The Journal of Systems & Software 193 (2022) 111456

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

EdgeWorkflow: One click to test and deploy your workflow applications to the edge☆

Jia Xu [a], Ran Ding [a], Xiao Liu [b,*], Xuejun Li [a,*], John Grundy [c], Yun Yang [d]

[a] School of Computer Science and Technology, Anhui University, Heifei, Anhui, China
[b] School of Information Technology, Deakin University, Geelong, Australia
[c] Faculty of Information Technology, Monash University, Melbourne, Victoria, Australia
[d] Department of Computing Technologies, Swinburne University of Technology, Melbourne, Australia

ARTICLE INFO

Article history:
Received 26 February 2022
Received in revised form 21 July 2022
Accepted 25 July 2022
Available online 30 July 2022

Dataset link: https://github.com/ISEC-AHU/EdgeWorkflow

Keywords:
Edge computing
Workflow system
Resource management
One-click deployment

ABSTRACT

In recent years, edge computing has become the ideal computing paradigm for various smart systems, such as smart logistics, smart health and smart transportation. This is due to its advantages including fast response times, energy efficiency and cost effectiveness over conventional cloud computing platforms. However, running complex computational scientific workflow tasks is still a very challenging issue at the edge, due to its typical three-layered computing environment consisting of an end device layer, an edge server layer, and a cloud server layer. A large number of recent studies have proposed different solutions for optimizing such computing resource management problems in an edge computing environment. However, since evaluation of most such studies is conducted through simulation, the effectiveness cannot be guaranteed in a real world environment. Therefore, to advance research on efficient execution and deployment problems for real world workflow applications using edge computing, an open-source edge workflow management system with comprehensive empirical evaluation capabilities is urgently required. This paper presents the first edge workflow system (named EdgeWorkflow) that is able to deploy user-created workflow applications to a real-world edge computing environment with "one-click" after optimizing the configuration with the simulation tool. With the aid of EdgeWorkflow, the user can automate the generation of specific edge computing environments, easily model and generate executable workflow applications with a visual modelling tool, effectively select various resource management methods included in the systems or apply their own resource management and task scheduling algorithms, efficiently monitor the statuses of computational tasks and obtain comprehensive reports on the execution results (such as those regarding time, cost and energy). We use an edge computing-based unmanned aerial vehicle (UAV) last-mile delivery system as a real-world case study, and a number of representative scientific workflows are employed for our experiments. Our experimental results show that EdgeWorkflow can effectively evaluate the performance of different resource management and workflow task scheduling algorithms and efficiently deploy and execute user-defined scientific workflow applications to user-specified edge computing environments.

1. Introduction

Thanks to the widespread adoption of the Internet of Things (IoT) and 5G technology, many smart applications, such as those in smart health, smart logistics and smart transportation, have entered our lives (Hong et al., 2019). These technologies have led to numerous heterogeneous smart end devices needing to access the Internet and requesting cloud services. These distributed end devices are connected via a cloud computing environment by various wired and wireless access networks (WANs), such as the 4G network, 5G network, and WiFi network. User requests and data are collected and transmitted to the cloud server by end devices in real time for further processing. However, on the one hand, traditional cloud data centres cannot support the processing of these requests and data in time. On the other hand, massive numbers of requests and large data transfers cause network congestion, high latency and wasted bandwidth. Taking the smart logistics scenario as an example, a large number of delivery devices are automated by various artificial intelligence-based applications (Lv et al., 2021). These applications require

☆ Editor: Jacopo Soldani.
* Corresponding authors.
E-mail addresses: xujia@ahu.edu.cn (J. Xu), dingran@stu.ahu.edu.cn (R. Ding), xiao.liu@deakin.edu.au (X. Liu), xjli@ahu.edu.cn (X. Li), john.grundy@monash.edu (J. Grundy), yyang@swin.edu.au (Y. Yang).

https://doi.org/10.1016/j.jss.2022.111456

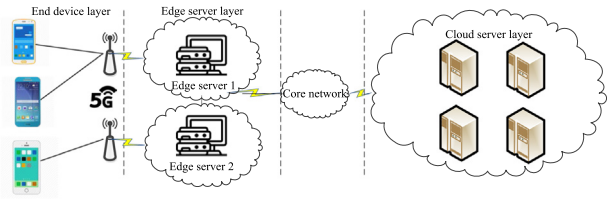

Fig. 1. The architecture of an edge computing environment.

end devices to collect sensor data and transfer them to the background for further processing. Their sensor data may reach the level of GBs per second (Wang et al., 2020a). If such applications are deployed using a conventional cloud computing environment, they would quickly consume too much network bandwidth. At the same time, the response time of these applications would also increase. However, because the computational tasks of smart logistics applications are time sensitive, transferring the end device's requests and sensor data to a cloud server for further processing and returning the results could lead to various serious problems, such as transportation accidents involving delivery vehicles and low delivery efficiency (Lv et al., 2021). To address such problems, moving computing resources from cloud servers to the edge of the network is the best solution.

Edge computing is a novel computing paradigm that has attracted much recent attention from both academia and industry due to its advantages, such as low latency, high bandwidth and location awareness. Therefore, edge computing has become a promising solution for smart systems. As depicted in Fig. 1, edge computing creates an extra edge server layer between the end device layer and the cloud server layer, as this added layer can provide computation resources that are much closer to the end devices (Lv et al., 2021; Wang et al., 2020a; Noghabi et al., 2020). With the support of edge servers, the time overhead for data transmission during computational offloading and the bandwidth demands of cloud data centres can be significantly reduced (Kong et al., 2021; Gao et al., 2021). As a result, the requirement of real-time response for many IoT smart applications can now be met. As shown in Fig. 1, the generic architecture of an edge computing environment consists of three layers: a layer of end devices, a layer of edge servers, and a layer of cloud servers (Yu and Li, 2021; Abouaomar et al., 2021; Liu et al., 2021). The end devices layer serves the end users directly and is also able to process some simple computation tasks. When a computation task is beyond the power of the end devices, either due to resource constraints or QoS (quality of service) constraints, the end devices offload the computation tasks to either the edge server layer or the cloud layer (Wang et al., 2021). The edge server layer serves the end devices and can offload tasks further to the cloud layer if they are beyond the power of the edge server. Finally, the cloud server layer deals with all the tasks that are offloaded from the end device or edge server layers. It also serves other purposes, such as data backup, due to its theoretically unlimited resources.

Despite the various advantages of edge computing environments, it is still an important but difficult problem to manage the computational tasks and resources efficiently in such environments (Kong et al., 2021; Gao et al., 2021). This issue is caused by two phenomena. First, there is a dependency between the computational tasks of a user's application, which means that the execution of one task may be constrained by another task. In general, these computational tasks with mutual dependency can be represented by workflow tasks. Second, the computing resources and network environment of an edge computing environment are more complex than those of a cloud computing environment. A workflow management system provides an interface for establishing, evaluating, monitoring and managing the given sequence of tasks (arranged as a workflow application). On that matter, a workflow management system has the unique advantage of managing complex workflow tasks and various computing resources (Hong et al., 2019).

Many research works focus on workflow tasks and resource management problems in edge computing environments (Rosa et al., 2021; Wang et al., 2019; Yin et al., 2018). However, the effectiveness verification of methods in different research works is conducted in the simulation stage. Some research works adopt simulation tools, such as CloudSim, iFogSim and FogWorkflowSim (Calheiros et al., 2011; Gupta et al., 2017; Liu et al., 2019). These simulation tools cannot provide a real edge computing environment for workflow task execution. In addition, the evaluations of resource management methods are based on the prediction of workflow task execution cost. However, the gap between the predicted execution cost and the real execution cost cannot be neglected in an edge computing environment. Other research works focus on real edge computing environment generation, such as SAND and PICasso (Akkus et al., 0000b; Lertsinsrubtavee et al., 0000c). Due to the lack of a workflow management system, these works cannot support the execution of different workflow applications, which severely limits their usability.

Considering the lack of a simple and easy-to-use edge-based workflow management system, this paper presents a "one-click edge workflow management system" (EdgeWorkflow) that supports visual modelling, customized computational task execution and one-click deployment. It contains two key components: a simulation platform and a real edge workflow execution engine. An open-source and easy-to-use edge computing workflow platform is required to serve as a common experimental platform to support a fair and comprehensive comparison with methods from relevant studies. To implement these solutions in real-world edge computing environments, an edge workflow execution engine is required to support the running of real workflow applications in smart systems. With the aid of EdgeWorkflow, the execution and deployment problems regarding workflow applications can be solved effectively, and user workflow applications can be executed more efficiently in the edge computing environment. The key contributions of this work include:

- we introduce a novel model for workflow applications and computing resources in an edge computing environment that supports a user-defined workflow structure and customized resource types;
- we design a novel workflow management system (EdgeWorkflow) framework that supports one-click deployment and can manage workflow applications in an edge computing environment efficiently. We implement EdgeWorkflow based on the proposed framework and a real edge computing environment;
- we use a case study-based validation of EdgeWorkflow by using an edge computing environment-based unmanned aerial vehicle (UAV) last-mile delivery system, to verify its effectiveness and efficiency. The experimental results highlight the unique advantage provided by EdgeWorkflow in terms of assessing the real performance (such as the energy consumption of the end device, workflow execution cost and workflow execution time) of edge workflow management methods.

The remainder of this paper is organized as follows. The research motivation of edge workflow management systems is presented in Section 2. Section 3 presents the service framework and integration framework of EdgeWorkflow. Section 4 illustrates the design and implementation of EdgeWorkflow in

detail. Section 5 discusses and demonstrates the various evaluation indicator-based experimental results and the threats to validity of EdgeWorkflow. Section 6 introduces the related work regarding the experimental resource management platform and computational resource management problem in an edge computing environment. Section 7 summarizes this paper and points out some future work. Finally, the discussion, and the adoption of EdgeWorkflow by the community and the control flow of the system are attached in Appendices[1] A, B and C.

2. Motivation

As an edge computing environment includes three different layers of resources and each has a unique type of resources, resource management is a complicated and challenging issue (Siriwardhana et al., 2021; Zhou et al., 2020). Generally, the problem of resource management in edge computing is twofold: the first is the computation offloading problem that determines which layer the computation tasks should be offloaded to, and the second involves the resource allocation and task scheduling problems within different layers (Khan et al., 2020). Clearly, a set of resource management strategies is required to effectively manage and optimize the usage of different layers of resources. Otherwise, many issues may occur, such as excessive energy consumption by the end devices, excessive time overhead and cost incurred when using edge and cloud services, and unsatisfactory QoS for end users (Bozorgchenani et al., 2020).

