Abstract—Data-driven predictive control (DPC) has been studied and applied in various scenarios. However, the challenge of computational efficiency remains. With the development of cloud computing, it provides potential solutions to the computational problem. Hence, this paper proposes a workflow-based fast DPC method in cloud-edge collaborative architecture. First, a workflow construction method of DPC is designed to make full use of the distributed computing ability of cloud computing. Next, to tackle the uncertainty in the cloud workflow processing, we design a cloud-edge collaborative scheme. In this scheme, an edge data-driven disturbance observer is proposed to estimate and compensate the uncertain event with guaranteed UUB stability. Then, an autonomous cloud control experimental system based on container technology is designed and implemented to execute the workflow-based DPC controller. Evaluations demonstrate that computation times are reduced by 45.19% and 74.35% for two real-time control examples, and by a maximum of 85.10% for a high-dimensional control example.

Note to Practitioners—This work is motivated by the challenge of the further combination of cloud computing and control system such as DPC. In the existing cloud-based control system, the computation mission of native control algorithm is deployed in a single cloud server directly. However, the structure of cloud environment is distributed, and the existing computation mode could not make full use of the parallel computing of cloud computing. Thus, the computation time could not be reduced significantly, and would still have serious effects on the control quality. In this work, a novel workflow-based DPC system in cloud-edge collaborative architecture is proposed, which decompose the native control mission into distributed cloud workflow with multiple smaller computation tasks. Then, an edge disturbance observer is designed to compensate the uncertainty occurring in the cloud workflow processing. As the evaluations show, the proposed workflow-based control system in cloud-edge collaborative architecture could be applied in the control mission with real-time requirement and the high-dimension control mission with intensive data.

Index Terms—Data-driven predictive control, cloud control system, cloud-edge collaborative control, container technology, cloud workflow processing.

I. INTRODUCTION

W

ith the development of Internet of Things and cyber-physical system, the scale and complexity of the control system increase rapidly [1]. This brings challenges for the control systems, of which the computing resources are limited and fixed. Cloud computing is a powerful solution, which outsources the workloads to virtual servers to improve the computation efficiency [2], [3], [4]. Cloud computing has advantages of sufficient computational resources, utilizing global information, handling large-scale data [5], which has been used in multi-agent, smart grids, autonomous vehicles, etc [6], [7], [8]. Based on cloud computing, cloud control system is concerned as a new paradigm to design and implement the control framework [9], [10], [11]. Since the introduction of cloud computing, more controlled plants are involved, and more data are produced and stored. It is hard even impossible to build the exact model of each plant, and the model-based control methods might be not applied. In the meanwhile, data-driven predictive control (DPC) method is developed to provide the control service without the need of the prior knowledge of system model. Therefore, data-driven predictive cloud control system (DPCCS) is proposed, to deal with the control problems only relying on the historical input and output data in the cloud control environment [12]. The typical structure of DPCCS is shown in Fig. 1. The latest input and output data are collected and uploaded to the cloud server. The DPC controller is deployed in the cloud server and produces the data-driven cloud predicted control sequence based on the received data. Then, the cloud predicted control sequence is downloaded to the local controlled plant and a complete control circle is finished. Time delay occurs in all stages of DPCCS, including the upload, computation and download courses. To reduce the influence of time delay, a compensator is designed to select a proper control variable, from the received cloud predicted control sequence based on the total time delay. The total time delay is measured as the difference between the recorded times of sending the sensed data and receiving the cloud signals.

However, the traditional scheme with time delay compensator is a passive solution to compensate the influence...
of time delay. In the traditional DPCCS, the computation mission of the native DPC method is deployed in a single cloud server directly, of which the computation mode is centralized. This kind of computation mode could not make full use of the parallel computing ability of cloud computing, while the distributed processing structure is one of the core advantages of cloud computing. Besides, only the power of cloud computing has been considered, the computing ability of edge device and the potential of the cloud-edge collaborative scheme have not been studied for DPCCS. The above reasons lead that time delay has not been reduced significantly, and would still have serious effects on the control quality. The detailed shortcomings of the traditional DPCCS are listed in the below.

1) The time delay compensator works only after time delay occurring, but has no effect on actively reducing the computation time.

2) When the total time delay is greater than a control period, a later ordered variable would be selected as the control input by the time delay compensator. But the variable ranking behind would bring larger control disturbance because of the accumulation of prediction errors.

3) In real-time control, the computation period for calculating control algorithm is constrained. Thus, the amount of the data and the scales of data matrices are restricted, which would affect the real-time control quality, especially for the high-dimension problem.

In this work, a novel cloud-edge collaborative workflow-based DPC system is proposed to actively reduce the computation delay, by utilizing the distributed ability of cloud computing. Cloud workflow processing has proved effective for improving the computation efficiency of scientific missions, such as deep learning, genetic calculation, etc., as it decomposes a complex, data-intensive application into smaller tasks and executes the tasks in the cloud environment [13], [14], [15], [16]. As shown in Fig. 2, cloud workflows are represented by the directed acyclic graphs (DAGs). The nodes represent the computational tasks, and the directed edges between the nodes determine the interdependencies between the tasks. Therefore, we propose the workflow-based DPC method, in which the DPC method with the centralized computation mode is restructured into the workflow form, based on the distributed numerical computation algorithm. When the workflow-based cloud controller starts, the tasks in the DPC workflow could be executed in parallel to reduce the computation delay in cloud control system.

