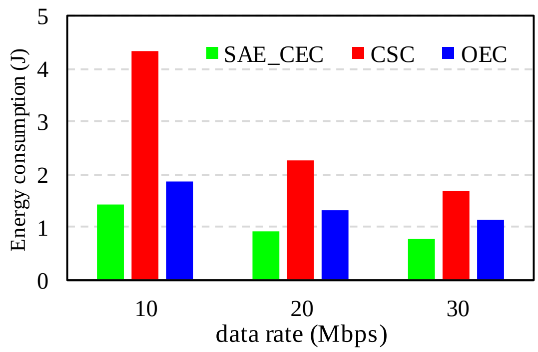


(a) Satisfaction degree in different data size of task

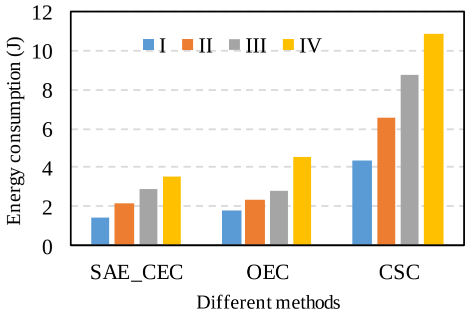


(b) Satisfaction degree in different time constraints
Figure 5. Comparison of satisfaction degree

Moreover, Fig 6 shows the average energy consumption of different methods in different data rate and task data size. Fig. 6(a) gives the results of energy consumption for different strategies. It is demonstrated that with the increment of data rate, the energy consumption will reduce, as all the methods spend less energy in communication. Meanwhile, the proposed scheme outperforms other strategies in terms of energy consumption because of edge server being close to device. In particular, when the data rate is 10Mpbs, our proposal can reduce more than 50% energy consumption. Fig. 6(b) shows the energy consumption in different task data size (I: 10, II, 15, III: 20, VI: 20 (Mb)) with same data rate. Similarly, when the data size increases, the energy consumption will be added. Fig. 6(b) also shows the advantage of our method in energy consumption.



(a) Energy consumption in different data rate



(b) Energy consumption in different task data size

Figure 6. Comparison of energy consumption

VI. Conclusions

In this paper, we have focused on resources scheduling to ensure the real-time requirement of smart manufacturing with consideration of integrating different computing resources. First, according to the feature of smart factories, AI task operations and network perspective, we provide the four-layer architecture with integration of historical heritage of computing resources. Then, we focus on the edge computing layer and propose a two-phase scheduling strategy to allocate the computing resources to meet the latency constraint. In the first phase, different factors are considered to select the edge computing server for supporting the computing services for the task with the low real-time constraint. Moreover, the second phase is explored for resolving cooperation of multiple ESs to construct a task of ESC operating the lower latency computing services. Finally, for verifying the feasibility of our proposal, a prototype platform is implemented. In particular, we conduct experiments to compare the traditionally-used methods with the proposed computing-resource scheduling strategy. The proposed computational resources allocation strategies have ensured the real-time for smart manufacturing with edge computing. In summary, the proposed frameworks and computing resources scheduling can accelerate the implementation of Industry 4.0 and smart factory.

Acknowledgment

This work was supported in part by the National Key Research and Development Project of China (No. 2017YFE0101000), the Joint Fund of the National Natural Science Foundation of China and Guangdong Province (No. U1801264), the Key Program of Natural Science Foundation of Guangdong Province (No. 2017B030311008), and the Science and Technology Program of Guangzhou, China (No. 201802030005). Imran's work is supported by the Deanship of Scientific Research, King Saud University through research group number RG-1435-051.

References

[1] J. Wan, S. Tang, D. Li, M. Imran, C. Zhang, C. Liu and Z. Pang, "Reconfigurable Smart Factory for Drug Packing in Healthcare Industry 4.0," IEEE Transactions on Industrial Informatics, vol. 15, no. 1, pp. 507-516, 2019.