A user's application consists of various computational tasks with dependencies, which means that the execution sequence of the tasks needs to follow certain constraints. In general, these complex computational tasks can be modelled as various kinds of workflow applications. Fig. 2 shows an example of a typical edge computing-based smart UAV (unmanned aerial vehicle) last-mile delivery system. This system can be divided into three layers: a UAV layer (as the end device), an edge server layer and a cloud server layer. The UAV layer is responsible for sensing and gathering environmental data during the flight mission. Due to the limited computing power and restricted battery life of the UAV, computation offloading strategies need to decide where the computing tasks should be executed based on the characteristics of these tasks; for example, they could be offloaded to the cloud, offloaded to the edge, or executed locally on the UAV itself. We can see the dependencies among the computing tasks in the receiver recognition process, which is represented as a DAG (directed acyclic graph)-based workflow application. The key issue for improving the QoS of edge computing-based smart applications is to effectively manage different computing resources and computing tasks. However, this can be a very challenging job for software developers who do not have in-depth knowledge and skills with respect to edge computing. Workflow technology has been widely used to automate task and resource management. The challenges for software developers mainly arise from three aspects, which are the generation of edge computing environment, the construction of the workflow model, and the efficient execution and monitoring of workflow tasks. Besides, there are many workflow applications that have been deployed in the edge computing environments. Therefore, designing effective resource management methods such as workflow offloading and scheduling algorithms is a very important research issue for edge computing workflow applications.

To make our workflow management system highly efficient and easy to use, also considering the challenges mentioned above, we identify five major requirements for edge computing-based workflow management systems.

[1] https://doi.org/10.6084/m9.figshare.20338605.

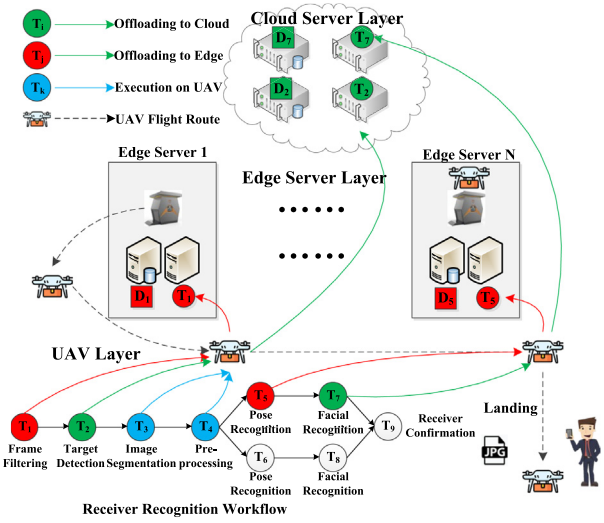

Fig. 2. An edge computing-based smart UAV last-mile delivery system.

(1) ***The generation and deployment of a user-specified real edge computing environment***: Different from a cloud computing environment, an edge computing environment has various heterogeneous computing resources and a network environment. Traditional cloud environments usually use virtualization technology to generate and deploy different computing resources. However, it cannot adapt to diversiform edge computing resources and network environments.

(2) ***The visual modelling and generation of submitted user workflow tasks for various smart application***s: Most of the existing workflow execution engines support scientific workflow application modelling. They lack the support for various smart applications. However, an edge computing environment has many smart applications. The computational tasks in these applications are highly customized and substantially different from those in a scientific workflow.

(3) ***A highly efficient workflow execution engine designed for an edge computing environment***: Current mainstream workflow engines were designed based on a cloud computing environments. Given the computing resource and network differences between edge computing and cloud computing, it would be unrealistic to transplant the cloud workflow engine to an edge computing environment directly.

(4) ***The real-time monitoring of the edge computing environment and workflow tasks***: Due to the real-time characteristics of edge computing environments, the performance metrics of workflow task execution change at very fast rates. As a result, the proposed engine needs to support edge computing environment monitoring in real time.

(5) ***The support for optimizing the various performance objectives of workflow applications***: According to the current resource management research works in edge computing environments, there are many evaluation indicators for workflow task execution, such as energy consumption, execution cost and execution time. Therefore, the proposed engine needs to support various evaluation indicators for workflow tasks.

Considering the lack of a simple and easy-to-use edge workflow engine and the above five requirements, this paper presents a one-click edge workflow management system (EdgeWorkflow) that supports visual modelling, customized computational task execution and one-click deployment. To the best of our knowledge, none of the existing workflow management systems simultaneously support one-click deployment and the modelling

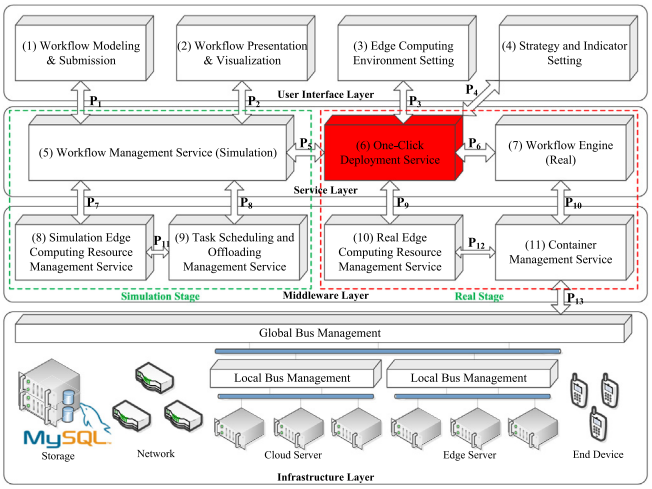

Fig. 3. The service framework of EdgeWorkflow.

of computing resources and networks in edge computing environments. Therefore, EdgeWorkflow provides a perfect platform for the investigation of computational offloading and resource management problems in edge computing environments.

3. Our approach

In this section, the general framework of EdgeWorkflow, which integrates a one-click workflow engine into edge computing is proposed. The general framework introduces all major services involved in edge computing workflow management by the user's perspective, from the user establishing an edge computing environment and modelling workflow tasks to executing workflow tasks. Then, the key modules of the middleware layer are introduced which include the workflow management simulator, one-click deployment mechanism and workflow engine. The details of each service module are described in integration framework.

3.1. Service framework

The service framework of EdgeWorkflow is shown in Fig. 3. It is divided into four layers: the infrastructure layer, middleware layer, service layer and user interface layer. EdgeWorkflow requires three types of user input data for the ***user interface layer*** which include workflow model settings, edge computing environment settings, and optimization strategy and indicator settings. The ***service layer*** generates the best workflow task execution plan by using a workflow management simulation service and deploys the user's workflow tasks to the real workflow engine. After the ***middleware layer*** receives the edge computing environment settings and user workflow tasks, the instructions for constructing a real edge computing environment are sent to the *bottom layer*. The ***infrastructure layer*** receives the instructions from the upper layer and establishes the real edge computing resource and network environment.

The main function of the ***user interface layer*** is to receive the user's requests and format them into structured, sensible data for the lower layer. This layer implements the requirements for visual modelling of workflow tasks and real-time monitoring of the edge computing environment and workflow tasks (Requirements 2, 4). It obtains several service modules for different user requests, including workflow modelling and submission (Service 1, Requirement 2), workflow presentation and visualization (Service 2), edge computing environment setting (Service 3) and strategy and indicator setting (Service 4). This layer allows users to monitor the workflow execution process in real time through a visual interface (Requirement 4).

The ***service layer*** receives structured data from the upper layer, which includes the workflow instance, edge computing environment setting, and strategy and indicator setting. Then, the best workflow instance execution plan is generated by the workflow management simulation service. Finally, the workflow instance is deployed and executed with Service 6. This layer implements the requirements for workflow task's generation and execution (Requirements 2, 3). It includes three main services: a workflow management simulation service (Service 5, Requirement 3), one-click deployment service (Service 6, Requirement 2) and real workflow engine (Service 7, Requirement 3). This layer helps to generate and execute the user's workflow task model efficiently.

The ***middleware layer*** receives the deployment instructions from the upper layer and sends user's workflow instance for execution in a real edge computing environment. It bridges the gap between the service layer and infrastructure layer. The middleware layer implements the requirements for the deployment of a user-specified real edge computing environment and optimizing the various performance objectives of workflow applications (Requirements 1, 5). It consists of four services: a simulation edge computing resource management service (Service 8, Requirement 5), task scheduling and offloading management service (Service 9, Requirement 5), real edge computing resource management service (Service 10, Requirement 5) and container management service (Service 11, Requirement 1). This layer supports the efficient execution of user's workflow instances.

The bottom layer is the ***infrastructure layer***, which contains various resources in the edge computing environment, such as storage, network and computing resources. This layer implements the requirements for the generation of a user-specified real edge computing environment (Requirement 1). The storage resources are implemented by the MySQL database, which provides the data access service for the upper layers. The network resources provide a network modelling and establishment function for the real edge computing environment. The computing resources provide different types of computing resources for user workflow instance execution which include cloud servers, edge servers and end devices. The infrastructure layer enables more efficient generation of the edge computing environment according to the user's requirements.

3.2. Integration framework

EdgeWorkflow enables the creation of a real edge computing environment and the execution of real workflow tasks. The simulation function for the evaluation of different resource and task management strategies is inherited from our previous work FogWorkflowSim (Gao et al., 2021). The computing resources in the edge computing environment are created using Docker containers (Wang et al., 2021). The integration framework of EdgeWorkflow is shown in Fig. 4. Specifically, it can be divided into three layers, which are the user interface layer, workflow management system layer, and storage and computing resource layer.

The user interface layer consists of four modules, which are workflow setting module, edge computing environment setting module, strategy setting module, and optimization objective setting module. The workflow setting module has three sub-modules. Firstly, the workflow customization sub-module can visually create workflow models with customized workflow structures. Secondly, the task binding sub-module binds the workflow task model with real computing tasks. At last, the workflow selection

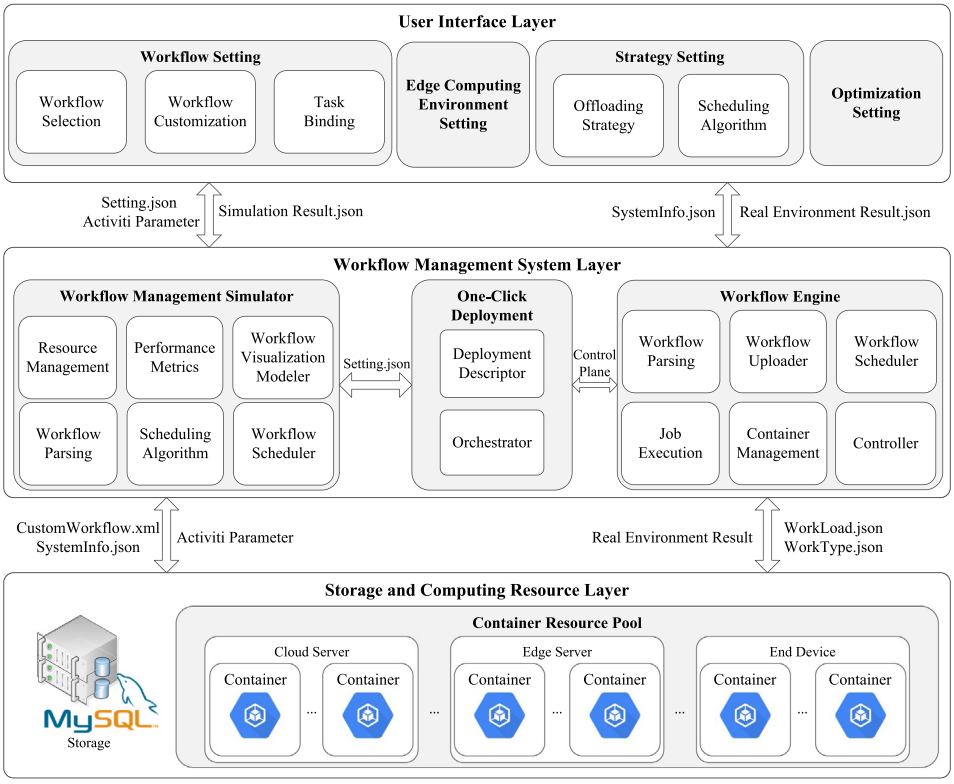

Fig. 4. The control and data flow of the one-click workflow engine.

sub-module allows the user to select the workflow instance for execution. The edge computing environment setting module is to configure the computing and network resources according to user's requirements. The strategy setting module has two sub-modules: computation offloading strategy sub-module and workflow scheduling algorithm sub-module. The function of computation offloading strategy sub-module is to select the task offloading strategy for user's workflow instance. The function of the task scheduling sub-module is to select the workflow scheduling algorithm for user's workflow instance. The function of optimization objective setting module is to choose the optimization objectives such as execution time of workflow instance, energy consumption of the end device and execution cost of workflow instance.

