Computer Communications 181 (2022) 173–181

Contents lists available at ScienceDirect

Computer Communications

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

Big data-driven scheduling optimization algorithm for Cyber–Physical Systems based on a cloud platform

Chao Niu [a,*], Lizhou Wang [b]

[a] School of Information Management, Central China Normal University, Wuhan, Hubei 430079, China
[b] School of Big Data, Taiyuan University of Technology, Taiyuan, Shanxi 030024, China

ARTICLE INFO

Keywords:
Cloud platform
Big data-driven
Cyber–Physical Systems
Scheduling optimization algorithm

ABSTRACT

In this paper, we study big data-driven Cyber–Physical Systems (CPS) through cloud platforms and design scheduling optimization algorithms to improve the efficiency of the system. A task scheduling scheme for large-scale factory access under cloud–edge collaborative computing architecture is proposed. The method firstly merges the directed acyclic graphs on cloud-side servers and edge-side servers; secondly, divide the tasks using a critical path-based partitioning strategy to effectively improve the allocation accuracy; then achieves load balancing through reasonable processor allocation, and finally compares and analyses the proposed task scheduling algorithm through simulation experiments. The experimental system is thoroughly analysed, hierarchically designed, and modelled, simulated, and the experimental data analysed and compared with related methods. The experimental results prove the effectiveness and correctness of the worst-case execution time analysis method and the idea of big data-driven CPS proposed in this paper and show that big data knowledge can help improve the accuracy of worst-case execution time analysis. This paper implements a big data-driven scheduling optimization algorithm for Cyber–Physical Systems based on a cloud platform, which improves the accuracy and efficiency of the algorithm by about 15% compared to other related studies.

1. Introduction

Cyber–Physical System (CPS) is an interdisciplinary system that evolves from the previous disciplines and fields of embedded systems, hardware-software hybrid systems, and sensor networks, etc. CPS is a system that integrates computation, communication, and control [1]. The CPS uses sensors to sense the environment, and then communicates the data and signals collected by the sensors with computing devices through the network, which process the data and make decisions in real-time and in an integrated manner, and then sends the decisions or instructions to the actuators through the network for control [2]. With the development of fields and technologies such as Ubiquitous Interconnection, Ubiquitous Computing, Internet of things (IoT), and network technologies (e.g., 5G networks, wireless self-organizing networks), frequent interaction and close integration between the physical world and information space become possible [3].

CPS is a deep integration of the information world with the physical world through the above-mentioned real-time feedback loop [2], which has the properties of real-time, reliability, security, diversity, and autonomy [4]. scheduling, intelligent medical care, industrial advanced manufacturing and automation, and intelligent manufacturing fields, as well as the new emerging smart home, smart city, and smart agriculture in recent years, and very many other fields [5,6]. With the development of computer hardware and software and the advent of the information age and the popularity of information technology, more and more fields are using real-time systems, which are used to ensure that tasks can be completed within a given deadline by task scheduling, and here in real-time systems, the deadlines are divided into two types: one is a hard deadline and the other is the soft deadline [7]. Real-time systems with hard deadlines are called hard real-time systems, which are overly sensitive to time and have extremely strict requirements for real-time performance [8]. Real-time systems with soft deadlines are called soft real-time systems. Our research is the first time to use this method for research, which is more robust than existing research methods.

For soft real-time systems, if tasks cannot be completed within the specified time, it may cause degradation of system performance [9]. With the change in computing paradigm, i.e., the emergence of cloud computing and later the emerging fog computing and edge computing, and the increase in computing power, Big Data, which could not be handled by conventional methods, has become processable and applicable. Processing Big Data without using random analysis methods (e.g., sampling) makes the knowledge derived from Big Data processing more comprehensive, more accurate, and more extensive in coverage and applicability [10]. The contributions of our paper can be concluded as follows: First, we divide the tasks using a critical

* Corresponding author.
E-mail address: milk8825@163.com (C. Niu).

https://doi.org/10.1016/j.comcom.2021.10.020
Received 24 March 2021; Received in revised form 6 October 2021; Accepted 12 October 2021
Available online 19 October 2021

path-based partitioning strategy to effectively improve the allocation accuracy. Second, this paper compares and analyses the proposed task scheduling algorithm through simulation experiments. Third, this paper implements a big data-driven scheduling optimization algorithm for Cyber–Physical Systems based on a cloud platform, which improves the accuracy and efficiency of the algorithm by about 15% compared to other related studies.

It solves the high response delay of terminal devices due to a large amount of uploaded data. Data is the key enabler of smart manufacturing. In manufacturing, big data refers to a large amount of multi-source, heterogeneous data generated during the product lifecycle and collected through the Internet of Things or Radio-frequency identification (RFID) to enable real-time monitoring of equipment and product status. For example, built-in sensors can continuously measure, monitor, and report on the operational status of production equipment and products; RFID can automatically identify, track, and manage large volumes of artifacts and materials needed for production. In this paper, a cloud–edge collaborative computing architecture is adopted to meet the realistic requirements of large-scale data collection in cloud manufacturing scenarios. The rest of the sections of this paper can be summarized as follows: Section 2 contributes discuss the related works; Section 3 devotes to scheduling optimization algorithms for big bata-driven Cyber–Physical System design for cloud platforms; In Section 4, we analysed the system performance index and optimized scheduling results; Section 5, we summarize and analyse the results.

2. Related work

Fallahpour et al. proposed a task offloading scheme based on mobile edge computing [11]. The scheme achieves resource allocation based on the amount of energy remaining in the mobile device. Eggensperger et al. proposed an autonomous control scheme for Remote Split Operations (UAVs) for mobile edge computing to address the more stringent performance requirements in terms of system latency and jitter due to the increase in heterogeneous IoT application scenarios [12]. UAVs are increasingly used in industrial application scenarios, such as laying and maintaining cables in hard-to-reach areas using UAVs [13]. In recent years, artificial intelligence has made extensive developments in manufacturing, transportation, finance, and medicine. However, with the deep penetration of machine learning and deep learning technologies in key application areas such as cloud manufacturing and unmanned vehicles, scholars at home and abroad explore advanced cloud-side collaboration methods to ensure real-time, accurate, and efficient interactions. Ghadimi et al. proposed a cloud-side collaboration task using the optimal residual space ratio to reduce the scheduling problem to a linear programming problem using a trade-off model [14]. Yu et al. proposed a Dynamic Tasks Scheduling algorithm based on the Weighted Bi-graph model (DTSWB) that transforms the scheduling problem into an optimal bi-graph matching problem and establishes an integer programming model [15].

The model consists of four main steps: task offloading, state information collection of service providers, the establishment of mapping relationships, and optimal matching of profit matrices, to finally achieve efficient dynamic scheduling of edge-side tasks [16]. The above solution is often only applicable to solving integer linear programming problems, which require all or some of the decision variables to be assumed to be non-negative integers, but in practical industrial settings, the solution is too idealistic and the integer solution obtained by rounding is usually inexact [17]. To address the above algorithm's limitations, Kong et al. proposed an improved chaotic bat algorithm that solves the traditional cloud computing scheduling algorithm's problem of balancing between task complexity and system performance and improves the update speed of dynamic parameters in the system by introducing a chaos factor and a second-order oscillation mechanism [18].