[2] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee and B. Yin, "Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges," IEEE Access, vol. 6, pp. 6505-6519, 2018.

[3] M. Wollschlaeger, T. Sauter and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," IEEE Industrial Electronics Magazine, vol. 11, no. 1, pp. 17-27, March 2017.

[4] J. He, G. Jia, G. Han, H. Wang and X. Yang, "Locality-Aware Replacement Algorithm in Flash Memory to Optimize Cloud Computing for Smart Factory of Industry 4.0," IEEE Access, vol. 5, pp. 16252-16262, 2017.

[5] W. Zhang, Y. Liu, G. Han, Y. Feng and Y. Zhao, "An Energy Efficient and QoS Aware Routing Algorithm Based on Data Classification for Industrial Wireless Sensor Networks," IEEE Access, vol. 6, pp. 46495-46504, 2018.

[6] Y. Bi, G. Han, C. Lin, Q. Deng, L. Guo and F. Li, "Mobility Support for Fog Computing: An SDN Approach," IEEE Communications Magazine, vol. 56, no. 5, pp. 53-59, May 2018.

[7] X. Li, D. Li, J. Wan, A. V. Vasilakos, C. F. Lai and S. Wang, "A review of industrial wireless networks in the context of industry 4.0," Wireless Networks, vol. 23, no. 1, pp. 23-41, 2017.

[8] J. Wan, B. Yin, D. Li, A. Celesti, F. Tao and Q. Hua, "An Ontology-based Resource Reconfiguration Method for Manufacturing Cyber-Physical Systems," IEEE/ASME Transactions on Mechatronics, vol. 23, no. 6, pp. 2537-2546, 2018.

[9] P. K. Illa and N. Padhi, "Practical Guide to Smart Factory Transition Using IoT, Big Data and Edge Analytics," IEEE Access, vol. 6, pp. 55162-55170, 2018.

[10] G. Han, M. Guizani, J. Lloret, S. Chan, L. Wan and W. Guibene, "Emerging Trends, Issues, and Challenges in Big Data and Its Implementation toward Future Smart Cities: Part 2," IEEE Communications Magazine, vol. 56, no. 2, pp. 76-77, Feb. 2018.

[11] E. Shellshear, R. Berlin and J. S. Carlson, "Maximizing Smart Factory Systems by Incrementally Updating Point Clouds," IEEE Computer Graphics and Applications, vol. 35, no. 2, pp. 62-69, 2015.

[12] D. Georgakopoulos, P. P. Jayaraman, M. Fazia, M. Villari and R. Ranjan, "Internet of Things and Edge Cloud Computing Roadmap for Manufacturing," IEEE Cloud Computing, vol. 3, no. 4, pp. 66-73, July-Aug. 2016.

[13] M. Marjanović, A. Antonić and I. P. Žarko, "Edge Computing Architecture for Mobile Crowdsensing," IEEE Access, vol. 6, pp. 10662-10674, 2018.

[14] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1657-1681, 2017.

[15] H. Li, K. Ota and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," IEEE Network, vol. 32, no. 1, pp. 96-101, 2018.

[16] Y. Sun, S. Zhou and J. Xu, "EMM: Energy-Aware Mobility Management for Mobile Edge Computing in Ultra Dense Networks," IEEE Journal on Selected Areas in Communications, vol. 35, no. 11, pp. 2637-2646, 2017.

[17] J. Xu, B. Palanisamy, H. Ludwig and Q. Wang, "Zenith: Utility-Aware Resource Allocation for Edge Computing," 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, 2017, pp. 47-54.

[18] L. Yin, J. Luo and H. Luo, "Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4712-4721, 2018.

[19] T. M. Fernándezcaramés, P. Fragalamas, M. Suárezalbela and M. Vilarmontesinos, "A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard," Sensors, vol. 18, no. 6, 2018.