The workflow management system layer is composed of the workflow management simulator module, one-click deployment module, and the workflow engine module. The workflow management simulator module is to facilitate the performance evaluation of different resource and task management strategies using a simulation edge computing environment so that users can select the most suitable strategies for their workflow applications before running them in the real edge computing environment which they need to pay for use of computing resources. Specifically, the workflow simulator module is composed of six sub-modules, which are resource management sub-module, performance metrics sub-module, workflow visualization modeller sub-module, workflow parsing sub-module, scheduling algorithm sub-module, and workflow scheduler sub-module. The resource management sub-module is to create the simulation edge computing environment according to user's settings. The performance metrics sub-module is to evaluate the execution plan of the workflow instance under user's optimization objectives. The workflow visualization modeller sub-module is to visualize the workflow model in the system interface. The workflow parsing sub-module is to analyse the standard workflow XML files and generate executable workflow models in the simulation edge computing environment. The scheduling algorithm sub-module contains various task scheduling algorithms. It can generate the best execution plan for user's workflow instance model in the simulation edge computing environment. Finally, the workflow scheduler sub-module is to schedule the running of workflow tasks on the simulation computing resources.

The one-click deployment module is the key module in EdgeWorkflow, which connects the workflow management simulator module and the workflow engine module. With the help of this module, the user can deploy the real edge computing environment with one click and easily execute their workflow instance. The one-click deployment module consists of two sub-modules: deployment descriptor sub-module and orchestrator sub-module. The deployment descriptor sub-module collects various types of deployment configuration information such as edge computing environment setting and workflow instance setting. The orchestrator sub-module creates the running environment for user's workflow instance and calls the workflow engine module by one-click deployment module.

The workflow engine module has two main functions. Firstly, it receives the deployment instructions from the one-click deployment module and executes the user's workflow instance in the real edge computing environment. Secondly, the workflow engine needs to monitor various performance metrics for workflow execution in real time. The workflow engine module is composed of six sub-modules: workflow parsing sub-module, workflow uploader sub-module, workflow scheduler sub-module, job execution sub-module, container management sub-module, and controller sub-module. The workflow parsing sub-module can parse the workflow model and then generate an executable workflow instance. The workflow uploader sub-module is to receive the workflow instance and upload it to scheduler for execution. The workflow scheduler sub-module is to manage the

Table 1
Parameter settings of real edge computing environment.

Device type	Type	MIPS	Cost
End device	Small	1000	0
	Middle	2000	0
	Large	3000	0
Edge server	Small	1300	0.48
	Middle	2600	0.78
	Large	3900	1.08
Cloud server	Small	1600	0.96
	Middle	3200	1.66
	Large	4800	2.36

execution of workflow tasks on the computing resources according to the computation offloading and workflow scheduling plans generated by the optimization algorithms. The job execution sub-module is responsible for workflow job execution according to the instructions from the workflow scheduler module. The container management sub-module is the same as the computing resource management module in the KNIX system (Rosa et al., 2021). Finally, the controller sub-module manages and coordinates all other sub-modules in the workflow engine module.

The storage and computing resource layer is composed of MySQL database and container resource pool. The MySQL database stores various system data generated by EdgeWorkflow, such as the data for workflow structure, edge computing resources, workflow tasks, and so on. The container resource pool can generate various types of computing resources in the edge computing environment to support workflow task execution.

4. Implementation of edgeworkflow

In this section, we introduce the implementation of the five major components in the system framework in detail, which include the edge computing environment, workflow management simulator, one-click deployment module, workflow engine and user interface. To make it easier for users to understand the organization structure and operational flow of EdgeWorkflow, the control flow of the workflow management simulator and workflow engine is provided in Appendix C.

4.1. Implementation of edge computing environment

The computing resource layer of EdgeWorkflow is implemented based on container technology and the KNIX serverless computing platform (KNIX, 2021). First, this layer reads the user's edge computing environment settings profile. Then, the edge computing resources and network environment are established to satisfy the different resource requirements of various smart systems. The different kinds of computing resources' processing speed and cost is shown in Table 1 (Liu et al., 2019; Cao et al., 2015). The uplink and downlink bandwidths of the end device layer are 20 Mbps and 40 Mbps, respectively (Liu et al., 2019). The working and idle powers of the end devices are 700 mW and 30 mW, respectively (Liu et al., 2019). The data transmission and receiving power of the end devices are 100 mW and 25 mW, respectively (Liu et al., 2019; Cao et al., 2015; Netjinda et al., 2014).

4.2. Implementation of workflow management simulator

4.2.1. Resource management sub-module

The resource management sub-module is divided into a resource module, a scheduling module and a controller module. The resource module is a virtual pool for computation and storage resources for various types of devices. The parameter settings for the simulation of an edge computing environment are described in Table 2 (Liu et al., 2019).

Table 2
Parameter settings of simulation edge computing environment.

Parameters	End device	Edge server	Cloud server
MIPS	1000	1300	1600
Running power (mW)	700	0	0
Idle power (mW)	30	0	0
Data transmission power (mW)	100	0	0
Data receiving power (mW)	25	0	0
Task execution cost	0	0.48	0.96

The scheduling module provides a library of various task scheduling algorithms. The controller module is a model base for the models of various performance metrics that evaluate the performance of workflow scheduling algorithms.

4.2.2. Performance metric sub-module

There are three performance metrics defined in the system library: execution time, energy consumption and cost. Given the above three-layer edge computing architecture, the execution of tasks can be divided into two cases, viz. local computation at the end devices or offloading to the edge servers/cloud servers (Liu et al., 2019). In addition to the given performance metrics, EdgeWorkflow also supports customized metrics by allowing users to import their own evaluation metric models into the system library (Gupta et al., 2017; Xu et al., 2019a; Kuang et al., 2019). However, due to space limit, the detail of computational task's execution time, cost and energy consumption models are attached in Appendix D.

4.2.3. Visual modeller sub-module

When the standard workflow in the system library does not meet the needs of users, users employ the visual modeller sub-module to create their own workflow DAG. During the process of creating a workflow DAG, users draw the workflow structure; input the task workload, input parameters and output parameters of the workflow nodes; transfer the task workloads of edges that connect nodes; finish the drawing of the workflow DAG structure; and then generate the workflow file in XML format through the workflow DAG for users to utilize by themselves.

4.2.4. Workflow parsing sub-module

The parser module is responsible for parsing the input workflow file in XML format into the task class, which represents the workflow task in the system. The attributes of task class include the task load and inter-task dependencies. The module parses all tasks and related attributes contained in the XML files into various attribute parameters in the task class.

4.2.5. Offloading and scheduling algorithm sub-module

In this part, we introduce the details of the task offloading and scheduling algorithms currently supported in EdgeWorkflow. In the edge computing environment, given the three different layers of resources, viz. the end device layer, edge server layer and cloud server layer, a task offloading strategy is required first to determine which level or levels of resources will be used for task computation before the task scheduling algorithm determines the order of the tasks to be run on the allocated resources. Due to space limit, the details of the offloading and scheduling algorithms are attached in GitHub.[2]

[2] https://github.com/ISEC-AHU/EdgeWorkflow.

(1) Task offloading algorithm

The strategies of task offloading in the edge computing environment can be generally divided into three types, viz. local computation, full offloading and partial offloading (Hayat et al., 2021). In local computation, all tasks are executed on the user's end device. This strategy is suitable for small task loads or when other edge resources are inaccessible. In full offloading, all tasks are offloaded wholly to the edge servers and/or cloud servers. This strategy is not suitable when the size of the data transmitted among the resources is excessive. In the partial offloading strategy, tasks can be offloaded partially to the edge servers and/or the cloud servers. This strategy is useful for tasks with complicated computations. For example, the end device can perform simple data pre-processing and then offload complicated computation tasks with pre-processed data to the edge servers or cloud servers. It has been reported that the overall computation time for complicated tasks can be reduced (Mach and Becvar, 2017). Given the complexity of many workflow tasks, EdgeWorkflow adopts the partial offloading strategy.

(2) Task scheduling algorithm

The current algorithm library of EdgeWorkflow includes six task scheduling algorithms: MinMin, MaxMin, FCFS, RoundRobin, PSO and the GA (Liu et al., 2019). Among them, the MinMin, MaxMin, FCFS and RoundRobin algorithms are representative heuristic-based algorithms, while PSO and the GA are widely used metaheuristic-based algorithms (Liu et al., 2019).

4.3. Implementation of one-click deployment mechanism

(1) Deployment descriptor

When the simulation process in the workflow management simulator is completed, the parameters of the edge computing environment in setting.json are passed to the one-click deployment module. The deployment descriptor module calls createSingleContainer() in the IndexService class of the workflow engine and creates a real edge computing environment with the same configuration for the current workflow application.

(2) Orchestrator

Before creating the real edge computing environment, the orchestrator is responsible for parsing the edge computing environment parameters of setting.json passed by the workflow management simulator and passing the parsing results to the deployment descriptor sub-module. During the creation of the real edge computing environment, the orchestrator is responsible for receiving the returned monitoring information in real time. After the creation of the real edge computing environment is complete, the orchestrator is responsible for passing feedback to the user to let the user know the creation details of the real edge environment. After the execution of the deployment descriptor sub-module is complete, the orchestrator passes the workflow engine scheduling parameters to the workflow engine module. After the workflow engine completes the scheduling and execution of the workflow tasks, the orchestrator receives the execution results of the workflow tasks returned by the workflow engine in the real environment.