This paper proposes a resource allocation method for IO intensive virtual machines based on edge-end devices and a task scheduling method for large-scale factory access based on the cloud–edge collaborative computing architecture to meet the data characteristics of multiple heterogeneous sources in the cloud manufacturing scenario. In this paper, we propose a resource allocation method for CPU-dense virtual machines based on edge-end devices and decouple storage and computation to reduce the interference between services, to achieve efficient services. Task scheduling scheme for large-scale factory access under cloud–edge collaborative computing architecture. A task scheduling method for large-scale factory access under cloud–edge collaborative computing architecture is given for industrial application scenarios. The task scheduling problem in complex scenarios is solved by three steps: multi-task graph merging, task graph partitioning, and processor scheduling.

3. Scheduling optimization algorithms for big data-driven Cyber–Physical System design for cloud platforms

3.1. A formal problem definition

With the advent of the era of big data and the maturity of cloud computing technology, the analysis and processing of big data increasingly rely on cloud computing platform. To analyse and process distributed big data using cloud computing platform, the first problem to be considered is how to dispatch distributed big data to the appropriate data centre of cloud computing platform. The distributed big data scheduling problem provides cloud service providers with reasonable scheduling strategies, which is of great significance to reduce the cost of cloud service providers and improve their service quality. How to reduce the cost for different user job models and data centre cost heterogeneity is a problem to be solved. In this section, we focus on two things. Firstly, we design an improved scheduling optimization algorithm; Secondly, we design big data driven Cyber–physical systems for cloud platforms.

3.2. Improved scheduling optimization algorithm design

Cloud computing is a type of distributed computing, which refers to the process of decomposing huge data computing processing programs into countless small programs through the network "cloud", and then processing and analysing these through a system composed of multiple servers. The applet gets the result and returns it to the user. When the algorithm is initialized, the initial solution of the problem is generated randomly. After selecting the initial value, the cost function of the problem is evaluated and then the initial value is fine-tuned to generate a new candidate solution from around the initial value. After selecting a new candidate solution, the value of the cost function is calculated and retained if the value is better than the previous candidate solution. However, if the value is worse than any other candidate solution, then the search process moves to the next candidate solution with a small probability and continues the comparison [19]. In the cloud manufacturing scenario, for different plant equipment and actual business needs, the size of the data volume sent by the end devices deployed in each plant is different, so it is necessary to design a design solution for the edge-side servers equipped in different plants, to effectively reduce the enterprise's procurement funds and avoid the waste of limited resources. By adopting the cloud-side collaborative computing architecture, various sensors are installed on the mechanical equipment in the large manufacturing production plant, which is sent to the edge server by the terminal equipment, and the edge server sends the data to the cloud server at the same time, and finally, the business processing of the large manufacturing production plant is realized by the cloud-side collaborative computing. Take fluid equipment as an example, by installing temperature, pressure, vibration, and other sensors on the equipment, the PLC device sends the data to the IO-intensive virtual machine of the edge-side server, and the IO-intensive virtual machine sends it to the cloud server through the Internet. Since most

of the important parameters of mechanical equipment do not change frequently, and the mechanical characteristics of mechanical equipment determine that most of the parameters will remain in a stable state for changes. In the case of the piston of a fluid device, for example, its state and the size of the outlet pressure remain strongly correlated in most cases, and the size of the outlet pressure gradually climbs from a safe initial value to a stable value and changes steadily according to the business needs of the device, as shown in Fig. 1.

In this section, the priority P of each virtual machine is obtained by the information of three dimensions of IO-intensive virtual machines, assuming the number of existing end devices t, the data type uploaded by an end device, n denotes the number of end device upload types, the data size corresponding to its upload data type is a, its corresponding sampling time is s, and the data collection interval is M. To meet the service requirements of both terminal devices and edge-end clusters, this chapter proposes a dynamic resource allocation scheme based on the second-order difference heuristic algorithm (SDH), assuming that the storage rate of the edge-end cluster is a, the resources of the host and the resources of its internal virtual machines are satisfied as:

$$\phi_t = \frac{\lim_{T\to\infty}\sum_{t=1}^{T} m_t}{M+1}, \forall t \in \{1,2,\ldots,T_a\} \tag{1}$$

This section further assumes that the available bandwidth of the system is W and that each cellular user is allocated a separate orthogonal sub-channel and M cellular users are allocated equal blocks of bands. To make the scenario more realistic, it is assumed that each M can request multiple content downloads based on its available storage capacity and actual demand [20]. Considering that each T has a different priority for the requested content, it is further assumed that the required content will be requested sequentially at each M and only one container will be downloaded at a time. Also, the CP can pre-cache popular content and deliver one content per content share.

$$\sum_{t=1}^{T_a}\phi_t = \frac{\lim_{T\to\infty} E\sum_{t=1}^{T} m_t}{M+1} \tag{2}$$

$$P_r P_t = h^i(\overline{P}) = \begin{cases} M+1-\phi_{i-1}\sum_{t=1}^{T} m_t \sum_{i=1}^{I-1}\phi_{i-k}, i=1,2,\ldots,t \\ 1-\phi_{i-1}, i=0 \end{cases} \tag{3}$$

As stated above, successful delivery of content requires consideration of both transmission rate and connection duration. Specifically, the connection duration can be expressed as the period between the establishment of the link and its departure from each other's transmission range. Therefore, this section adopts connection duration as an important measure of link quality, expressed in terms of IE. More specifically, if the connection duration obeys an exponential distribution with a parameter. And the different interaction processes are independently identically distributed, the probability density function can be expressed as:

$$IE\left[P_t\right] = \overline{P} + \lim_{\substack{T\to\infty \\ I\to\infty}}\sum_{t=1}^{T} m_t \sum_{i=1}^{I-1}\phi_{i-k}\prod(h^j(\overline{P})) \tag{4}$$

To achieve successful delivery of content, the connection duration needs to be longer than the actual transmission time M. Correspondingly, the probability of successful link establishment during transmission from J to I can be expressed as:

$$\begin{aligned} J &= \frac{M}{T}\lim_{\substack{T\to\infty \\ I\to\infty}}\sum_{t=1}^{T} m_t \sum_{i=1}^{I-1}\phi_{i-k}\prod(h^j(\overline{P})) - \sum_{i=1}^{I-1}\phi_{i-k}\prod(h^j(\overline{P})) \\ &= Tr\left|\overline{P}\right| + \frac{M}{T}\lim_{\substack{T\to\infty \\ I\to\infty}}\sum_{t=1}^{T} m_t \sum_{i=1}^{J-1}\phi_{i-k}\prod(h^j(\overline{P})) + \sum_{i=1}^{I-1}\phi_{i-k}\prod(h^j(\overline{P})) \\ &= Tr\left|\overline{P}\right| + \frac{M}{T}\lim_{\substack{T\to\infty \\ I\to\infty}}\sum_{t=1}^{T} m_t \sum_{i=1}^{J-1}\phi_{i-k}\prod(h^j(\overline{P}))\prod(h^j(\overline{P})) \end{aligned} \tag{5}$$