However, in the DPC workflow tasks, the non-computation overheads are relatively high. As shown in Fig. 3, a middle-tier task is made up by one computation part and four non-computation parts, including the receiving data, deserialization, serialization and sending data parts. The computation part is the designed main function of the DPC workflow task, which is divided from the complete DPC mission. The parts of receiving and sending data are used to connect with the parent and children tasks. In the computation part, the data formats are 'float', 'double', etc. But, these data formats could not be delivered in the network space. Thus, serialization and deserialization are required, to convert the data formats in the computation part with the serialized data formats, such as 'bytes', 'string', 'json', etc. Since DPC is a control method relying on analyzing data, there are a lot of data transmitting between the DPC workflow tasks. Therefore, the overheads of receiving and sending data, serialization and deserialization are heavy, and have no benefits for reducing the completion time of the workflow-based DPC mission.
Next, to reduce the non-computational overheads and guarantee the control accuracy in the meanwhile, a cloud-edge collaborative scheme with disturbance observer (DOB) is proposed. In the DPC method, there are a lot of matrix operations on the large-scale Hankel matrices. The involved Hankel matrices are high-dimension and low-rank [16]. That is to say, most of the matrix information could be truncated. Then, the data amount of network transmission, serialization and deserialization could be decreased by the truncation operation, and so the non-computational overheads are reduced. However, the truncation operation would bring some uncertainties in the workflow processing. The uncertain event would cause adverse effects on the control performance, which is not what we expected. DOB is a disturbance estimation module, in which an auxiliary dynamical system estimates the effects of uncertainty acting on a system [17]. Once the uncertainty is estimated, it is possible to compensate for it by using the right magnitude of control effort. Therefore, we propose the cloud-edge collaborative workflow-based DPC system with DOB to solve this problem. In this system, the workflow-based DPC controller is deployed in the cloud environment, and a DOB is designed close to the edge controlled plant. The DOB is used to estimate the uncertainty produced in the workflow processing, and generate the cloud-edge composite control variable based on the received cloud control signals.

Finally, a workflow-based cloud control experimental system based on container technology is designed and established, to conduct the DPC workflow. This experimental system is used to evaluate the results of computation acceleration and control quality. At present, most of cloud workflow processing is studied based on virtual machine (VM) [18], [19]. VM technology provides a virtual operation system to access the computing resources, and create the isolated spaces for the tasks. However, the operation costs of VM are quite high, including the creation, startup, configuring times and configuring complexity. To reduce the costs, container technology is created and applied [20], [21], [22]. This technology enables that the smaller isolated processes could be created as well as started in an instant, and released automatically when the tasks are finished. Besides, the low cost of reusing container template makes control algorithm more convenient as a service. As summarized in Table I [23], container technology also has potential benefits on the computing resource usage, overhead cost and migration speed across host.

Motivated by the above reasons, the main contributions of this paper on the novel cloud-based DPC method are summarized in the following.

1) A workflow-based DPC method is proposed to tackle the issue of accelerating the data-intensive control computation missions. Through the truncated distributed singular value decomposition (SVD) algorithm, we develop a method of restructuring the DPC algorithm into DAG, which is the cloud workflow form.

2) A cloud-edge collaborative control scheme of the cloud accelerated controller and edge DOB-based uncertainty compensator is designed. To further improve the computation speed, the truncation precision could be adjusted. The upper bound of cloud control variable error is proved based on the truncation precision. Then, DOB is designed to estimate and compensate the uncertainty. The uniformly ultimately bounded (UUB) stability of the error dynamics of this DOB is guaranteed. The practical algorithm of this cloud-edge collaborative control scheme is also provided.

3) A practical containerised workflow-based cloud control experimental system is designed and implemented to carry out the workflow-based DPC missions. Specifically, the initialization, execution and communication schemes of the system are designed and established.

4) Finally, three control examples, consisting of two real-time systems and one high-dimension numerical system, are conducted to verify the performance of the proposed workflow-based DPC method and the containerised workflow-based cloud control experimental system. In the real-time control examples, which are the ball-beam system and vehicle tracking system, the computation times are reduced by 45.19% and 74.35%, respectively. In the high-dimension numerical example, the computation time is reduced by 85.10% at most.

The rest of this paper is organized as follow. Section II presents the preliminary of the DPC method. Section III provides the overview of the proposed cloud-edge collaborative containerised workflow-based DPC system with DOB. The computational features analyses and the construction method of the DPC workflow are presented in Section IV. The DOB and the cloud-edge collaborative control scheme with DOB are proposed in V. The design and establishment of the containerised workflow-based cloud control experimental system are provided in Section VI. The evaluations and discussions are provided in Section VII. The final section presents the conclusion and the future work.

## II. PRELIMINARY OF DATA-DRIVEN PREDICTIVE CONTROL

In this preliminary, the DPC method is recalled [24], [25]. Suppose the measurements of the inputs \( y(n), n = 1, 2, \ldots, 2N + j \) and outputs \( u(n), n = 0, 1, \ldots, 2N + j - 2 \) are available. The block Hankel matrices for \( u(n) \), represented as \( \underline{U}_p \) and \( \underline{U}_f \), with \( N \)-block rows and \( j \)-block

| Parameter                  | Container technology | VM technology |
|----------------------------|----------------------|---------------|
| Kernel state               | Booting needed       | Operational   |
| Creation time              | \(< 1 \, \text{s}\)   | 10 \, \text{min} |
| Start/stop time            | \(< 50 \, \text{ms}\) | 30-40 and 5-10 \, \text{s} |
| Startup overhead           | Low                   | Relatively high |
| Configuring complexity     | Medium                | High          |
| CPU/memory usage           | Low                   | High          |
| Overhead costs             | Low                   | High          |
| Migration                  | Quick                 | Slow          |
| Template reuse cost        | Low                   | High          |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
columns are defined as

$$U_p = \begin{bmatrix} u(0) & u(1) & \ldots & u(j-1) \\ u(1) & u(2) & \ldots & u(j) \\ \vdots & \vdots & \ddots & \vdots \\ u(N-1) & u(N) & \ldots & u(N+j-2) \end{bmatrix},$$

$$U_f = \begin{bmatrix} u(N) & u(N+1) & \ldots & u(N+j-1) \\ u(N+1) & u(N+2) & \ldots & u(N+j) \\ \vdots & \vdots & \ddots & \vdots \\ u(2N-1) & u(2N) & \ldots & u(2N+j-2) \end{bmatrix}.$$  