4.4. Implementation of workflow engine

(1) Workflow parsing sub-module

The workflow parsing sub-module is responsible for parsing the input XML-formatted workflow file into the task class of the workflow task that the system can recognize. The task class includes two important attributes: the task load and dependencies between tasks, as well as some other attributes.

(2) Workflow uploader sub-module

As an auxiliary module, the workflow uploader sub-module is responsible for verifying and saving the workflow DAG structure submitted by users and verifying the code and input and output parameters of workflow tasks. When user selects a customized workflow and uses the workflow uploader sub-module. The workflow uploader sub-module verifies the workflow structure and workflow task code submitted by the user. The EdgeWorkflow support computational tasks based on Java. Finally, the user's customized task code binds with the workflow task according to the Taskbinding class for workflow engine scheduling use.

(3) Workflow scheduler sub-module

As the entry of the workflow scheduling algorithm, the workflow scheduler sub-module is the most important sub-module. When WorkflowEngine decides which computer in the edge computing environment a workflow task will be executed on, the WorkflowEngine class invokes an algorithm in the workflow scheduler sub-module according to the size of each task, the number of tasks and the configured edge computing environment. The workflow scheduler sub-module is responsible for creating a virtual machine and for the submission, update, and return of workflow tasks.

(4) Job execution sub-module

The job execution sub-module is responsible for the execution of workflow tasks in the edge computing environment and the monitoring of the execution process. After the execution of the workflow scheduler sub-module is completed, the workflow engine passes parameters such as the execution host and task quantity of each task to the workflow scheduler sub-module. After receiving the related parameters of the workflow task, the workflow execution sub-module executes the workflow task according to the corresponding environment and monitors the execution process in real time. After the completion of the execution process, information such as the start time, end time and execution host of the task are returned to the workflow engine.

(5) Container management sub-module

The container management sub-module is responsible for creating, deleting and managing containers based on the edge computing environment in the user's workflow application. When executing a workflow application, the workflow engine calls createSingleContainer() in the IndexService class and uses the created container to generate a real edge computing environment for executing the corresponding workflow application.

(6) Controller sub-module

The controller sub-module is responsible for receiving instructions from the user interface layer and scheduling the above sub-modules uniformly. The controller sub-module is also responsible for data interaction in the ***in***frastructure layer and the user interface layer. When the execution of the user's workflow application is completed, the results of the application are returned to the user interface layer and presented to the user in the form of statistical graphs and tables.

4.5. Implementation of user interface

EdgeWorkflow is implemented in Java, and the source code is available on GitHub.[3] To demonstrate its capability, we provide a Web UI[4] so that users can access EdgeWorkflow remotely and run their created edge workflow applications on our computing infrastructure. As shown in Fig. 5, the main page of EdgeWorkflow is divided into four areas: workflow and edge computing environment setting (Area 1), execution result details (Area 2), workflow

[3] https://github.com/ISEC-AHU/EdgeWorkflow.

[4] http://www.iseclab.org.cn:8080/EdgeWorkflow.

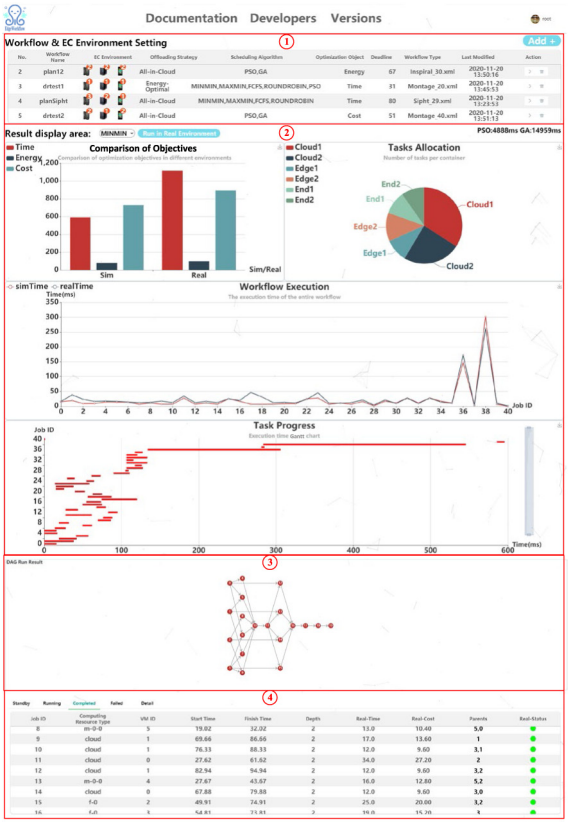

Fig. 5. Main page of EdgeWorkflow Web UI.

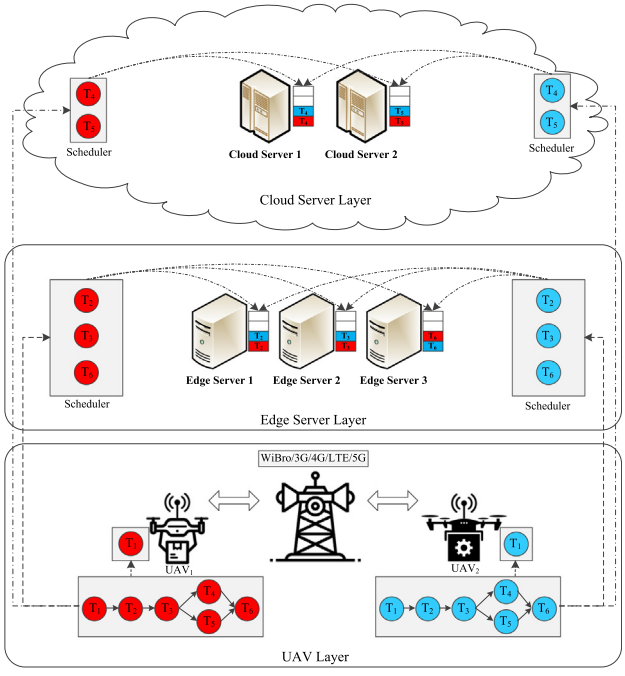

Fig. 6. A typical workflow task offloading and scheduling plan.

execution result visualization and edge computing environment setting (Area 3), and dynamic runtime monitoring (Area 4). To create a new edge workflow application, users can click the "Add" button in Fig. 5. Due to space limit, the details of the user operations and system interface are attached in Appendix E.

5. Evaluation

EdgeWorkflow is the first workflow execution engine that supports one-click deployment and the modelling of computing resources and networks in edge computing environments simultaneously. To showcase the ease of EdgeWorkflow in terms of function and performance, this section shows the various performance indicators for workflow execution based on different computational resource optimization problems in edge computing environment. First, a typical case study is used to demonstrate the usefulness of the EdgeWorkflow system for the end device's energy consumption optimizing problem under deadline constraint. Second, when optimizing the makespan of workflow tasks, we evaluate the two most important performance metrics in the EdgeWorkflow system, which are the task execution time with different computing resources and the task execution time differences between the simulation and real environments. Finally, for different optimization metrics, various optimization algorithms in the EdgeWorkflow system were chosen to optimize the task execution time, cost and energy consumption of end devices.

5.1. Case study: The workflow task management problem in an edge computing-based UAV last-mile delivery smart system

This section demonstrates the impact of workflow task management plans on energy consumption, task execution time and cost metrics between the simulation and real environments by using the workflow task management problem of energy consumption optimization under deadline constraint. The case study is chosen by a real-world UAV delivery system framework (EXPRESS). EXPRESS is a UAV last-mile delivery smart system that combines the MEC environment and blockchain technology from our previous work (Xu et al., 2019b).

A typical edge computing environment for UAV last-mile delivery smart systems is shown in Fig. 6, which includes a UAV layer, an edge server layer and a cloud server layer (Xu et al., 2019b). The UAV layer has two UAVs, and each UAV needs to execute a workflow instance abstracted by an artificial intelligence application (Xu et al., 2019a,b). The connection between the UAV layer and cloud server layer is a wide area network (WAN), and the data transmission speed is 512 KB per second. The connection between the UAV layer and edge server layer is a local area network (LAN), and the data transmission speed is 100 MB per second (Xu et al., 2019b; Yu et al., 2020). Assuming that the CPU frequencies of the UAV, edge server and cloud server are 0.7 GHz, 1.4 GHz and 2.1 GHz respectively, the cost of the edge server and cloud server are $1.08 and $2.36 dollars per hour, respectively. The transmission power, task execution power and idle power of the UAVs are 0.1 W, 0.7 W, and 0.03 W, respectively (Xu et al., 2019b). The parameter settings of workflow tasks are show in Tables 3 and 4. The deadline constraint of each workflow is 12 s. The task management problem in EXPRESS focuses on optimizing the energy consumption of end devices while ensuring workflow deadline constraint.

A typical workflow task offloading and scheduling plan generated by the PSO algorithm is shown in Fig. 6. For each workflow instance, we can clearly see that one computational task is executed in the UAV layer, three computational tasks are offloaded to the edge server and two computational tasks are offloaded to the cloud server. Tables 3 and 4 show the experimental results of the workflow instance makespan, cost and energy consumption in the simulation and real edge computing environment.

In general, all evaluation indicators for executing two workflow instances have differences between the simulation and real edge computing environments. First, the makespan of two workflow instances are lower than the deadline constraint in the

Table 3
Results of the simulation and real edge computing environment for UAV_1.

Task	Task transmit/ Receive data (KB)	Workload (MIPS)	Task offloading decision	Simulation			Real		
				Task execution/ Transfer time (s)	E/(J)	C/($) 10^{-3}	Task execution/ Transfer time (s)	E/(J)	C/($) 10^{-3}
T_1	1000/2000	500	UAV	0.71/0	0.50	0	0.71/0	0.50	0
T_2	6000/5000	2000	Edge server	1.43/0.11	0.05	0.43	1.43/0.11	0.05	0.43
T_3	3000/2500	3000	Edge server	2.14/0.05	0.07	0.64	2.14/0.05	0.07	0.64
T_4	800/500	1500	Cloud server	0.71/2.54	0.28	0.50	0.71/2.54	0.28	0.50
T_5	650/500	2500	Cloud server	1.19/2.25	0.25	0.83	1.19/2.25	0.25	0.83
T_6	6000/4000	2500	Edge server	1.79/0.10	0.06	0.54	3.58/0.10	0.12	1.07
Total				9.77	1.21	2.94	11.56	1.27	3.47

Table 4
Results of the simulation and real edge computing environment for UAV_2.