The successful establishment of a link in this section is a crucial factor in measuring successful delivery. Also, a transmission that does not meet the QoS requirements does not meet the requirements. For a more accurate description, this section further uses the probability of satisfying QoS as a measure of successful delivery. Specifically, this section uses a logistic function to describe this probability [21]. As the actual data rate exceeds the required QoS requirement for the corresponding content delivery, the logistic function converges more closely to 1 as the transmission rate increases and can be expressed as:

$$\phi_k = \frac{M}{T}\lim_{T\to\infty}\sum_{t=1}^{T} m_t \tag{6}$$

The reliability and stability of the link are judged by assuming whether the SDP is greater than a threshold value and whether it meets the QoS requirements. Unstable and reliable links can cause download failures. Therefore, the optimization problem in this chapter can be modelled as:

$$\frac{df(\nabla 0,k,t)}{d\nabla 0,k,t} = \beta\alpha t - \sum_i\sum_j \beta\alpha t \tag{7}$$

$$\Delta^{(m)} = \alpha_{0,m}, T_{0,m,t} - \sum_i\sum_j \alpha_{0,m}, T_{0,m,t} \tag{8}$$

$$\Delta^{(k)} = \delta_{0,m}, T_{0,m,t} - \sum_i\sum_j \delta_{0,m}, T_{0,m,t} \tag{9}$$

$$w_{0,k,t} = \frac{h^U_{0,k,t}}{\left\|h_{0,k,t}\right\|^2\cdot\beta\alpha t} \tag{10}$$

$$R_1 = H\cdot S_1 + N\sum_i\sum_j \delta_{0,m}, T_{0,m,t} \tag{11}$$

$$R_1' = H\cdot S_1' + N'\sum_i\sum_j \delta'_{0,m}, T'_{0,m,t} \tag{12}$$

For biased load attacks in distributed smart grids, this chapter proposes an attack detection and separation method based on unknown input observers. First, the distributed physical dynamic grid model is established by considering the influence of interconnection information among multiple generators on generator frequency, and then the attack detection algorithm based on the unknown input observer is proposed. The accuracy of internal physical dynamic estimation can be improved by the designed unknown input observer dealing with interconnection relations between grid neighbours and external disturbances [22]. Further, distributed adaptive thresholds are calculated to replace the traditional a priori thresholds by considering model uncertainty and external disturbances. Also, considering the topology of the distributed smart grid, a separation algorithm based on the attack feature matrix is further proposed. Notably, the proposed distributed separation method is based on two steps consisting of local and global attack separation. Combining the local and global attack sets allows the separation of the actuator attack sets that suffer from biased load attacks. Finally, the performance of the proposed detection algorithm is simulated and verified on an IEEE 8-bus smart grid system, and the effectiveness of the distributed detection and separation algorithm is simulated and verified with a large-scale IEEE 126-bus smart grid example. The information world is a double helix structure, the outer ring is the quality circle PDCA (Deming ring), and the inner ring is the 4 major types of information entities for daily production work. The outer ring is the activity, the inner ring is the product, and the mathematical model behind it is the PetriNet graph. The outer ring produces the inner ring, and the inner ring represents the production of the physical world. A self-consistent and self-evolving system is formed.

The perturbation risk vector starts from the vector perspective and elaborates the perturbation risk mathematical model, which transforms the single value of the resolution difficulty into a three-dimensional vector, which can reflect the different bias degrees of the perturbation and more intuitively represent the distribution of different perturbations; the perturbation risk vector size Radio Access Network (RAN), which is the square root of the perturbation vector mode, directly reflects the numerical size of the perturbation vector and provides a

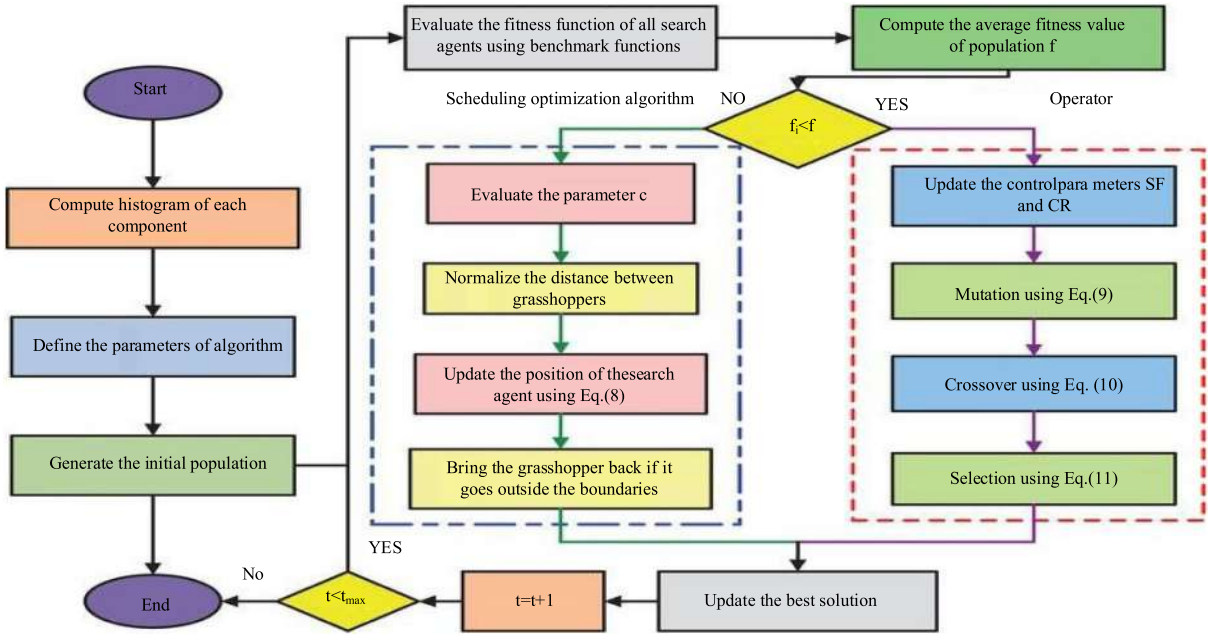

Fig. 1. Scheduling optimization algorithm framework.

Table 1
Repeatability comparison of perturbation risk size.

Disturbance risk assessment	Numerical number	Numerical ratio	Number of repeated solutions	Repetition rate
Project revenue note (PRN)	125	0.15	447	0.25
paraventricular nucleus of hypothalamus (RVN)	127	0.47	1456	0.174
PRN-1	425	0.41	1427	0.72
RVN-1	147	0.54	145	0.47

standard for quantitative analysis of the perturbation; the risk vector priority size of the bias vector r on the plane A is defined, which reflects the different bias degree size of the perturbation risk vector priority and provides a reference for further analysis of the perturbation risk priority. The risk vector priority size of the bias vector r on plane A is defined, which reflects the different bias degree size of the perturbation risk vector priority and provides a reference for further analysis of the perturbation risk priority. Our algorithm is more complex than other algorithms.