Each block element in the above Hankel matrices is a column vector of the input. Similarly, the Hankel matrices for \(y(n)\), represented as \(Y_p\) and \(Y_f\), can be written. Following the above subspace notations, the data column vectors over a prediction horizon for the inputs and outputs, respectively, are similarly defined as

$$u_f(k) = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-1) \\ u(k-N) \\ u(k-N+1) \\ \vdots \\ u(k-1) \end{bmatrix}, \quad y_f(k) = \begin{bmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ y(k+N) \\ y(k-N+1) \\ y(k-N+2) \\ \vdots \\ y(k) \end{bmatrix}.$$  

Then, we define the intermediate variable \(w_p\) as

$$w_p(k) = \begin{bmatrix} y_p(k) \\ u_p(k) \end{bmatrix}.$$  

Using regression analysis approach [24], we obtain

$$y_f(k) = L_w w_p(k) + L_u u_f(k) + L_e e_f(k),$$  

where \(e_f(k)\) is the column vector consisting of the noise \(e(n), n = k, k+1, \ldots, k+N-1\), and \(L_w, L_u, L_e\) are the corresponding coefficient matrices with proper dimensions. Since \(e_f(k)\) is made up by noise, the optimal prediction \(\hat{y}_f(k)\) is given as

$$\hat{y}_f(k) = L_w w_p(k) + L_u u_f(k).$$  

Then, \(L_w\) and \(L_u\) are required to calculate \(\hat{y}_f(k)\). In the DPC method, the Hankel matrices are used to compute \(L_w\) and \(L_u\). Refer to [25], the generalized regression of input and output Hankel matrices is obtained as The Hankel matrix of the predicted outputs, \(\hat{Y}_f\) is computed as

$$Y_f = L_w W_p + L_u U_f + L_e F.$$  

where \(W_p = [Y_p^T \ U_p^T]^T\) and \(F\) is the Hankel matrix of noise. Similarly, the Hankel matrix of the predicted outputs, \(\hat{Y}_f\) is computed as

$$\hat{Y}_f = L_w W_p + L_u U_f.$$  

By solving the following least squares problem

$$\min_{L_w, L_u}\|Y_f - [L_w \ L_u] [W_p \ U_f]^\top\|_F^2,$$  

where \(\cdot\|_F\) means \(F\)-norm, the coefficient matrices \(L_w\) and \(L_u\) could be computed by

$$[L_w \ L_u] = Y_f [W_p \ U_f]^\top.$$  

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
where $\dagger$ is the Morre-Penrose pesudo-inverse. Since $W = \begin{bmatrix} W_p \\ U_f \end{bmatrix}$ may be not square, the Morre-Penrose pesudo-inverse is used.

Finally, consider an optimal control problem with the objective function

$$J = (r_f - \hat{y}_f(k))^T (r_f - \hat{y}_f(k)) + u_f(k) (\lambda I) u_f(k).$$

where $r_f$ is the reference value of the output, $\lambda$ is an adjustable coefficient and $I$ is a unit matrix. Substituting (7) into (12) and taking derivative with respect to $u_f$, the data-driven predictive control law without constraints can be computed as

$$u_f(k) = (\lambda I + L_u^T L_u)^{-1} L_u^T (r_f - L_u w_p(k)).$$

### III. OVERVIEW

The structure of the workflow-based DPC system with DOB in cloud-edge collaborative architecture is provided as Fig. 4. This structure is made up of the cloud control platform and the edge controlled node. Some important modules are described in Table II. The cloud control platform is in the upper part of the structure. After receiving the input and output data from the edge controlled node, data are stored in the cloud data buffer. In the execution of the DPC workflow, the data buffer is kept as an unchanged scale by deleting the oldest data and adding the latest data. Then, the stored data are transmitted to the workflow data entry. In the meanwhile, the data-driven predictive control sequence is computed based on the decomposed results.

### Algorithm 1 Numerical Calculation Algorithm of the DPC Method

Input: The present time $k$, the constricted Hankel matrices $W_p$ and $U_f$, the least input and output signals $u(k)$ and $y(k)$ and the control cycle times $T$.

Output: Data-driven predictive control sequence $u_f(k)$.

1: for $k$ in $1$ to $T$ do
2: Update the Hankel matrices $W_p$ and $U_f$ by the least input and output signals $u(k)$ and $y(k)$.
3: Conduct SVD operation on $V_p = \begin{bmatrix} W_p \\ U_f \end{bmatrix}$ where $W_p$ is defined as the formula (8) and obtain the decomposed results:

$$V_p = M \cdot S \cdot N^T$$

4: Calculate the Morre-Penrose pesudo-inverse of $V_p$ by the decomposed results:

$$V_p^\dagger = N \cdot S^{-1} \cdot M^T.$$ 

5: Calculate the coefficient matrices $L_w$ and $L_u$:

$$L_w \cdot L_u = Y_f \cdot V_p.$$ 

6: Calculate the optimal predictive control sequence $u_f(k)$:

$$u_f(k) = (\lambda I + L_u^T L_u)^{-1} L_u^T (r_f - L_u w_p(k)).$$

7: end for

### A. Computation Amount Analysis of the DPC Method

From the preliminary, the data-driven predictive control law is calculated by (11) and (13). First, the coefficient matrices $L_w$ and $L_u$ are calculated as (12). Then, the $L_w$ and $L_u$ are used to compute the optimal data-driven predicted control strategy as (13). In the practical numerical calculation, the Morre-Penrose pesudo-inverse is obtained based on the SVD algorithm. Letting $V_p = \begin{bmatrix} W_p \\ U_f \end{bmatrix}$ and conducting SVD operation on $V_p$, we obtain