Task	Task transmit/ Receive data (KB)	Workload (MIPS)	Task offloading decision	Simulation			Real		
				Task execution/ Transfer time (s)	E/(J)	C/($) 10^{-3}	Task execution/ Transfer time (s)	E/(J)	C/($) 10^{-3}
T_1	1000/2000	500	UAV	0.71/0	0.50	0	0.71/0	0.50	0
T_2	6000/5000	2000	Edge server	1.43/0.11	0.05	0.42	2.86/0.11	0.10	0.86
T_3	3000/2500	3000	Edge server	2.14/0.05	0.07	0.64	4.28/0.05	0.07	1.28
T_4	800/500	1500	Cloud server	0.71/2.54	0.28	0.50	1.42/2.54	0.30	0.99
T_5	650/500	2500	Cloud server	1.19/2.25	0.25	0.83	2.38/2.25	0.30	1.67
T_6	6000/4000	2500	Edge server	1.79/0.10	0.06	0.54	1.79/0.10	0.06	0.54
Total				9.77	1.21	2.94	14.53	1.63	5.34

simulation environment. For UAV_1 and UAV_2's workflow instance, the total makespan is 9.77 s in the simulation environment. However, For UAV_1, the makespan of the same workflow instance in the real environment is 11.56 s, which is 18.32% higher than that in the simulation environment. Second, the energy consumption of two workflow instances is 1.21 J in the simulation environment. However, For UAV_1, the energy consumption of the same workflow instance in the real environment is 1.27 J, which is 7.26% higher than that in the simulation environment. This is because most simulation environments generate evaluation results by using the workflow model and evaluation model, and the workflow tasks are not executed in a real environment. Therefore, various influential factors in the real edge computing environment are not considered in the simulation environment. For example, there is a task delay event in the first workflow instance execution process. Task 6 of both UAV_1 and UAV_2 are offloaded to edge server 3 simultaneously. We can clearly see that UAV_2's Task 6 enters the execution queue at first in Fig. 6. UAV_1's Task 6 needs to wait for the completion of the preceding task before it can start execution.

For UAV_2's workflow instance, the total makespan and energy consumption are still 9.77 s and 1.21 J in the simulation environment because the simulation stage cannot fully consider the influential factors generated by other device workflow instances. However, there are four task delay events in the real workflow instance execution process. The makespan and energy consumption of the same workflow instance in the real environment are 14.53 s and 1.63 J, which are 48.72% and 34.71% higher than that in the simulation environment. This also results in the total makespan of UAV_2's workflow instance beyond the deadline constraint. Besides, the task management plan generated in simulation environment fails in the real environment. Therefore, the actual execution result of UAV_1 and UAV_2's workflow instance is higher than that of the simulation environment in terms of makespan and energy consumption.

In edge computing-based UAV last-mile delivery smart systems, most workflow applications are time-sensitive. This means that when the makespan of the workflow instance exceeds the given deadline constraint, the workflow execution plan is meaningless. In addition, users typically need to rent the computing resources for edge computing according to their workflow execution plan. Once the workflow instance's deadline constraint is not satisfied, the users must pay an additional price for overtime. Taking UAV_2's workflow instance as an example, the total cost is $\$5.34 \times 10^{-3}$ in the real environment. However, the same workflow instance's total cost in the simulation environment is $\$2.94 \times 10^{-3}$, which is 44.94% lower than that in the real environment. Notably, although the experimental result of the simulation environment clearly demonstrates the computational task's execution time, cost and energy consumption in most simple situations, it still cannot fully verify the effectiveness of the utilized resource management methods when influential factors occur, such as task delays in a real edge computing environment. Therefore, evaluation in a real edge computing environment is essential for verifying the effectiveness of the chosen resource management method.

5.2. Performance of EdgeWorkflow

For a simple and easy-to-use edge workflow execution engine, the experimental results of different indicators should be efficient, stable and effective. Therefore, this section focuses on the workflow makespan optimization problem and evaluates the two key performance indicators of EdgeWorkflow, which are the task execution time with different computing resources and the task execution time difference between the simulation and real environments.

For demonstration purposes, EdgeWorkflow currently runs on the Alibaba Cloud with the following configuration: an Intel(R) Xeon(R) Platinum 8369HC dual core CPU at 3.4 GHz, 4 GB of RAM, a 40 GB ROM, and the Ubuntu 18.04 64-bit operating system. The container uses Docker Version 19.03.6. The database is MySQL 5.7.31. EdgeWorkflow is developed in Java JDK 1.8. EdgeWorkflow can also be deployed in any user-created computing environment for their smart systems, as the computing resources are managed by the KNIX system in the form of containers (KNIX, 2021).

When optimizing the makespan of workflow tasks, there are two performance metrics that are most likely to affect the optimization results in the real edge computing environment. They

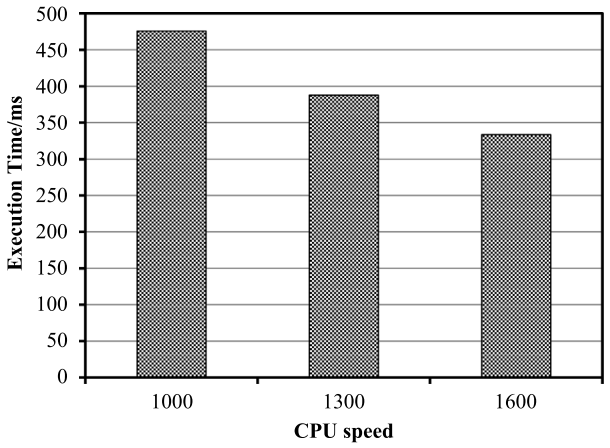

Fig. 7. The task execution time with different computing resources in EdgeWorkflow.

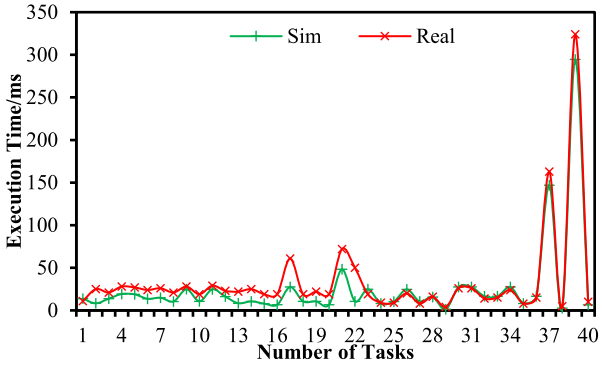

Fig. 8. The differences between the task execution time of the simulation and real environment in EdgeWorkflow.

are the execution time stability of computing resources and the makespan effectiveness of the workflow. To verify the relationship between the workflow task execution time and different numbers of computing resources, we select Montage workflow and measure the workflow execution time with different types of computing resources. The small (1000 MIPS), middle (1300 MIPS) and large types (1600 MIPS) of computing resources in the cloud server layer are selected. The experimental results are shown in Fig. 7. **With increasing CPU speed, the workflow execution time gradually decreases.** For example, when the CPU speed is 1000 MIPS, the workflow execution time is 475.9 ms. When this value increases to 1300 MIPS, the Montage workflow execution time is reduced to 387.8 ms. This shows that the workflow execution time drops by 18.51% since the CPU speed increases by 30%. When this value increases to 1600 MIPS, the Montage workflow execution time is reduced to 333.6 ms. This shows that the workflow execution time drops by 13.97% since the CPU speed increases by 23.07%. Therefore, ***the workflow task execution time is highly relevant to the CPU speed of the computing resources***. This shows that the computing resources of EdgeWorkflow have strong stability.

As described in the case study section, the workflow task's execution time in the simulation edge computing environment is different from that in the real environment and cannot fully verify the effectiveness of the resource management methods. Fig. 8 demonstrates the difference between the task execution time of the simulation and real environment in EdgeWorkflow. In general, **the task execution time in the real environment is**

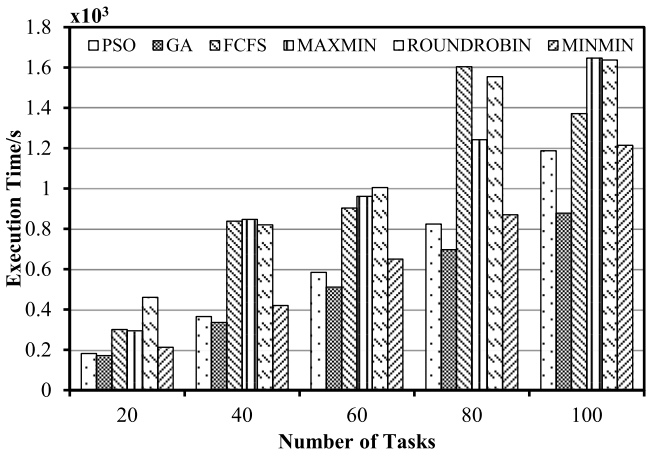

Fig. 9. Comparison of task execution time.

Table 5
Parameter settings of edge computing environment.

Parameters	End device	Edge server	Cloud server
MIPS	1000	1300	1600
Running power	700	0	0
Idle power	30	0	0
Data transmission power (mW)	100	0	0
Data receiving power (mW)	25	0	0
Task execution cost ($)	0	0.48	0.96
Number of device	2	2	2

on average 20% higher than that in the simulation environment. This is because the resource conflict and task delay factors influence the workflow instance's execution process and increase the task execution time. In some situations, the task execution time in the real environment is lower than that in the simulation environment, such as for Task 27 and Task 28. This occurs because the workload of the computational task is too low to trigger factors such as resource conflict and task delay. Therefore, this experimental result demonstrates the high effectiveness of EdgeWorkflow.

5.3. Evaluation of different workflow instances

We describe the evaluation of EdgeWorkflow to demonstrate its effectiveness and usability. First, the performance of different algorithms is compared under the task execution time, cost and energy consumption of end device according to the different optimization objectives, and the algorithm with the best experimental result is found. Second, the workflow execution Gantt chart and distribution situation of task offloading plan are demonstrated. EdgeWorkflow supports various popular scientific workflow structures with different numbers of tasks. Here, we use the Montage workflow (20, 40, 60, 80, 100 tasks) (Deelman et al., 2020) as an example and compare various task scheduling algorithms under different optimization objectives. The parameter settings of the edge computing environment are shown in Table 5.

(1) Workflow task execution time optimization

The experimental results of six task scheduling algorithms in terms of their task execution time are shown in Fig. 9. **It can be seen that the PSO and GA algorithms always achieve better performance in terms of task execution time than the other four algorithms**. As the number of tasks increases, the gap between GA and PSO increases. For example, when the number of tasks is 20, the task execution time under GA is 6.43% lower than that under PSO. When the number of tasks becomes 100, the gap

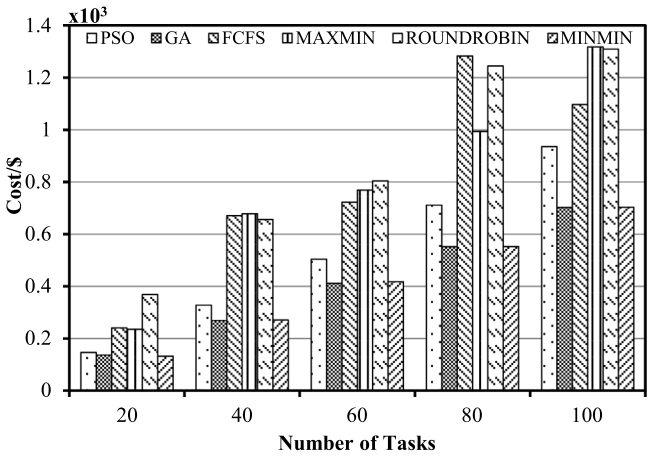

Fig. 10. Comparison of task execution cost.