Meanwhile, the new perturbation vector method can calculate more risk prioritization results, traversing the values of parameters S. From the mathematical expressions, different perturbation risk evaluation values can be calculated, as shown in Table 1 for comparison [23]. Where the number of values refers to the number of different numerical results that can be generated by the perturbed risk model, the value ratio refers to the ratio of the number of different values generated by the model to the total number of 1000 combinations, the number of repetitions of the solutions in the number of repeated solutions indices, and the repetition rate is the ratio of the number of repeated solutions to the number of values. The results show that the new mathematical model can obtain more different solutions in the numerical results, achieving the differentiation of different perturbations and providing a more continuous set of solutions to explain different perturbations.

In the high-level analysis stage, the method performs the Front-End step on the information layer middleware program corresponding to the CPS task and derives its control flow graph for performing control flow analysis, which analyses the control flow of the program to determine the execution frequency of the paths and the timing information of the control flow nodes, etc., and annotates them on the control flow graph for subsequent stages of analysis. Also, the method proposes to process and analyse the big data related to the CPS task input set to derive big data knowledge for narrowing down the task input set, and the optimized task input set for control flow analysis and physical layer analysis can help reduce the analysis complexity and improve the analysis accuracy.

3.3. Big data-driven Cyber–Physical System design for cloud platform

The information layer code-level timing analysis is the first high-level analysis phase of the analysis methodology proposed in this paper, and its results are used as input for the subsequent phases. The information layer code-level timing analysis attempts to determine the boundaries of the task execution time when executing the information layer tasks (in this case, the programs or code segments processed by the information layer, noting the distinction with the concept of CPS tasks, which refer to the overall system tasks and the programs or code segments processed by the information layer as a component of the CPS tasks) on the information layer hardware platform. The time for a particular execution depends on the path of the task taken by the control and the time spent on this hardware for the statements or instructions on this path. Therefore, determining execution time bounds must consider all potential (possible run-to) control flow paths and the execution time of this set of paths. The modular approach to the timing analysis problem divides the entire task into a series of subtasks (subroutines or smaller code segments), which are called Basic Blocks if they are not divided further down. They are used to deal with the properties of the control flow, while others deal with the execution time of instructions or instruction sequences on a given hardware.

The flow of the information level code-level timing analysis is given in Fig. 2. The flow except for the dotted line is the general analysis flow. These steps can help improve the accuracy of the timing analysis, but they require specific hardware platforms and detailed information about the parameters and properties of this hardware; if the dashed steps are not used, they can be replaced by conservative estimates, and general results can be obtained.

Based on in-depth analysis of actual problems, the relevant factors are decomposed into several levels from top to bottom according to different attributes. The factors of the same level are subordinate to the factors of the upper level or have an influence on the factors of the upper level, and at the same time dominate the next level. Level

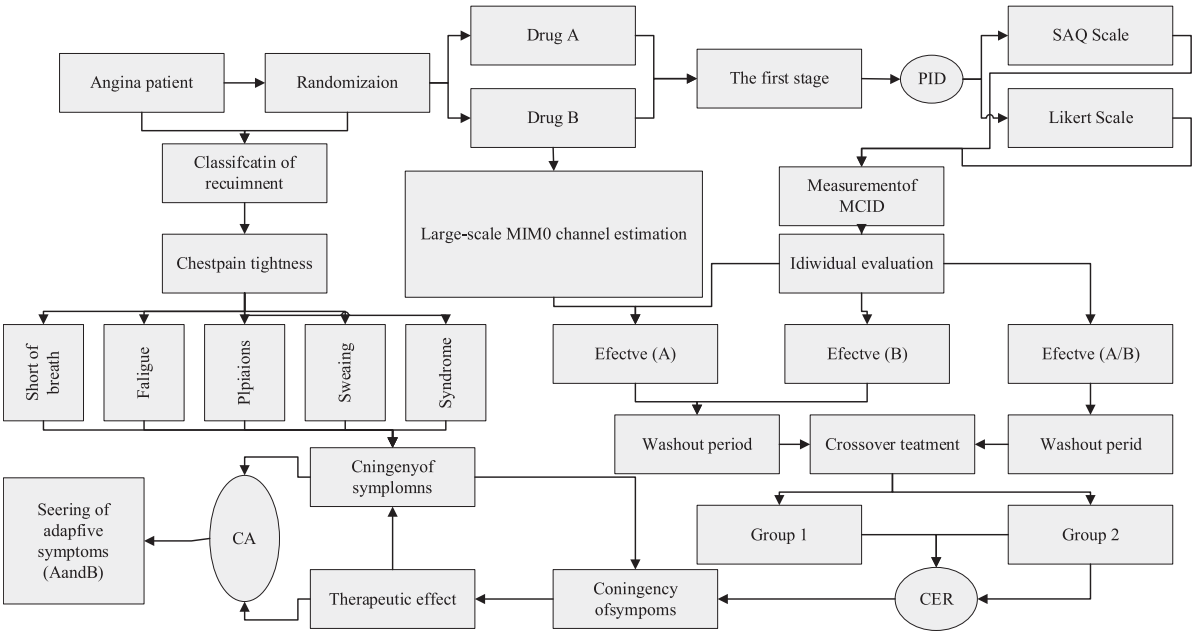

Fig. 2. Information layer code level analysis flow.

factors may be affected by underlying factors. The uppermost layer is the target layer, usually only one factor, and the lowermost layer is usually the scheme or object layer, and there can be one or several layers in between, usually the criterion or indicator layer. When there are too many criteria (for example, more than 9), the sub-criteria layer should be further decomposed. The information layer task to be analysed obtains its worst-case execution time (WCET) on one (or sometimes several) of its execution paths. if the input and initial state leading to the execution of this worst-case path is known, the problem is easily solved by simply starting the task in this initial state using this input and measuring the execution time. However, in general, such worst-case inputs and initial states are unknown and difficult or impossible to determine. A phase called control flow analysis (CFA) determines information about control flows through the task to improve the accuracy of subsequent analyses. The purpose of CFA is to gather information about the possible execution paths. The set of paths is always limited because the task must be guaranteed to terminate [24]. Often it is difficult to determine a specific set of paths, and any superset of that set of paths is a safe approximation, the smaller the superset the better. If the execution times of some paths are negligible in the computation of the upper bound, it is safe to exclude them, since these excluded paths do not contribute to the computation of the upper bound.

It is worth noting that a complex instruction processor may execute the instruction stream in a different order than that determined by the CFA. This is due to its pipelining techniques (prefetching and delayed branching), branch prediction, and speculative or out-of-order execution, all concepts considered in Low-Level Analysis, which do not affect the safety of the analysis, but do affect the accuracy of the analysis results.