[20] M. Aazam, S. Zeadally and K. A. Harras, "Deploying Fog Computing in Industrial Internet of Things and Industry 4.0," IEEE Transactions on Industrial Informatics, vol. 14, no. 10,

pp. 4674-4682, 2018.

[21] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas and Q. Zhang, "Edge Computing in IoT-Based Manufacturing," IEEE Communications Magazine, vol. 56, no. 9, pp. 103-109, 2018.

[22] L. Li, K. Ota and M. Dong, "Deep Learning for Smart Industry: Efficient Manufacture Inspection System with Fog Computing," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4665-4673, 2018.

[23] Z. Zhou, J. Hu, Q. Liu, P. Lou, J. Yan and W. Li, "Fog Computing-Based Cyber-Physical Machine Tool System," IEEE Access, vol. 6, pp. 44580-44590, 2018.

[24] X. Li, D. Li, J. Wan, C. Liu and M. Imran, "Adaptive Transmission Optimization in SDN-Based Industrial Internet of Things with Edge Computing," IEEE Internet of Things Journal, vol. 5, no. 3, pp. 1351-1360, 2018.

[25] N. Dao, Y. Lee, S. Cho, E. Kim, K. Chung and C. Keum, "Multi-tier multi-access edge computing: The role for the fourth industrial revolution," 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2017, pp. 1280-1282.

[26] D. Park, S. Kim, Y. An and J. Y. Jung, "Lired: a light-weight real-time fault detection system for edge computing using LSTM recurrent neural networks," Sensors, vol. 18, no. 7, 2018.

[27] V. C. Gungor and G. P. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches," IEEE Transactions on Industrial Electronics, vol. 56, no. 10, pp. 4258-4265, 2009.

[28] X. Ding, Y. Tian and Y. Yu, "A Real-Time Big Data Gathering Algorithm Based on Indoor Wireless Sensor Networks for Risk Analysis of Industrial Operations," IEEE Transactions on Industrial Informatics, vol. 12, no. 3, pp. 1232-1242, 2016.

[29] P. O'Donovan, C. Gallagher, K. Bruton and D. T. J. O, 'Sullivan," A fog computing industrial cyber-physical system for embedded low-latency machine learning industry 4.0 applications," Manufacturing Letters, vol. 15, pp. 139-142, 2018.

[30] C. Chen, M. Lin and C. Liu, "Edge Computing Gateway of the Industrial Internet of Things Using Multiple Collaborative Microcontrollers," IEEE Network, vol. 32, no. 1, pp. 24-32, 2018.

[31] F. Tiago, P. Fragalamas, S. Manuel and D. Manuel, "A fog computing based cyber-physical system for the automation of pipe-related tasks in the industry 4.0 shipyard," Sensors, vol. 18, no. 6, 2018.

[32] J. Xu, B. Palanisamy, H. Ludwig and Q. Wang, "Zenith: Utility-Aware Resource Allocation for Edge Computing," 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, 2017, pp. 47-54.

[33] T. Yang, Y. Hu, M. C. Gursoy, A. Schmeink and R. Mathar, "Deep Reinforcement Learning based Resource Allocation in Low Latency Edge Computing Networks," 2018 15th International Symposium on Wireless Communication Systems (ISWCS), Lisbon, 2018, pp. 1-5.

[34] H. Yu, Q. Wang and S. Guo, "Energy-Efficient Task Offloading and Resource Scheduling for Mobile Edge Computing," 2018 IEEE International Conference on Networking, Architecture and Storage (NAS), Chongqing, 2018, pp. 1-4.

[35] X. Wang, K. Wang, S. Wu, S. Di, H. Jin, K. Yang and S. Ou, "Dynamic Resource Scheduling in Mobile Edge Cloud with Cloud Radio Access Network," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 11, pp. 2429-2445, 2018.

[36] L. Ni, J. Zhang, C. Jiang, C. Yan and K. Yu, "Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1216-1228, 2017.