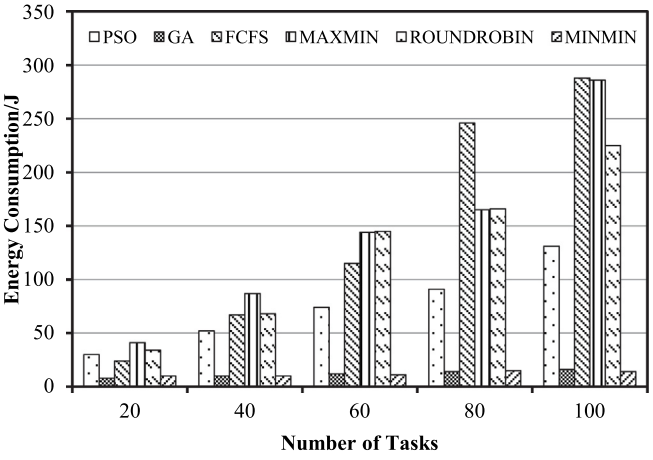

Fig. 11. Comparison of the energy consumption values of the end devices.

is increased to 35.19%. Therefore, GA is the most effective task scheduling algorithm for reducing the task execution time under the current experimental settings.

(2) Workflow task execution cost optimization

The results in terms of the task execution cost of the six task scheduling algorithms are shown in Fig. 10. Since the FCFS, MAXMIN, ROUNDROBIN and MINMIN algorithms are only applicable to optimize the task execution time, they are only involved in the comparison as benchmark algorithms in this part. **It is clear that the two metaheuristic algorithms always find the optimal task scheduling plan to minimize the task execution cost compared to the other four algorithms**. In addition, as the number of tasks grows, the gap between GA and PSO increases. For example, when the number of tasks is 20, the task execution cost under GA is 6.57% lower than that under PSO. When the number of tasks becomes 100, the gap is increased to 33.14%. Therefore, **GA is the most effective task scheduling algorithm** for reducing the task execution cost under the current experimental settings.

(3) End device's energy consumption optimization

Fig. 11 shows the experimental results of the six task scheduling algorithms in terms of the energy consumption of the end devices. The execution energy consumption values of the end devices **with the GA algorithm are always the lowest compared to those of the other five algorithms.** With the increase in the number of tasks, the performance gap between GA and other algorithms increases as well. For example, when the number of

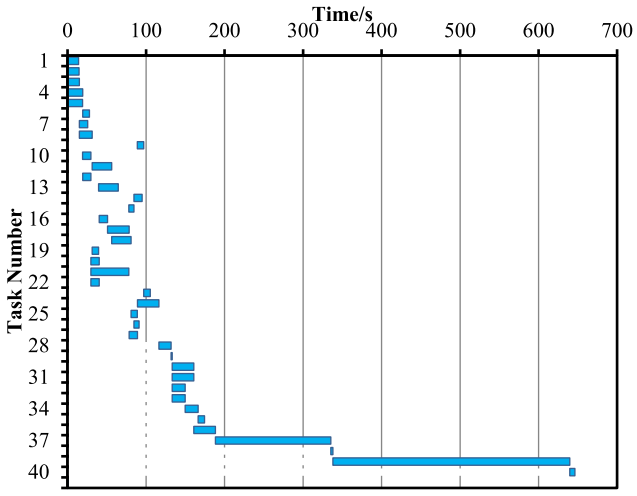

Fig. 12. The Gantt charts of various workflow tasks.

tasks is 20, the task execution time under GA is 400% lower than that under PSO. When the number of tasks becomes 100, the gap is increased to 718.75%. Therefore, **GA is the most effective task scheduling algorithm** for reducing the energy consumption of end devices under the current experimental settings.

The test performance of all competing algorithms for three optimization objectives is summarized in Table 6, which includes the best and mean (variance) of total execution time, cost and energy consumption across 10 independent runs. In Table 6, it can be seen that the GA algorithm achieves the lowest average (variance) execution time, cost and energy consumption, respectively.

For a deeper understanding of the workflow task execution process, EdgeWorkflow provides workflow execution Gantt charts for users. The 40-task Montage workflow execution Gantt chart is shown in Fig. 12. It is easy to see the timing plot of each workflow task. For example, computational tasks 1–4 are executed in parallel. Computational tasks 7 and 10 are executed in sequence. In addition, the tasks with the highest execution time cost can be easily found, which are task 37 and task 39 in Fig. 12.

For different workflow tasks, **the most suitable computing resources may be different because of the varying characteristics of computational tasks**. For example, computationally intensive tasks may be more suitable for the cloud server. Data-intensive tasks may be more suitable for the edge server. The task offloading strategy needs to offload these computational tasks to the most suitable resources according to their characteristics. Therefore, **the task distribution situation of the current offloading plan is a key indicator that makes it easy to optimize the computing resource configuration**. As shown in Fig. 13, EdgeWorkflow provides a visual display of the task distribution situation. For the current task offloading plan, the cloud server, edge server and end device are accounted for 27%, 54% and 19% of the total, respectively. In addition, it can be observed that most computational tasks are offloaded to the edge server, which means that this offloading plan needs the most computing resources in the edge server.

5.4. Threats to validity

In this section, we discuss the threats to the validity of our EdgeWorkflow system, which include external threats and internal threats.

External threats to validity. There are two threats to the external validity of EdgeWorkflow, which are the instrumentation

Table 6
The best and mean (standard deviation) optimization objects of six algorithm across 10 independent runs.

Optimization objectives		PSO		GA		FCFS		MAXMIN		ROUNDROBIN		MINMIN	
		best	*mean(std.)*	*best*	*mean(std.)*	*best*	*mean(std.)*	*best*	*mean(std.)*	*best*	*mean(std.)*	*best*	*mean(std.)*
Time	20	200	205(14.54)	**156**	**159(4.71)**	304	311(44.32)	295	305(56.10)	455	469(93.16)	156	164(80.46)
	40	349	361(59.60)	**338**	**345(23.29)**	861	873(88.54)	843	852(42.54)	825	836(89.73)	326	341(71.33)
	60	891	908(65.96)	**506**	**514(32.77)**	892	900(54.90)	939	954(94.40)	1011	1024(76.23)	507	518(70.32)
Cost	20	160	164(9.31)	**124**	**127(1.69)**	242	248(28.74)	236	244(35.90)	362	375(76.54)	124	131(57.33)
	40	276	288(59.30)	**268**	**276(29.99)**	688	698(58.45)	673	681(27.13)	661	668(46.76)	260	272(45.64)
	60	712	726(42.21)	**404**	**411(21.63)**	712	720(30.03)	750	763(60.42)	808	819(48.79)	404	414(45.01)
Energy	20	66	67(0.62)	**7**	**8(0.50)**	20	21(1.11)	41	43(6.49)	33	34(0.71)	7	8(1.43)
	40	111	112(1.07)	**11**	**12 (0.68)**	73	75(8.77)	83	85(9.83)	71	73(2.18)	11	13(2.18)
	60	181	185(6.71)	**12**	**13(2.22)**	116	119(4.22)	140	144(7.6)	142	145(6.46)	13	14(0.93)

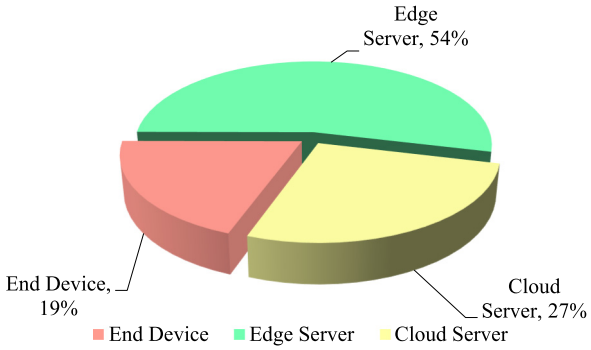

Fig. 13. The task distribution situation of the current offloading plan.

threat and selection of subject threat. First, the instrumentation threat means that the deployment environment may influence the experimental results of EdgeWorkflow. This is due to the performance metrics (execution time, energy consumption) of workflow execution heavily depending on the performance of the underlying infrastructure. Therefore, for the same workflow application, the performance metric of the workflow may yield different results. However, as we can see from Section 4.1, the infrastructure layer of EdgeWorkflow is implemented based on container technology and the KNIX serverless computing platform (Hong et al., 2019). Container technology is a method for packaging a workflow task that includes everything needed to run a workflow application: the code, runtime, system tools, system libraries and settings (Lv et al., 2021). Therefore, the workflow task can be run with dependencies while remaining isolated from other processes and infrastructure layers. With the help of container technology, the gaps between the performance metrics yielded by different infrastructure layers are minimized. Second, the selection of subject threat refers to the representativeness and supportiveness of the workflow visual modelling module in EdgeWorkflow. Currently, there are different kinds of workflow application structures. Different structures have different performance. Most workflow systems are designed either to support scientific workflow applications (which are computation-centric) or business workflow applications (which are transaction-centric). The scope of workflow applications in this paper is mainly for scientific workflow applications and is not ideal for business workflow applications. Meanwhile, in order to run user-created workflow applications, there are mainly two aspects of users' effort needed to make their workflows supported by EdgeWorkflow. First of all, for the workflows with modelling requirements, the users need to model the workflow application by using the Edgeworkflow modelling language (XML file) or visual workflow modelling feature and ensure the subcomponents (e.g., data store, compute, UI, etc.) are modelled for simulation. Then, for the workflows that are already represented by other models, the users need to convert existing workflow model to this XML format. In EdgeWorkflow, the design of the workflow visual modelling module is based on the Activiti engine, which is a kind of lightweight, java-centric open-source workflow execution engine for supporting real-world business process automation requirements (Wang et al., 2020a). When users need to deploy their workflow applications in real edge computing environments, the EdgeWorkflow system's task binding feature supports computational tasks based on Java. In addition, the beta version of EdgeWorkflow has received nearly 4000 service requests to date (Noghabi et al., 2020). Many researchers have begun to use EdgeWorkflow to implement their own methods for solving workflow application and resource management problems in edge computing environments. Therefore, the workflow visual modelling module of EdgeWorkflow should be representative.