Processor behaviour analysis requires an architecture model that need not be a concrete model that implements all the functions of the target hardware; a simplified model that is conservative concerning timing behaviour is sufficient in WCET analysis. Such an abstract processor model is either part of the engine used for processor behaviour analysis or is an input to the construction of such an engine. In any case, the construction of the abstract processor model is done at the time of tool construction.

All approaches based on hardware architecture models have an inherent limitation: they rely on the timing accuracy of the model. In general, computer vendors do not disclose enough information about the microarchitecture, so it becomes necessary to develop and securely verify the accuracy of the timing model. Without such verification, any WCET tool based on a hardware abstraction model cannot be trusted without further guarantees. Without such verification, other model verification methods must be adopted, for example, the measurement can do this: comparing the measured execution time with the predicted bounds. Another approach is the trace validation method, which checks whether the external observable traces are projections of the traces predicted by the model. Not all events predicted by the model are externally observable, but both methods are like tests; they can find the location of the presence of errors (if any), but cannot prove that no errors exist. Stronger assurance can be provided by checking for equivalence between various levels of abstraction. A related research activity currently underway is the formal derivation (Formal Derivation) of an abstract processor model from a concrete model, as shown in Fig. 3. The accuracy of our results and the results of other related studies has increased by about 15%.

A typical processor contains factors that make execution time context-dependent, such as memory, cache, pipeline, and branch prediction. The execution time of individual instruction, or even its need to access memory, depends on the execution history. To find bounds on the exact execution time for a given task, it is necessary to analyse the occupancy status of these processor components for all paths leading to the task's instructions. Processor behaviour analysis determines the non-variance of these occupancy states for a given task. In principle, a complete tool should take into account the peripherals of the processor (e.g. the complete memory hierarchy, the bus, and the rest of the peripheral units). So far, a better term is hardware subsystem behaviour analysis, which is done on a linked executable file, since only this file contains all the necessary information. Hardware subsystem behaviour analysis is based on an abstract model of processors, memory subsystems, buses, and peripherals that is conservative concerning the timing behaviour of specific hardware, i.e., its predicted execution time is never shorter than the execution time observable on a specific processor.

For the deployment requirements of industrial IoT in a smart manufacturing environment, the cloud–edge collaborative computing architecture mainly includes device sensing layer, data resource layer, service layer, application layer, and operation, and maintenance management layer which is responsible for the maintenance of the whole

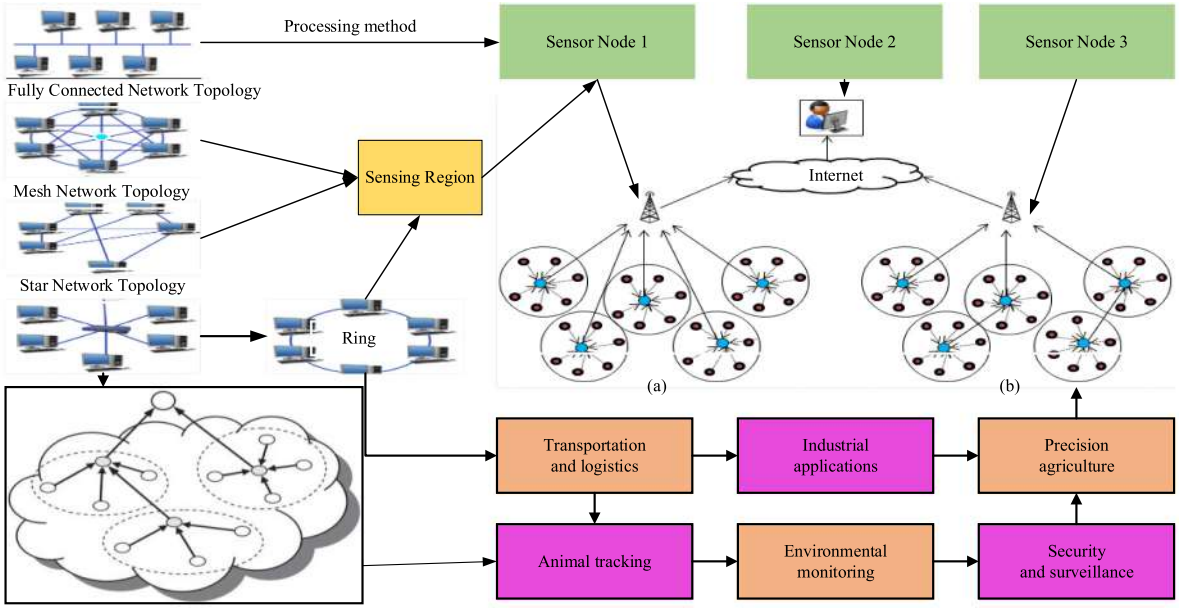

Fig. 3. Cyber physical system framework.

data link. In terms of specific operations, the cloud component is mainly responsible for model training of the collected data, and the edge component is mainly responsible for providing real-time services for factory equipment by acquiring models in the data dictionary [24]. OpenStack and Starling are currently the most widely used open-source cloud computing platforms and the latest distributed edge computing platforms respectively, which can help enterprises build their cloud computing services and edge computing services.

The edge-side server synchronizes incremental data to the remote centralized data storage module [25,26]. The data processing module gets the corresponding data from the remote centralized data storage module according to the user's requirements; the data processing module performs weighty big data analysis according to the model parameters provided by the data dictionary module and synchronizes them to the remote data dictionary module; the remote data dictionary module will synchronize the data with the edge-side data dictionary module according to the specific requirements [27,28]. The edge-side server and the remote centralized server will periodically analyse and mine the stored data to update the data dictionary to ensure the accuracy of the decision messages [29].

4. Analysis of results

4.1. System performance index analysis results

The Flume architecture is distributed and can scale Flume nodes and numbers as needed. Sqoop is a bridge between relational databases and HDFS, allowing data to be transferred between relational databases and HDFS. Sqoop is a bridge between relational databases and HDFS, enabling data transfer between relational databases and HDFS. To enable the cloud platform to support data entry services for multiple data sources, the cloud data collection subsystem deploys Flume, Sqoop component, Spring-boot Opus data collection server, and Albacore server respectively. The flume component can realize data entry services for network port number data, logs, and other text data, and it supports Incremental data import. The Sqoop component enables the interconversion of traditional relational database data with HUFFS and HBase data to ensure efficient and secure data import and export operations in different systems. Abs Core server is the basis for the visualization and management of the cloud platform. This part of the work is described below. The overall data monitoring diagram is

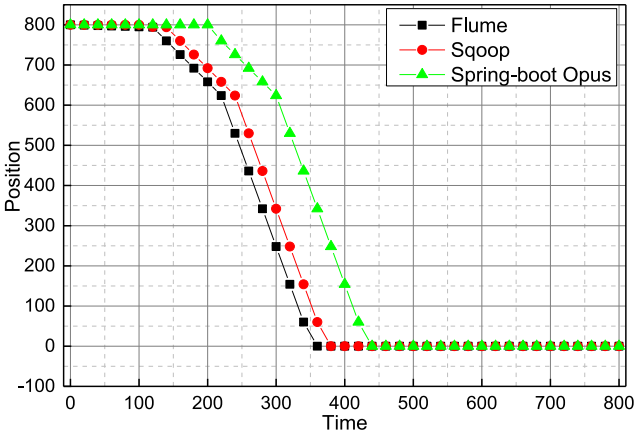

Fig. 4. Real-time monitoring curve.

shown in Fig. 4. This page includes real-time equipment data such as over-controlled tank water level, boiler temperature, fan pressure, and double-capacity tank water level. Managers can understand the real-time production data of current equipment through this page and realize the unified monitoring of different on-site equipment based on the cloud.