$$V_p = M \cdot S \cdot N^T$$

---

**TABLE II**

| Module                  | Description                                                                 |
|-------------------------|-----------------------------------------------------------------------------|
| Cloud Resource Pool     | An abstract concept for describing all computing resource including physical server, VM and container |
| Docker Register         | A sharing warehouse for storing reusable images of pre-defined tasks in DPC method |
| Workflow Template       | The template consisting of the task images and the topology structure between the tasks |
| Workflow Template Entry | A middle module to provide the template of the DPC workflow pulled from Docker Register |
| Workflow Data Entry     | A middle module to provide the input and output data of the controlled plant for the DPC workflow |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
where $S$ is the singular value and $M, N$ are decomposed coefficient matrices. As the result, the Morre-Penrose pseudo-inverse of $V_p$ could be calculated by the decomposed results as

$$V_p^\dagger = NS^{-1}M^T.$$  (15)

The numerical calculation procedure of the DPC method is summarized as Algorithm 1. To discuss the computational features, the calculation of the DPC method is divided into four stages, which are:

- SVD: conduct the SVD operation.
- Pseudo-inverse: obtain the Morre-Penrose pseudo-inverse.
- Coefficient matrices: compute the coefficient matrices.
- Control sequence: compute the final control sequence.

Further, to analyse the averaged calculation time of each stage, a series of control simulation examples with different dimensions are carried out in a cloud server. The type of the used server is ‘ecs.hfc6.xlarge’ type of the used server is 

$$ECS.HFC6.XLARGE$$

different dimensions are carried out in a cloud server. The calculation time increases from 0.4934 s for 2-dimension to 15.2524 s for 64-dimension. It is hard even impossible to finish the real-time control for the DPC method.

In this work, the acceleration is achieved by cloud workflow processing method. The construction method of the DPC workflow is proposed in Sections IV-B and IV-C. Besides, notice that the sum of the percentages of the four stages are less than 100%. The time of missing parts are spent by the constructions and updates of the Hankel matrices and other non-computation operations. This problem is solved by the design of communication topology switching, which is provided in Section VI-C.

### B. Parallelization Based on the Truncated Distributed SVD Algorithm

To accelerate the computation of the DPC method, we consider it in a distributed parallelization framework. The distributed SVD algorithm can achieve the distributed decomposition in a lightweight way [26], which inspires us to establish the DAG of SVD, for computing the Morre-Penrose pseudo-inverse.

Assume an $m \times n$ matrix $A$ is split column-wise into submatrices $A_1$ and $A_2$ with the sizes $m \times n_1$ and $m \times n_2$, respectively, and $n_1 + n_2 = n$. Conduct SVD operations on $A_1$ and $A_2$ and obtain the results $A_1 = U_1 \Sigma_1 V_1^T$, $A_2 = U_2 \Sigma_2 V_2^T$. Then the SVD of $A = [A_1 A_2]$ can be written as

$$\begin{bmatrix} A_1 & A_2 \end{bmatrix} = \begin{bmatrix} U_1 \Sigma_1 & U_2 \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T & 0 \\ 0 & V_2^T \end{bmatrix} = E \begin{bmatrix} V_1^T & 0 \\ 0 & V_2^T \end{bmatrix}$$  (16)

where $E = [U_1 \Sigma_1 & U_2 \Sigma_2]$. Conduct SVD on $E$ and obtain

$$\begin{bmatrix} A_1 & A_2 \end{bmatrix} = U \Sigma V^T \begin{bmatrix} V_1^T & 0 \\ 0 & V_2^T \end{bmatrix} = U \Sigma V^T$$  (17)

where $V^T = \tilde{V}^T \begin{bmatrix} V_1^T & 0 \\ 0 & V_2^T \end{bmatrix}$ is the product of two orthogonal matrices and hence is also an orthogonal matrix.

The low rank approximations would be conducted after the individual SVDs are finished, of which the criteria is that only the square matrices corresponding to the singular values are retained. Assume $rank(A) = r$, $rank(A_1) = k$ and $rank(A_2) = l$. Thus, $A_1 \approx U_1 \Sigma_1 V_1^T$ and $A_2 \approx U_2 \Sigma_2 V_2^T$ indicate rank-$k$ and rank-$l$ approximations, respectively. The result is truncated to a rank-$r$ approximation as (18). This operation is therefore named as merge-and-truncate (MAT) operation rather than a simple merging. When there are several partitions, the MAT operation can be conducted pairwise using a DAG-based strategy as created in Fig. 5.

$$\begin{bmatrix} A_1 & A_2 \end{bmatrix} \approx \begin{bmatrix} U_{1i} \Sigma_{1i} & U_{2i} \Sigma_{2i} \end{bmatrix} \begin{bmatrix} V_{1i}^T & 0 \\ 0 & V_{2i}^T \end{bmatrix} = U_r \Sigma_r V_r^T.$$  (18)
Algorithm 2 Distributed Truncated SVD Algorithm

Input: A matrix to be decomposed $A_{m \times n}$, block width $col$.
Output: Decomposed results $U$, $S$, and $V$.