Internal threats to validity. Most of the internal threats to validity are derived from the parameter setting of the algorithm and the effectiveness of the experimental results. There are two threats to the internal validity of EdgeWorkflow which are the interaction effects threat between the experimental variables and the experimental result effectiveness threat. First, the interaction effects threat is caused by having different types of experimental parameter settings for various algorithms and edge computing environments. To overcome this threat, all comparative experiments follow the single-variable principle, which means that all experimental parameter settings are the same, except for the comparison variable. Furthermore, most of the existing resource management methods in EdgeWorkflow are commonly used algorithms. We use the recommended ranges for the parameter settings to achieve the best performance with EdgeWorkflow. As a result, the threat of interaction effects between the experimental variables can be minimized effectively. Second, the experimental result effectiveness threat is caused by the experimental result gap between simulation and real environment. This is due to the fact that the simulation edge computing environment cannot take into account factors such as computing resource conflicts in real environments. In order to fully consider the effect of such factors in the final experimental results, the difference between the task execution times of the simulation and the real environment in EdgeWorkflow is demonstrated in Section 5.2, Fig. 8. In general, the experimental results show that the difference between the simulation and the real environment experimental results of the EdgeWorkflow system is stable at about 20%, which proves the stability and effectiveness of the system.

6. Related work

The resource management problem is one of the most crucial issues in an edge computing environment. Many resource management methods have been proposed to optimize the different indicators of workflow application execution. In this situation, a

general experimental platform for edge computing environments is urgently needed. In recent years, a large number of existing experimental platforms have been developed for cloud and edge computing environments. Here, we discuss three different types of research works associated with resource management experimental platforms in edge computing environments.

In the cloud computing environment, there are many workflow simulation and real experimental platforms for solving resource management problems. For example, CloudSim is the most commonly used and popular workflow simulation platform in cloud environments (Calheiros et al., 2011). However, CloudSim was designed according to the architecture of the cloud and hence cannot be used to simulate the much more complicated resource types and network topologies in edge environments (Calheiros et al., 2011). Chen and Deelman (2012) proposed WorkflowSim by extending CloudSim, which is a simulation toolkit for scientific workflows in distributed environments. WorkflowSim provides a higher layer of workflow management but still cannot support the generation of a real edge computing environment. Wang and Altintas (2012) proposed the Kepler scientific workflow engine, which supports the high-level workflow design and execution of scientific workflows. Nevertheless, Kepler still cannot support modelling and generating edge computing resources and customized user workflows. Therefore, these workflow simulation and real experimental platforms can only support the evaluation of workflow execution in cloud environments. At the same time, they are also limited to the traditional cloud environment and cannot support the first and third key requirement of the edge workflow system in Section 1, which involves the one-click generation of a real edge computing environment and workflow execution engine of edge computing environment.

Several workflow simulation experiment platforms in edge computing environments have been proposed for solving the evaluation problem. Gupta et al. (2017) proposed iFogSim, which is a modelling and simulation toolkit for resource management problems in edge computing environments. Nevertheless, iFogSim does not use a workflow system to manage resources, and hence, its capability is limited. To solve the evaluation problem of workflow task execution in an edge computing environment, Liu et al. (2019) proposed an automated edge workflow simulation tool. FogWorkflowSim overcomes many drawbacks possessed by the previously developed methods and supports user workflow task modelling and execution and edge computing environment modelling simultaneously. However, FogWorkflowSim is used for research purposes only and cannot support the execution of workflow tasks in a real edge computing environment. We can clearly see that the most important limitation of these experimental platforms is that they are established in a simulation environment and lack support for real edge computing environments and workflow task modelling and generation. Therefore, they cannot support the second and fourth key requirements of the edge workflow system in Section 1, which involve the modelling, generation workflow tasks in various smart applications and real-time monitoring of the edge computing environment and workflow tasks.

There are many workflow execution engines that are applied in various scenarios, such as Activiti, JBPM, Pegasus, Taverna, and KNIX (KNIX, 2021; Deelman et al., 2020; Medeiros, 2021; Swiderski, 2021; Wolstencroft et al., 2013). These can be divided into two types: business workflow engines and scientific workflow engines. Business workflow engines are used to manage the instance intensive business processes of large software applications. For instance, Activiti is a kind of lightweight, Java-centric open-source workflow execution engine for supporting real-world business process automation requirements (Medeiros, 2021). Scientific workflow engines are used in computation intensive scientific workflow applications. For example, Deelman et al. (2020) developed a scientific workflow management system for scientific applications called Pegasus. It achieves reliable, scalable workflow execution across a wide variety of computing infrastructures. Therefore, these workflow execution engines have the power to support modelling and generating various workflow tasks, which include large software applications and scientific applications. However, most of them are limited to do not support the various performance objectives optimizing of workflow applications, which is the fifth key requirement of the edge workflow system in Section 1.

Generally speaking, the problem of resource management in edge computing is mainly in twofold: the first is the computation offloading problem that determines which layer the computation tasks should be offloaded and the second is the resource allocation and task scheduling problem within different layers (Huang et al., 2021; Hao et al., 2021; Plachy et al., 2021). Clearly, a set of resource management strategies is required to effectively manage and optimize the usage of different layers of resources. Yu et al. (2020) proposed a joint task offloading and resource allocation algorithm based on the successive convex approximation. This algorithm reduces the service response time and energy consumption of the end device. Lu et al. (2020) investigated the offloading problem of multiple edge servers in the mobile edge computing environment. A deep reinforcement learning-based offloading strategy is proposed for solving the task offloading problem. The experimental results show that the proposed algorithm has better performance in energy consumption of end device and task execution latency. Zhang et al. (2020) studied the task offloading problem in edge computing-based vehicular networks. The MDP-based task offloading algorithm was presented to minimize the task execution delay. Wang et al. (2020b) proposed an imitation learning-based task scheduling algorithm for online vehicular in edge computing environment. This algorithm can solve the computational task scheduling with a few samples offline. From these research works, we could know that the resource management problem in edge computing environment is very hot topic. However, most of the research works only perform in simulation level. The simulation experiments cannot verify the validity of the algorithms effectively. Therefore, an edge workflow engine is urgently needed.

For monitoring problems in the cloud or edge computing environments, existing research work is divided into two main categories, which are infrastructure monitoring and SLA monitoring respectively. In the aspect of infrastructure monitoring, Xu et al. (2019) focused on the problem of data incompleteness during task offloading process in edge computing environment and propose blockchain-enabled computation offloading and resource monitoring methods. Battula et al. (2019) analysed the resource monitor service of traditional methods in cloud computing environment. A support and confidence-based (SCB) approach is proposed to monitor and minimize resource usage in edge computing environment. In the aspect of SLA monitoring, Khan et al. (2022) focused on the SLA monitor problem for cloud-hosted services and proposed a blockchain-based automatic SLA monitoring system for ensuring transparency and trustworthiness between cloud providers and consumers. Battula et al. (2022) considered the unique characteristic of fog resources and proposed an automated SLA management framework for fog computing environment to monitor and enforce SLAs in a more trustworthy manner. Similar to the research work of resource management, most infrastructure and SLA monitoring studies in cloud or edge computing environments are designed for simulation environments. The simulation environments cannot verify the effectiveness of the monitoring algorithms effectively. Therefore, the real-time monitor system of the edge computing environment and workflow tasks is needed.

Table 7
Comparison of various workflow management toolkits with proposed EdgeWorkflow.

Toolkit	Features						
	Real EC environment support	Workflow customized support	Workflow execution engine	Real-time monitoring support	Various evaluation indicators	Visual modelling	Different management algorithms
CloudSim (Calheiros et al., 2011)			✓				✓
WorkflowSim (Chen and Deelman, 2012)			✓				✓
Kepler (Wang and Altintas, 2012)		✓	✓		✓	✓	✓
Edge-Fog Cloud (Mohan and Kangasharju, 2017)					✓		✓
iFogSim (Gupta et al., 2017)				✓	✓		✓
FogWorkflowSim (Liu et al., 2019)			✓	✓	✓		✓
Activiti (Medeiros, 2021)		✓	✓			✓	
JBPM (Swiderski, 2021)		✓	✓			✓	
Pegasus (Deelman et al., 2020)		✓	✓				
Taverna (Wolstencroft et al., 2013)			✓		✓		
KNIX (KNIX, 2021)	✓	✓					
EdgeWorkflow	✓	✓	✓	✓	✓	✓	✓

In summary, the abovementioned different types of research works all indicate that current workflow simulations or real experimental platforms have various serious problems. None of the existing workflow engines can be effectively and easily deployed in edge computing environments. Therefore, a truly simple and easy-to-use edge workflow engine requires both efficient workflow task execution and one-click deployment. EdgeWorkflow covers all five requirements and provides users with a perfect platform for the investigation of computational offloading and resource management problems in an edge computing environment. The advantages of EdgeWorkflow over the current mainstream workflow management toolkits are shown in Table 7.

7. Conclusions and future work

Edge computing environments provide a new solution for the deployment of various workflow-based smart applications because of their low latency and highly flexible computing resources. This paper proposes a highly efficient one-click workflow management system for edge computing environments (EdgeWorkflow). Compared with the existing workflow management system, EdgeWorkflow realizes the one-click deployment feature by three functions: (1) automatic generation and deployment of the edge computing environment; (2) scientific and customized visual modelling, generation and deployment of workflow tasks; and (3) real-time monitoring for the workflow execution process. In addition, a graphical user interface is developed for monitoring edge computing environments and various evaluation indicators for workflow tasks in real time. The effectiveness and efficiency of EdgeWorkflow are validated by a case study involving the workflow task management problem of an edge computing-based UAV last-mile delivery smart system and the evaluation of different scientific workflow instances. To the best of our knowledge, EdgeWorkflow is the first edge workflow execution engine for evaluating the validity of various resource management methods that support various practical and scientific workflows and different evaluation indicators.

In the future, we will expand its capability to support more built-in and user's arbitrary applications submitted in the form of containers and web services so that it can be used for the development of many different edge computing-based smart systems. Besides, the cost of converting other existing workflow models to the EdgeWorkflow model is also worthy to be studied in the future.

CRediT authorship contribution statement

Jia Xu: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Ran Ding:** Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Xiao Liu:** Validation, Formal analysis, Investigation, Resources, Writing – review & editing, Supervision, Project administration. **Xuejun Li:** Validation, Formal analysis, Investigation, Resources, Writing – review & editing, Project administration. **John Grundy:** Validation, Formal analysis, Investigation, Resources, Writing – review & editing. **Yun Yang:** Validation, Formal analysis, Investigation, Resources, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data and code are available at https://github.com/ISEC-AHU/EdgeWorkflow

Acknowledgement

This work was supported by the National Natural Science Foundation of China Project No. 61972001. John Grundy is supported by Australian Research Council Laureate Fellowship FL190100035.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jss.2022.111456.

References

Abouaomar, A., Cherkaoui, S., Mlika, Z., Kobbane, A., 2021. Resource provisioning in edge computing for latency sensitive applications. IEEE Internet Things J..

Akkus, I.E., Chen, R., Rimac, I., Stein, M., Satzke, K., Beck, A., Aditya, P., Hilt, V., 0000b. {SAND}: Towards High-Performance Serverless Computing, pp. 923–935.