This interface allows you to obtain real-time information on whether abnormal faults occur in the current equipment. When the range of data change of different monitoring items set in the remote data server exceeds the alarm threshold, the remote data server will send an alarm message to the current monitoring management interface. When the monitoring management interface receives the alarm information, the right side of the page will display the current specific alarm information and the page will sound the alarm at the same time. The administrator can customize the alarm thresholds of different concern items through this function. In this interface, the alarm thresholds of the three monitoring items can be customized according to the actual needs.

The simulation experiment intends to simulate five scenarios, each specific scenario represents the scenario of a different external environment in which the melding processing room is located, and each scenario corresponds to a random seed, and each specific random seed generates a specific random sequence with reproducibility. The

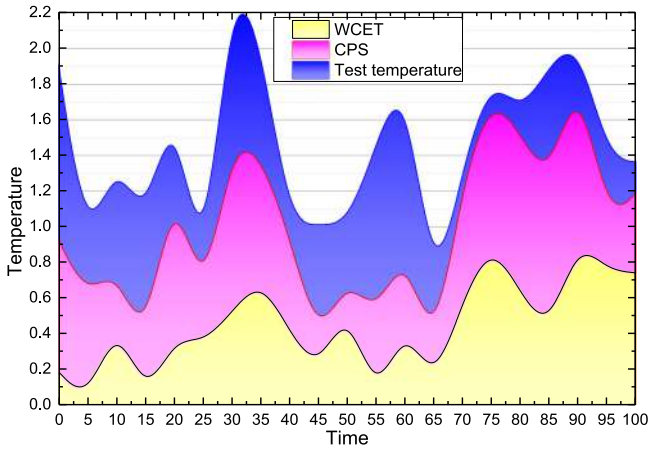

Fig. 5. Big data knowledge analysis simulation.

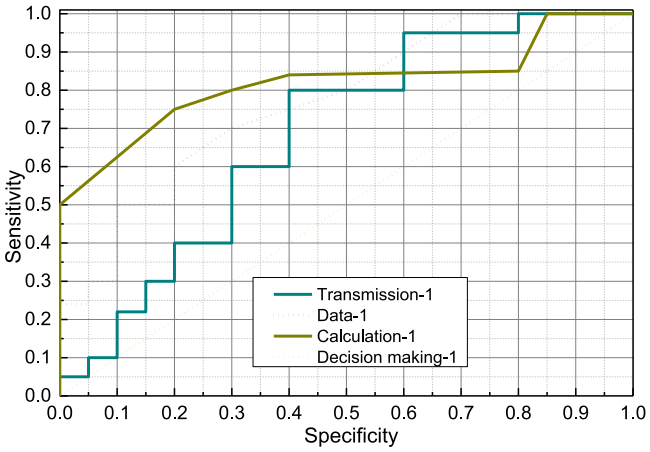

Fig. 6. Experimental results data.

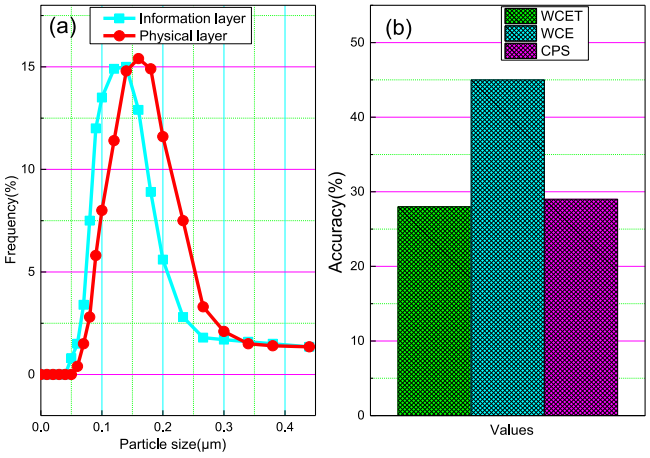

Fig. 7. Comparative analysis of experimental results data.

experiment first simulates the process of big data knowledge analysis and records the lowest temperature of the scene derived from big data analysis under a certain scenario. The role of the lowest temperature is that it is the input range boundary of WCET analysis, and the non-trivial WCET estimate can be obtained accordingly; relatively, the temperature range of an industrial-grade product is 0 °C~2.2 °C, and this experiment takes its lowest temperature −2.0 °C as the ordinary lower boundary of the operating temperature, it is assumed that the operation will not start below this temperature, and the WCET estimate is obtained based on the ordinary lower boundary of the operating temperature, as shown in Fig. 5. The sampling time of the external temperature is set to 0 so that every 100-time points are used to simulate the big data of the external ambient temperature for different samples of the scenario. It should be noted that the sampling time is only used to simulate the big data knowledge analysis simulation, and in a certain warming task of the temperature control system, the external temperature at the time of the task is treated as a constant due to the short task time. After the big data knowledge analysis simulation, a simulation of the CPS warming task is performed for each specific scenario, and the external temperature is treated as a constant during the simulation, depending on the random seed corresponding to the scenario.

In the simulation of a particular scenario, the initial room temperature of the melding process room is set to be the same as the outside temperature, and the CPS warming task starts from time point 0 and goes through the feedback loop process of sensor sampling, network transmission, data analysis, calculation, decision making, decision execution, and action on the physical world, as shown in Fig. 6.

Using the analysis method proposed in this paper, we can obtain the execution time and corresponding WCET estimates of the information layer and the physical layer of the CPS task in a more comprehensive way; however, if we use the traditional WCET analysis method for real-time systems, we can only obtain the execution time and WCET estimates of the information layer but not the execution time and corresponding WCET estimates of the physical layer, so we cannot calculate the total in practice, the execution time of the physical layer accounts for a larger portion of the CPS task execution time than the information layer (because the physical layer interaction needs to follow the laws of nature and physics, while the information layer is faster in computation), and if the execution time of the physical layer is not considered, the analysis result may be lower than the actual execution need and thus unsafe. In each scenario of the experiment, the information layer execution time and physical layer execution time do not exceed the corresponding information layer WCET estimates, and the total physical layer WCE execution time is generally about 0% to 16% richer than the total WCET estimates, which indicates that the WCET analysis framework and method of CPS proposed in this paper are safe and effective. The actual execution time still has a large margin from the expected worst-case WCET estimate, as shown in Fig. 7.