1: function PARALLEL_SVD_BY_COLS($A_{m \times n}$, col)
2:     Calculate the number of the truncated blocks
3:          $N_{c} = \text{round}(m/\text{col} + 0.45)$
4:       where the round command means taking the nearest integer.
5:     Build a series of lists to store the truncated blocks and decomposed results $l_{i} = \text{list}(1)$, $l_{i} = \text{list}(1)$, $l_{i} = \text{list}(1)$, $l_{i} = \text{list}(1)$
6:          where the list command means creating an empty list.
7:     Fill the list $l_{i}$ by the $N_{c}$ column blocks of $A$.
8:     for each column block $A_{block}$ in the list $l_{i}$ do
9:         Do SVD on $A_{block}$ and obtain $U_{block}$, $\Sigma_{block}$, $V_{block}$
10:            $A_{block} = U_{block} \Sigma_{block} V_{block}^{T}$.
11:     end for
12:     end function
13: end function

Algorithm 2 Distributed Truncated SVD Algorithm

Input: A matrix to be decomposed $A_{m \times n}$, block width $col$.
Output: Decomposed results $U$, $S$, and $V$.

1: function PARALLEL_SVD_BY_COLS($A_{m \times n}$, col)
2:     Calculate the number of the truncated blocks
3:          $N_{c} = \text{round}(m/\text{col} + 0.45)$
4:       where the round command means taking the nearest integer.
5:     Build a series of lists to store the truncated blocks and decomposed results $l_{i} = \text{list}(1)$, $l_{i} = \text{list}(1)$, $l_{i} = \text{list}(1)$, $l_{i} = \text{list}(1)$
6:          where the list command means creating an empty list.
7:     Fill the list $l_{i}$ by the $N_{c}$ column blocks of $A$.
8:     for each column block $A_{block}$ in the list $l_{i}$ do
9:         Do SVD on $A_{block}$ and obtain $U_{block}$, $\Sigma_{block}$, $V_{block}$
10:            $A_{block} = U_{block} \Sigma_{block} V_{block}^{T}$.
11:     end for
12: end function

The above steps are summarized in Algorithm 2. The matrix $A_{m \times n}$ is first partitioned column-wise and the SVD of each partition is computed in the function PARALLEL_SVD_BY_COLS.
TABLE IV
RELATIONSHIPS BETWEEN TASKS IN WORKFLOW AND FUNCTIONS

| Image | Index | Level | Function |
|-------|-------|-------|----------|
| $Image_A$ | Task$_1$ | 1 | Pre-controller in data preparation stage and data router in the DPC stage |
| $Image_B$ | Task$_2$ – Task$_{11}$ | 2 | Compute the truncated SVD of each block matrices of $O_i$ |
| $Image_C$ | Task$_{12}$ – Task$_{18}$ | 3, 4 | 1. Merge the parent tasks’ results 2. Compute the new truncated SVD |
| $Image_D$ | Task$_{19}$ | 5 | 1. Merge all tasks’ results 2. Calculate the predictive control sequence |

1) The first task Task$_1$ occupies two roles in the workflow-based DPC method. In the begining, Task$_1$ serves as a pre-controller to prepare the initial data. The data are sent the children tasks to build the first batch of the Hankel matrices. Then, when the collected initial data are enough, Task$_1$ would become a data router in the DPC stage. The data received from the controlled plant would be transmitted to the children tasks to update the Hankel matrices.

2) The ten slices of Hankel matrices are built and updated in Task$_2$ – Task$_{11}$, respectively. In Task$_2$ – Task$_{11}$, the sliced blocks are decomposed using the function DOMergeOfBlocks.

3) Task$_{12}$ – Task$_{18}$ carry out the MAT operations on the results of Task$_2$ – Task$_{11}$, i.e., merge the parent tasks’ results and compute the new truncated SVDs.

4) Finally, the export task Task$_{19}$ collects all the results of Task$_1$, Task$_{16}$, Task$_{17}$ and Task$_{18}$ as well as merges them, and later calculates the cloud-based predicted control sequence.

In the workflow processing, the MPT is used to represent the maximal parallel tasks in a workflow, which means that at most MPT tasks are running in parallel during the workflow execution. In this case, the MPT is 10 and number of tasks is 19. This subsection only provides one case with a specific granularity. Using the same method, other structures of the DPC workflow could be constructed with different parameters for different kinds and scales of problems and different results would be obtained.

D. Computational Complexity Analysis

To study the improvement of this cloud workflow method, the computational complexity of Algorithm 2 is analyzed. Referring to [26], the floating point operations (flops) number of a complete SVD on an $m \times n$ matrix $A$ is approximated as $6mn^2 + 16m^3$. Without loss of generality, we assume all the blocks are divided with the same column width. Partition the matrix $A$ column-wise into $N$ blocks and the size of each block is $m \times s$ where $s = n/N$.

In Algorithm 2, the first step is to execute $Image_C$. Thus the flops number of the SVD of each block is $6ms^2 + 16s^3$. By the low rank approximation, assume that each SVD is truncated to a $k$-rank matrix. Then at each level of the DAG, the tasks with $Image_D$ are executed. The flops number of $Image_D$ consists of two functions, which are the cost of merging $(2ms^2)$ and the cost of new truncated SVD $(4mn^2 + 176k^3)$. The cost of the latter term is made up of the cost of SVD $(6 \cdot (2k) \cdot (2k)^2 + (2k)^3 = 176k^3)$ and the cost of updating the matrix $V (4nk^3)$. This image task is required to repeat $N - 1$ times. The total flops number is

$$T(m, n, N) = N(6ms^2 + 16s^3) + (N - 1)(6mn^2 + 176k^3).$$

Since $k \leq s$, the total number of flops

$$T(m, n, N) < \frac{12mn^2}{N} + \frac{192n^3}{N^2}.$$  

Thus, when the degree of parallelism $N$ grows, the proposed method can accelerate the DPC algorithm effectively. Since most of the tasks can be executed in parallel, the total time cost would be reduced further.

Remark 1: In the practical establishment of the DPC workflow, we place the last two tasks with $Image_C$ into the export task since the time costs of the two tasks are relatively less. Therefore, there are only 7 tasks with $Image_C$ in Fig. 6.