Battula, S.K., Garg, S., Montgomery, J., Kang, B., 2019. An efficient resource monitoring service for fog computing environments. IEEE Trans. Serv. Comput. 13 (4), 709–722.

Battula, S.K., Garg, S., Naha, R., Amin, M.B., Kang, B., Aghasian, E., 2022. A blockchain-based framework for automatic SLA management in fog computing environments. J. Supercomput. 1–31.

Bozorgchenani, A., Mashhadi, F., Tarchi, D., Monroy, S.S., 2020. Multi-objective computation sharing in energy and delay constrained mobile edge computing environments. IEEE Trans. Mob. Comput..

Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. - Pract. Exp. 41 (1), 23–50.

Cao, S., Tao, X., Hou, Y., Cui, Q., 2015. An energy-optimal offloading algorithm of mobile computing based on HetNets. pp. 254–258.

Chen, W., Deelman, E., 2012. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. 1–8..

Deelman, E., da Silva, R.F., Vahi, K., Rynge, M., Mayani, R., Tanaka, R., Whitcup, W., Livny, M., 2020. The pegasus workflow management system: Translational computer science in practice. J. Comput. Sci. 101200.

Gao, H., Huang, W., Duan, Y., 2021. The cloud-edge-based dynamic reconfiguration to service workflow for mobile ecommerce environments: A QoS prediction perspective. ACM Trans. Internet Technol. (TOIT) 21 (1), 1–23.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S.K., Buyya, R., 2017. IFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Softw.: Pract. Exp. 47 (9), 1275–1296.

Hao, Y., Cao, J., Wang, Q., Du, J., 2021. Energy-aware scheduling in edge computing with a clustering method. Future Gener. Comput. Syst. 117, 259–272.

Hayat, S., Jung, R., Hellwagner, H., Bettstetter, C., Emini, D., Schnieders, D., 2021. Edge computing in 5G for drone navigation: What to offload? IEEE Robot. Autom. Lett. 6 (2), 2571–2578.

Hong, Z., Chen, W., Huang, H., Guo, S., Zheng, Z., 2019. Multi-hop cooperative computation offloading for industrial IoT–edge–cloud computing environments. IEEE Trans. Parallel Distrib. Syst. 30 (12), 2759–2774.

Huang, X., Zhang, W., Yang, J., Yang, L., Yeo, C.K., 2021. Market-based dynamic resource allocation in mobile edge computing systems with multi-server and multi-user. Comput. Commun. 165, 43–52.

Khan, K.M., Arshad, J., Iqbal, W., Abdullah, S., Zaib, H., 2022. Blockchain-enabled real-time SLA monitoring for cloud-hosted services. Cluster Comput. 25 (1), 537–559.

Khan, L.U., Yaqoob, I., Tran, N.H., Kazmi, S.A., Dang, T.N., Hong, C.S., 2020. Edge-computing-enabled smart cities: A comprehensive survey. IEEE Internet Things J. 7 (10), 10200–10232.

2021. Knix MicroFunctions (formerly known as SAND serverless). https://knix.io/.

Kong, X., Wang, K., Wang, S., Wang, X., Jiang, X., Guo, Y., Shen, G., Chen, X., Ni, Q., 2021. Real-time mask identification for COVID-19: an edge computing-based deep learning framework. IEEE Internet Things J..

Kuang, Z., Li, L., Gao, J., Zhao, L., Liu, A., 2019. Partial offloading scheduling and power allocation for mobile edge computing systems. IEEE Internet Things J. 6 (4), 6774–6785.

Lertsinsrubtavee, A., Ali, A., Molina-Jimenez, C., Sathiaseelan, A., Crowcroft, J., 0000c. PiCasso: A lightweight edge computing platform, pp. 1–7.

Liu, G., Chen, X., Zhou, R., Xu, S., Chen, Y.-C., Chen, G., 2021. Social learning discrete particle swarm optimization based two-stage X-routing for IC design under intelligent edge computing architecture. Appl. Soft Comput. 104, 107215.

Liu, X., Fan, L., Xu, J., Li, X., Gong, L., Grundy, J., Yang, Y., 2019. FogWorkflowSim: An automated simulation toolkit for workflow performance evaluation in fog computing, pp. 1114–1117.

Lu, H., Gu, C., Luo, F., Ding, W., Liu, X., 2020. Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning. Future Gener. Comput. Syst. 102, 847–861.

Lv, Z., Chen, D., Lou, R., Wang, Q., 2021. Intelligent edge computing based on machine learning for smart city. Future Gener. Comput. Syst. 115, 90–99.

Mach, P., Becvar, Z., 2017. Mobile edge computing: A survey on architecture and computation offloading. IEEE Commun. Surv. Tutor. PP (99), 1.

Medeiros, E.D., 2021. Activiti: Open source business automation. http://www.activiti.org/.

Mohan, N., Kangasharju, J., 2017. Edge-Fog cloud: A distributed cloud for Internet of Things computations. 1–6..

Netjinda, N., Sirinaovakul, B., Achalakul, T., 2014. Cost optimal scheduling in iaas for dependent workload with particle swarm optimization. J. Supercomput. 68 (3), 1579–1603.

Noghabi, S.A., Cox, L., Agarwal, S., Ananthanarayanan, G., 2020. The emerging landscape of edge computing. GetMob.: Mob. Comput. Commun. 23 (4), 11–20.

Plachy, J., Becvar, Z., Strinati, E.C., di Pietro, N., 2021. Dynamic allocation of computing and communication resources in multi-access edge computing for mobile users. IEEE Trans. Netw. Serv. Manag..

Rosa, W., Clark, B.K., Madachy, R., Boehm, B., 2021. Empirical effort and schedule estimation models for agile processes in the US DoD. IEEE Trans. Softw. Eng..

Siriwardhana, Y., Porambage, P., Liyanage, M., Ylianttila, M., 2021. A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications and technical aspects. IEEE Commun. Surv. Tutor..

Swiderski, M., 2021. Jbpm - open source business automation toolkit. http://www.jbpm.org/.

Wang, J., Altintas, I., 2012. Early cloud experiences with the kepler scientific workflow system. Procedia Comput. Sci. 9, 1630–1634.

Wang, X., Ning, Z., Guo, S., 2020a. Multi-agent imitation learning for pervasive edge computing: a decentralized computation offloading algorithm. IEEE Trans. Parallel Distrib. Syst. 32 (2), 411–425.

Wang, X., Ning, Z., Guo, S., Wang, L., 2020b. Imitation learning enabled task scheduling for online vehicular edge computing. IEEE Trans. Mob. Comput..

Wang, L., Wang, K., Pan, C., Xu, W., Aslam, N., Nallanathan, A., 2021. Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing. IEEE Trans. Mob. Comput..

Wang, H., Xu, C., Guo, B., Ma, X., Lu, J., 2019. Generic adaptive scheduling for efficient context inconsistency detection. IEEE Trans. Softw. Eng..

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., 2013. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. Nucleic Acids Res. 41 (W1), W557–W561.

Xu, J., Li, X., Liu, X., Zhang, C., Fan, L., Gong, L., Li, J., 2019a. Mobility-aware workflow offloading and scheduling strategy for mobile edge computing. 184–199..

Xu, J., Liu, X., Li, X., Zhang, L., Yang, Y., 2019b. Express: An Energy-Efficient and Secure Framework for Mobile Edge Computing and Blockchain based Smart Systems** This research is in part supported by the National Natural Science Foundation of China Project No. 61972001. 1283–1286.

Xu, X., Zhang, X., Gao, H., Xue, Y., Qi, L., Dou, W., 2019. Become: Blockchain-enabled computation offloading for IoT in mobile edge computing. IEEE Trans. Ind. Inf. 16 (6), 4187–4195.

Yin, L., Dong, W., Liu, W., Wang, J., 2018. On scheduling constraint abstraction for multi-threaded program verification. IEEE Trans. Softw. Eng. 46 (5), 549–565.

Yu, Z., Gong, Y., Gong, S., Guo, Y., 2020. Joint task offloading and resource allocation in UAV-enabled mobile edge computing. IEEE Internet Things J. 7 (4), 3147–3159.

Yu, R., Li, P., 2021. Toward resource-efficient federated learning in mobile edge computing. IEEE Netw. 35 (1), 148–155.

Zhang, X., Zhang, J., Liu, Z., Cui, Q., Tao, X., Wang, S., 2020. Mdp-based task offloading for vehicular edge computing under certain and uncertain transition probabilities. IEEE Trans. Veh. Technol. 69 (3), 3296–3309.

Zhou, F., Hu, R.Q., Li, Z., Wang, Y., 2020. Mobile edge computing in unmanned aerial vehicle networks. IEEE Wirel. Commun. 27 (1), 140–146.

Jia Xu received his bachelor's, master's and Ph.D. degrees in computer science and technology from the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China, in 2010, 2017 and 2022, respectively.

He was a software engineer focusing on industrial projects and solutions in iFLYTEK Co., Ltm from 2017–2018. He is currently an Postdoctoral Researcher with the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China. His current research interests include mobile edge computing, workflow systems, cloud computing, and resource management.

Ran Ding received his bachelor's degree in School of Economics and Management , Anhui University of Science and Technology, Huainan, Anhui , China. He is currently pursuing a master degree with the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China. His current research interests include mobile edge computing, workflow systems, task scheduling.

Xiao Liu received his bachelor's and master's degrees in information management and information system from the School of Management, Hefei University of Technology, Hefei, China, in 2004 and 2007, respectively, and his Ph.D. degree in computer science and software engineering from the Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia, in 2011.

He was an associate professor at the Software Engineering Institute, East China Normal University, Shanghai, China during 2013 to 2015. He is currently an associate professor with the School of Information Technology, Deakin University, Melbourne. His current research interests include workflow systems, cloud and edge computing, big data analytics, and human-centric software engineering.

Xuejun Li received his Ph.D. degree in computer application technology from the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China, in 2008.

He is currently a full professor with the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China. His major research interests include mobile edge computing, workflow systems, cloud computing, and intelligent software.

John Grundy received the B.Sc. (Hons), M.Sc., and Ph.D. degrees in computer science from the University of Auckland, New Zealand. He is currently Australian Laureate Fellow and a professor of software engineering at Monash University, Melbourne, Australia. He is an associate editor of the IEEE Transactions on Software Engineering, the Automated Software Engineering Journal, and IEEE Software. His current interests include domain-specific visual languages, model-driven engineering, large-scale systems engineering, and software engineering education. More details about his research can be found at https://sites.google.com/site/johncgrundy/.

Yun Yang received a Ph.D. degree in computer science from the University of Queensland, Brisbane, QLD, Australia, in 1992. He is currently a full professor with Department of Computing Technologies, Swinburne University of Technology, Melbourne, VIC, Australia. His current research interests include software technologies, cloud and edge computing, workflow systems, and service computing. He is an associate editor of the IEEE Transactions on Parallel and Distributed Systems.