Without the idea and method of using big data knowledge to help WCET analysis proposed in this paper, it would lead to a conservative and mundane WCET valuation using a mundane task input set; the use of big data knowledge to help WCET analysis proposed in this paper corrects the mundane WCET valuation by correcting the lowest temperature value of the scene in the experiment, thus improving the accuracy by about 28%~45%. Forty-one percent accuracy, which shows that this idea can effectively help to improve the accuracy of WCET analysis. The results of the experiments are analysed, and the results are compared with those of related methods, which prove the effectiveness of the WCET analysis method of CPS proposed in this paper. It also shows that the idea of using big data knowledge to help WCET analysis of CPS proposed in this paper can effectively help improve the accuracy of WCET analysis.

4.2. Analysis of optimization scheduling results

Increasing the number of edge servers has a significant impact on computing time overhead, and as the number of edge-side servers they deploy increases, the Carpal Tunnel Syndrome (CTS) algorithm and Directed Acyclic Graph (DAG) algorithm have good advantages over the traditional centralized traversal approach. Fig. 8 shows the impact of the increase of edge on computing time overhead. In Fig. 8, as time

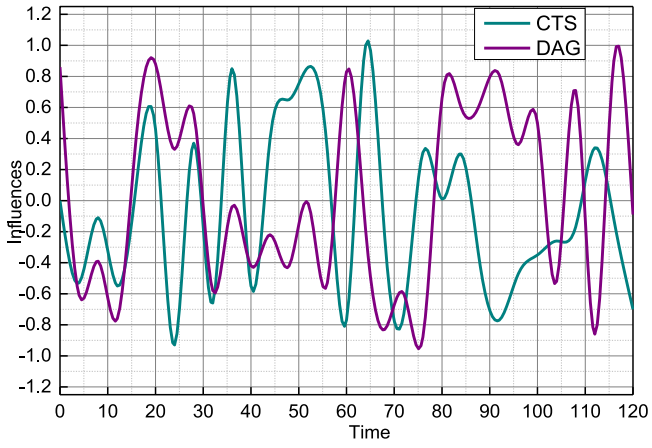

Fig. 8. The impact of the increase of edge on computing time overhead.

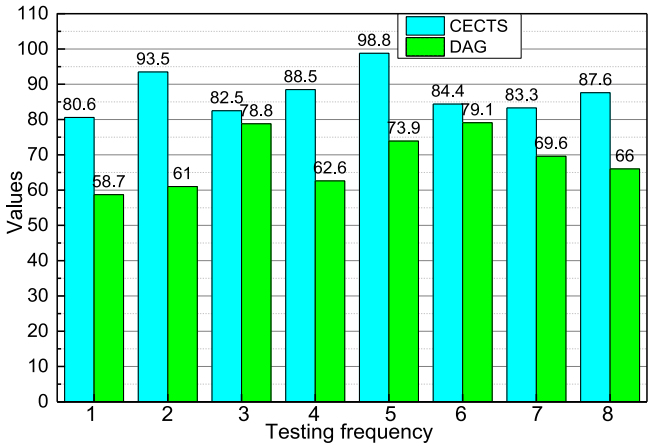

Fig. 10. Degree of the fairness of CECTS algorithm and DAG algorithm.

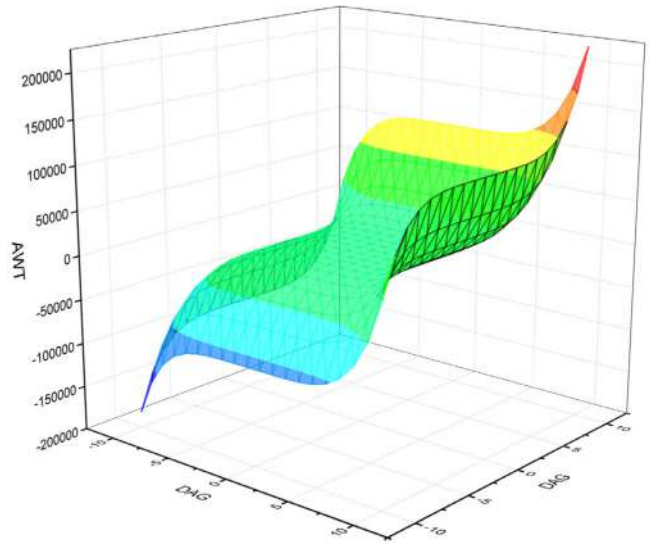

Fig. 9. Slack value of the algorithm.

increases, the corresponding results are not linear. This also means that some of these features are not relevant.

It is the input terminal of the module, used by the MCU to notify the module, whether the MCU is ready, whether the module can send information to the MCU, and the effective level of RTS is low. CTS is the input signal, used to judge whether it can send data to the other party, low level is effective, low level indicates that the device can send data to the other party. Three algorithms are used to schedule each of the 10 DAG task graphs, and Fig. 9 shows the comparison of the Slack values of the three algorithms for the 3-core processor case. The three algorithms are used to schedule the DAG tasks separately for the different number of DAG task models, and the span makes span values, average wait time AWT and average Slack values are obtained for all tasks, as shown in Fig. 9.

From the experimental data, it is concluded that in terms of both the span of task scheduling and the average waiting time of tasks, there is a corresponding increase in the number of DAG. Overall, the DAG algorithm has the best performance, and the Sequential algorithm has the worst performance. From the experimental data, the DAG algorithm reduces the task scheduling span by a minimum of 17% and a maximum of 28% compared to the DAG algorithm, and the DAG algorithm reduces the average task waiting time by a minimum of 74% and a maximum of 80% compared to the Sequential algorithm. In terms of average task wait time, the DAG algorithm reduces the average task wait time by a minimum of 47% and a maximum of 54%. The DAG algorithm has a minimum reduction of 84% and a maximum reduction of 93% over the Sequential algorithm. Comparing our work with the performance of other existing tools/models, we can find that the efficiency and accuracy of our models have been improved.

The fairness of the scheduling algorithm is an indication of how reliable the multiple DAG task scheduling algorithm is, and is an important indicator of how fairly the algorithm can handle the demands of tasks of different priority levels. Fig. 10 illustrates the degree of fairness of the CECTS algorithm and the DAG algorithm.

The new industrial model led by smart manufacturing starts to be widely used. In this chapter, a task scheduling method for large-scale factory access under the cloud–edge collaborative computing architecture is given for industrial application scenarios, and the task scheduling problem in complex scenarios is solved by three steps: multi-task graph merging, task graph partitioning, and processor scheduling. Simulation results show that compared with other scheduling algorithms, the algorithm in this paper can reduce the time overhead of processing redundant tasks and schedule the tasks to the best processor under cloud–edge collaborative computing, thus improving the speed of task processing. The tasks can be processed more efficiently in the case of resource-constrained large-scale edge-side servers.