V. DOB-BASED CLOUD-EDGE COLLABORATIVE CONTROL SCHEME

In addition to parallelization, the computation efficiency could be further improved by adjusting the truncation precision in the distributed SVD algorithm. However, the adjustment also brings the loss of control accuracy of the cloud-based DPC method. In this work, the loss of control accuracy could be considered as an uncertain event in cloud control system. Therefore, this uncertainty could be thought as the input disturbance and estimated by the DOB deployed in the edge controlled node.

For the convenience of understanding, a series of variables are recalled or defined uniformly in this section. We assume that in the non-truncation situation, the coefficient matrices, i.e., the cloud parameters and control variable are obtained. These parameters and variable are accurate and written as $L_w$, $L_u$ and $u(k)$, respectively. The $u(k)$ is the first variable in the predicted control sequence $u_f(k)$. In the truncation situation, the corresponding cloud parameters and control variable are noted as $\hat{L}_w$, $\hat{L}_u$ and $\hat{u}(k)$, respectively. The $u_{cloud}(k)$ is the first term of the cloud predicted control sequence $\hat{u}_f(k)$. Thus, the errors between the above variables of the non-truncation and truncation situations are derived in the below.

$$L_w = \hat{L}_w + \Delta L_w,$$  

$$L_u = \hat{L}_u + \Delta L_u,$$  

$$u_f(k) = \hat{u}_f(k) + \Delta u_f(k),$$  

$$u(k) = u_{cloud}(k) + \Delta u(k)$$

where the $\Delta L_w$, $\Delta L_u$, $\Delta u_f(k)$ and $\Delta u(k)$ are the corresponding errors, respectively.

The error of the control variable $\Delta u(k)$ is considered as the uncertainty in cloud control system. The upper bound
of the cloud predicted control sequence error $\Delta u_f(k)$ is dependent on the cloud parameters errors $\Delta L_w$, $\Delta L_u$. Then, the upper bound of the cloud control variable error $\Delta u(k)$ is smaller than that of $\Delta u_f(k)$ because of the scaling relationship of $F$-norm. The detailed derivation is provided in Section V-A. The co-design of the DOB and DOB-based cloud-edge composite control law is designed in Section V-B. Finally, the upper bound of $\Delta u(k)$ is significant for the guarantee of the UUB stability of the DOB, which is analyzed in Section V-C.

### A. Upper Bounds of Cloud Parameter Errors $\Delta L_w$, $\Delta L_u$ and Cloud Control Variable Error $\Delta u(k)$

Now, we adopt the truncated distributed SVD algorithm to compute the Morre-Penrose pseudo-inverse of $V_p$ and obtain

$$V_p = \sum_{i=1}^{r} M_i S_i N_i^T$$  \hspace{1cm} (25)

where $r$ is the rank of $V_p$, $M_i$, $S_i$, $N_i$, $i = 1, 2, \ldots, r$ are the decomposed results of $V_p$, and the singular values are sorted from largest to smallest.

Using the truncated distributed SVD algorithm and setting the truncation precision as $\epsilon_1$, we obtain

$$\hat{V}_p \approx \sum_{i=1}^{k} M_i S_i N_i^T$$  \hspace{1cm} (26)

where $k$ is the number of singular values satisfying the truncation precision $S_i \geq S_{\text{max}} \times \epsilon_1$ and $S_{\text{max}}$ is the maximal singular value of $V_p$.

Thus the approximate Morre-Penrose pseudo-inverse of $\hat{V}_p$ is computed by

$$\hat{V}^\dagger_p = \sum_{i=1}^{k} N_i S_i^{-1} M_i^T.$$  \hspace{1cm} (27)

We use this approximate Morre-Penrose pseudo-inverse $\hat{V}^\dagger_p$ to compute the intermediate parameters $\hat{L}_w$, $\hat{L}_u$ and the data-driven predictive control law $\hat{u}_f(k)$. However, though the workflow-based DPC method with adjustable truncation precision could improve the computation efficiency significantly, it is obvious that the acceleration method introduces some errors. The relationships between the errors $\Delta L_w$, $\Delta L_u$ and the cloud parameters $\hat{L}_w$, $\hat{L}_u$ have been provided in (21) and (22), respectively. Then, based on the Morre-Penrose pseudo-inverse of $V_p$, the $[\hat{L}_w \hat{L}_u]$ could be calculated as

$$[\hat{L}_w \hat{L}_u] = \sum_{i=1}^{k} Y_f N_i S_i^{-1} M_i^T.$$  \hspace{1cm} (28)

Thus, the input-output relationship of (7) would be replaced by

$$\hat{y}_f(k) = \hat{L}_w w_p(k) + \hat{L}_u \hat{u}_f(k).$$  \hspace{1cm} (29)

The relationship between the error $\Delta u_f(k)$ and cloud predicted control sequence $\hat{u}_f(k)$ has been provided in (23). Next, the upper bound of the cloud predicted control sequence error $\Delta u_f(k)$ is derived. According to (30), the cloud predicted control sequence $\hat{u}_f(k)$ is dependent on the cloud parameters $\hat{L}_w$ and $\hat{L}_u$. Thus, $\Delta u_f(k)$ is dependent on the errors of the cloud parameters $\Delta L_w$ and $\Delta L_u$. In the below, the upper bounds of $\Delta L_w$ and $\Delta L_u$ are calculated.