5. Conclusion

In this paper, we propose a resource allocation and task scheduling scheme for large-scale factory access under cloud–edge collaborative computing architecture, describe the structure and workflow of each scheme, and propose a second-order difference heuristic algorithm for IO-intensive virtual machines; an optimal virtual machine performance algorithm for CPU-intensive virtual machines; and a CECTS algorithm for task scheduling. The unbalanced load of resources and tasks in the production system will not match the actual demand of low latency and high responsiveness in industrial sites. In this paper, we propose a resource allocation and task scheduling scheme for large-scale factory access under cloud–edge collaborative computing architecture and application scenarios in electrical processing, tire manufacturing, and coal mining. Finally, the proposed scheme is implemented in simulation and performance analysis of the entire system by using common open-source software in the industry. We implemented a big data-driven cyber–physical system scheduling optimization algorithm based on the cloud platform. Compared with other related research, the accuracy and efficiency of the algorithm have increased by about 15%. At the end of this paper, the experimental results of the proposed scheme are compiled separately and combined with the theoretical analysis, it is verified that the resource allocation and task scheduling scheme for

large-scale factory access under the cloud-side collaborative computing architecture can well solve the current problems of load imbalance encountered by industrial enterprises in the process of intelligent transformation, bringing significant improvement in latency performance, and the experimental results match the test results. In the future, we will work to use the Internet of Things to become a possible use case. In the future, we will further improve the accuracy and efficiency of the algorithm on this basis.

CRediT authorship contribution statement

Chao Niu: Formal analysis, Methodology. **Lizhou Wang:** Wrote the manuscript.

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.

References

[1] M. Chen, J. Yang, J. Zhou, et al., 5G-smart diabetes: Toward personalized diabetes diagnosis with healthcare big data clouds, IEEE Commun. Mag. 56 (4) (2018) 16–23.
[2] J. Wang, A. Liu, T. Yan, et al., A resource allocation model based on double-sided combinational auctions for transparent computing, Peer-to-Peer Netw. Appl. 11 (4) (2018) 679–696.
[3] S. Amaran, N.V. Sahinidis, B. Sharda, et al., Simulation optimization: a review of algorithms and applications, Ann. Oper. Res. 240 (1) (2016) 351–380.
[4] W. Wang, L. Zuo, X. Xu, A learning-based multi-RRT approach for robot path planning in narrow passages, J. Intell. Robot. Syst. 90 (1–2) (2018) 81–100.
[5] Z. Li, S.S. Zhong, L. Lin, Novel gas turbine fault diagnosis method based on performance deviation model, J. Propul. Power 33 (3) (2017) 730–739.
[6] M.C. Chen, S.Q. Lu, Q.L. Liu, Uniqueness of weak solutions to a Keller–Segel–Navier–Stokes system, Appl. Math. Lett. 121 (2021) 107417.
[7] X. Xu, Machine Tool 4.0 for the new era of manufacturing, Int. J. Adv. Manuf. Technol. 92 (5–8) (2017) 1893–1900.
[8] S. Kounev, N. Huber, F. Brosig, et al., A model-based approach to designing self-aware it systems and infrastructures, Computer 49 (7) (2016) 53–61.
[9] H. Xiang, Y. Li, H. Liao, et al., An adaptive surrogate model based on support vector regression and its application to the optimization of railway wind barriers, Struct. Multidiscip. Optim. 55 (2) (2017) 701–713.
[10] G. Chen, E. Wang, X. Sun, et al., An intelligent approval system for city construction based on cloud computing and big data, Int. J. Grid High Perform. Comput. (IJGHPC) 8 (3) (2016) 57–69.
[11] A. Fallahpour, A. Amindoust, J. Antuchevičienė, et al., Nonlinear genetic-based model for supplier selection: a comparative study, Technol. Econ. Dev. Econ. 23 (1) (2017) 178–195.
[12] K. Eggensperger, M. Lindauer, H.H. Hoos, et al., Efficient benchmarking of algorithm configurators via model-based surrogates, Mach. Learn. 107 (1) (2018) 15–41.
[13] Y. Wang, L.A. Kung, W.Y.C. Wang, et al., An integrated big data analytics-enabled transformation model: Application to health care, Inf. Manage. 55 (1) (2018) 64–79.
[14] N. Ghadimi, A. Akbarimajd, H. Shayeghi, et al., A new prediction model based on multi-block forecast engine in smart grid, J. Ambient Intell. Humaniz. Comput. 9 (6) (2018) 1873–1888.
[15] X. Yu, Disaster prediction model based on support vector machine for regression and improved differential evolution, Nat. Hazards 85 (2) (2017) 959–976.
[16] I. Boussaïd, P. Siarry, M. Ahmed-Nacer, A survey on search-based model-driven engineering, Autom. Softw. Eng. 24 (2) (2017) 233–294.
[17] M. Mohammadi, F. Talebpour, E. Safaee, et al., Small-scale building load forecast based on hybrid forecast engine, Neural Process. Lett. 48 (1) (2018) 329–351.
[18] F. Kong, Y. Wang, Multimodal interface interaction design model based on dynamic augmented reality, Multimedia Tools Appl. 78 (4) (2019) 4623–4653.
[19] G. Zacharewicz, S. Diallo, Y. Ducq, et al., Model-based approaches for interoperability of next generation enterprise information systems: state of the art and future challenges, Inf. Syst. E-Bus. Manage. 15 (2) (2017) 229–256.
[20] T. Mitchell, W. Cohen, E. Hruschka, et al., Never-ending learning, Commun. ACM 61 (5) (2018) 103–115.
[21] D. Hooshyar, M. Yousefi, H. Lim, A systematic review of data-driven approaches in player modeling of educational games, Artif. Intell. Rev. 52 (3) (2019) 1997–2017.
[22] C. Yang, L. Yang, M. Zhou, et al., LncADeep: an ab initio lncRNA identification and functional annotation tool based on deep learning, Bioinformatics 34 (22) (2018) 3825–3834.
[23] W. Wei, X. Xia, M. Wozniak, et al., Multi-sink distributed power control algorithm for cyber–physical-systems in coal mine tunnels, Comput. Netw. 161 (2019) 210–219.
[24] W. Wei, H. Song, H. Wang, et al., Research and simulation of queue management algorithms in ad hoc networks under ddos attack, IEEE Access 5 (2017) 27810–27817.
[25] W. Wang, N. Kumar, J. Chen, et al., Realizing the Potential of Internet of Things for Smart Tourism with 5G and AI, IEEE Netw. 34 (6) (2020) 295–301.
[26] M. Kishani, M. Tahoori, H. Asadi, Dependability analysis of data storage systems in presence of Soft Errors, IEEE Trans. Reliab. 68 (1) (2019) 201–215.
[27] W. Wang, X. Zhao, Z. Gong, et al., An attention-based deep learning framework for trip destination prediction of sharing bike, IEEE Trans. Intell. Transp. Syst. 22 (7) (2020) 4601–4610.
[28] A. Lavender, O. Elisha, The use of biometrics in multifactor authentication (MFA) for cloud computing data storage, Int. J. Comput. Appl. 178 (18) (2019) 30–37.
[29] J. Yang, H. Wang, Z. Lv, et al., Multimedia recommendation and transmission system based on cloud platform, Future Gener. Comput. Syst. 70 (2) (2017) 94–103.