By the SVD-based representation, the accurate cloud parameters $L_w$ and $L_u$ in the non-truncation situation is provided by

$$[L_w \ L_u] = \sum_{i=1}^{r} Y_f N_i S_i^{-1} M_i^T.$$  \hspace{1cm} (31)

Substituting (31) to (28) and calculating $\Delta L_w$ and $\Delta L_u$

$$[\Delta L_w \ \Delta L_u] = \sum_{i=k+1}^{r} Y_f N_i S_i^{-1} M_i^T.$$  \hspace{1cm} (32)

Thus,

$$\Delta L_w = \sum_{i=k+1}^{r} Y_f N_i S_i^{-1} M_{w,i}^T,$$  \hspace{1cm} (33)

$$\Delta L_u = \sum_{i=k+1}^{r} Y_f N_i S_i^{-1} M_{u,i}^T.$$  \hspace{1cm} (34)

where $M_{w,i}$ is the first $(m + l)j$ rows of $M_i$ and $M_{u,i}$ is the last $mj$ rows of $M_i$. Since $M_i$ and $N_i$ are the results of SVD, they are unitary matrices of which the 2-norms are 1. In this work, $\| \cdot \|$ means the 2-norm. Defining $S_{\text{min}}$ is the minimal singular value of $V_p$, we obtain

$$\| \Delta L_w \| \leq \alpha(r - k)S_{\text{min}}^{-1}Y_f \| = \epsilon_2,$$  \hspace{1cm} (35)

$$\| \Delta L_u \| \leq \beta(r - k)S_{\text{min}}^{-1}Y_f \| = \epsilon_3$$  \hspace{1cm} (36)

where $\alpha = \| M_{w,i}^T \|$, $\beta = \| M_{u,i}^T \|$ and $\alpha^2 + \beta^2 = 1$.

Then, we compute the upper bound of cloud control law error $\Delta u_f(k)$. Based on (13) and (30), it holds that

$$\begin{align*}
(\lambda I + L^T_u L_u) u_f(k) &= L^T_u r_f - L^T_u L_w w_p(k), \\
(\lambda I + \hat{L}^T_u \hat{L}_u) \hat{u}_f(k) &= \hat{L}^T_u r_f - \hat{L}^T_u \hat{L}_u w_p(k).
\end{align*}$$  \hspace{1cm} (37)

Substituting (21) and (22) to (37), we obtain

$$\begin{align*}
[\lambda I + (\hat{L}_u + \Delta L_u)^T(\hat{L}_u + \Delta L_u)]u_f(k) &= (\hat{L}_u + \Delta L_u)^T r_f - (\hat{L}_u + \Delta L_u)^T(\hat{L}_u + \Delta L_u) w_p(k).
\end{align*}$$  \hspace{1cm} (39)

Further, subtracting (39) to (38), we obtain

$$\begin{align*}
[\lambda I + (\hat{L}_u + \Delta L_u)^T(\hat{L}_u + \Delta L_u)]u_f(k) &= - (\hat{L}^T_u \Delta L_u + \Delta L^T_u \hat{L}_u + \Delta L^T_u \Delta L_u) \hat{u}_f(k) + \Delta L^T_u r_f \\
&= (\hat{L}_u - L^T_u \Delta L_w + L^T_u \Delta L_u) w_p(k).
\end{align*}$$  \hspace{1cm} (40)
Then,
\[
\|[\lambda I + (\hat{L}_u + \Delta L_u)^T(\hat{L}_u + \Delta L_u)]\Delta u_f(k)\|
\leq \| (\hat{L}_u + \Delta L_u)^T \Delta L_u \hat{u}_f(k) + \Delta L_u r_f \|
\leq (2\|\hat{L}_u\|\|\Delta L_u\| + \|\Delta L_u\|^2)\|\hat{u}_f(k)\| + \|\Delta L_u\|\|r_f\|
\leq (\|\hat{L}_u\|\|\Delta L_u\| + \|\Delta L_u\|^2)\|\hat{u}_f(k)\| + \|\Delta L_u\|\|\hat{L}_u\| + \|\Delta L_u\|\|\hat{L}_u\| + \|\Delta L_u\|\|\hat{L}_u\| + \|\Delta L_u\|\|\hat{L}_u\| + \|\Delta L_u\|\|w_p(k)\|.
\]
\[
(41)
\]
Since \((\hat{L}_u + \Delta L_u)^T(\hat{L}_u + \Delta L_u)\) is positive-definite, it follows that \(\|\Delta u_f(k)\| \leq \|[\lambda I + (\hat{L}_u + \Delta L_u)^T(\hat{L}_u + \Delta L_u)]\Delta u_f(k)\|\), which further results in
\[
\|\Delta u_f(k)\| \leq \frac{1}{\lambda} (2\|\hat{L}_u\|\|\Delta L_u\| + \|\Delta L_u\|^2)\|\hat{u}_f(k)\|
+ \frac{1}{\lambda} \left( \|\Delta L_u\|\|r_f\| + \|\Delta L_u\|\|\hat{L}_u\| + \|\Delta L_u\|\|\hat{L}_u\| + \|\Delta L_u\|\|w_p(k)\| \right).
\]
\[
(42)
\]
Then, we obtain the upper bound of cloud predicted control sequence error \(\Delta u_f(k)\)
\[
\|\Delta u_f(k)\| \leq \frac{1}{\lambda} (2\epsilon_3\|\hat{L}_u\| + \epsilon_3^2\|\hat{u}_f(k)\|
+ \epsilon_3\|\hat{L}_u\|\|r_f\|
+ \frac{1}{\lambda} \left( \epsilon_2\|\hat{L}_u\| + \epsilon_3\|\hat{L}_u\| + \epsilon_3\|w_p(k)\| \right).
\]
\[
(43)
\]
Finally, since \(\|\Delta u(k)\| \leq \|\Delta u_f(k)\|\), the upper bound of cloud control variable error is also derived. Note that \(\hat{L}_u, \hat{u}_f(k), r_f, w_p(k)\) are known in the computation of cloud controller. Thus, the upper bound of the cloud control variable error could be obtained.

Remark 2: From (35) and (36), the law of parameters setting could be obtained. If the truncation precision \(\epsilon_1\) is set low, the computation speed would be improved. But the number of the reserved singular values \(k\) becomes smaller and \(r - k\) becomes larger. Thus, the right sides of the two inequations, i.e., the upper bounds of the cloud parameter errors become larger, which means the accuracy of the workflow-based DPC is decreased. The derived formulas are in agreement with the experimental results provided in Section VII.

Remark 3: In the practical conduction, the number of the retained singular values also could be used to adjust the truncation accuracy. In the truncation operation of the workflow-based DPC method, there are a large amount of the singular values, which are in the same order of magnitudes. If we adjust the truncation accuracy by increasing or decreasing \(\epsilon_1\), the change of the upper bounds of the uncertainty would be large. Then, the trade-off between the computation efficiency and control performance would be hard to handle. Therefore, in the third control example: high- or decreasing the retained singular values also could be used to adjust errors become larger, which means the accuracy of the inequations, i.e., the upper bounds of the cloud parameter error is also derived. Note that \(\hat{u}_f(k), \hat{b}(k), i = 1, 2, \ldots, N\) is the first block-row of \(\hat{L}_u\) at sampling time \(k\), \(\hat{b}(k)\) is the first block-element of \(\hat{L}_u\) at sampling time \(k\) and \(\{u(i), i = k - N, k - N + 1, \ldots, k - 1\}\) are the input and output data in the edge data buffer, respectively. The real system considering the uncertainty is given by
\[
y(k + 1) = \sum_{i=1}^{N} \hat{a}_i(k)y(k - N + i)
+ \sum_{i=0}^{N-1} \hat{b}_i(k)u(k - N + i) + \hat{b}(k)u_{cloud}(k)
\]
\[
(44)
\]
where \(\hat{a}_i(k), \hat{b}_i(k), i = 1, 2, \ldots, N\) is the first block-row of \(\hat{L}_u\) at sampling time \(k\) and \(\{u(i), i = k - N, k - N + 1, \ldots, k - 1\}\), \(\{y(i), i = k - N + 1, k - N + 2, \ldots, k\}\) are the input and output data in the edge data buffer, respectively.

B. DOB and DOB-Based Cloud-Edge Composite Control Law

For the convenience of analysis, we select the first term in the cloud predicted control sequence \(\hat{u}_f(k)\) as the cloud control variable \(u_{cloud}(k)\), which would be transmitted to the edge controlled node. In the \(k\)-th time, the first variable of the predicted control sequence, \(u_{cloud}(k)\) would be adopted to the plant. Thus, only the corresponding parameters to \(\hat{u}_f(k)\) need to be sent to the plant. That is to say, based on (29), the nominal system is
\[
\hat{y}(k + 1) = \sum_{i=1}^{N} \hat{a}_i(k)y(k - N + i)
+ \sum_{i=0}^{N-1} \hat{b}_i(k)u(k - N + i) + \hat{b}(k)u_{cloud}(k)
\]
\[
(44)
\]
where \(\hat{a}_i(k), \hat{b}_i(k), i = 1, 2, \ldots, N\) is the first block-row of \(\hat{L}_u\) at sampling time \(k\), \(\hat{b}(k)\) is the first block-element of \(\hat{L}_u\) at sampling time \(k\) and \(\{u(i), i = k - N, k - N + 1, \ldots, k - 1\}\), \(\{y(i), i = k - N + 1, k - N + 2, \ldots, k\}\) are the input and output data in the edge data buffer, respectively.

The real system considering the uncertainty is given by
\[
y(k + 1) = \sum_{i=1}^{N} \hat{a}_i(k)y(k - N + i)
+ \sum_{i=0}^{N-1} \hat{b}_i(k)u(k - N + i) + \hat{b}(k)u_{cloud}(k)
\]
\[
(45)
\]
where \(d(k) = \Delta u(k)\) is considered as the input disturbance, and \(\|d(k)\| = \|\Delta u(k)\| \leq \epsilon_4\).

By [27], we design the DOB in the form of
\[
\hat{d}(k) = P(k) + Ly(k)
\]
\[
(46)
\]
where \(\hat{d}(k)\) is the estimation of disturbance vector \(d(k)\), \(L\) is the observer gain matrix to be designed, and \(P(k)\) is the auxiliary variable vector updated by
\[
P(k + 1) = -L\sum_{i=1}^{N} \hat{a}_i(k)y(k - N + i)
+ \sum_{i=0}^{N-1} \hat{b}_i(k)u(k - N + i) + \hat{b}(k)u_{cloud}(k)
+ \hat{b}(k)\hat{d}(k).
\]
\[
(47)
\]
Substituting (47) to (46) and calculating \(\hat{d}(k + 1)\)
\[
\hat{d}(k + 1) = P(k + 1) + Ly(k + 1),
\]
\[
\hat{d}(k + 1) = L\hat{b}(k)(d(k) - \hat{d}(k)).
\]
\[
(49)
\]
Defining the estimation error as \(\hat{d}(k) = d(k) - \hat{d}(k)\) and subtracting both sides of (49) from \(d(k + 1)\) gives
\[
d(k + 1) - \hat{d}(k + 1) = d(k + 1) - L\hat{b}(k)d(k).
\]
\[
(50)
\]
The error dynamics of the proposed observer is provided by
\[ \dot{d}(k + 1) = -L \hat{b}(k) \hat{d}(k) + d(k + 1). \]  
(51)

In this system, the DOB-based cloud-edge composite control law is designed as
\[ u(k) = u_{\text{cloud}}(k) + u_{\text{com}}(k) \]  
(52)
where \( u_{\text{cloud}}(k) \) is the cloud control input and \( u_{\text{com}}(k) \) is the edge DOB-based compensative control variable.

Since the lumped disturbance is considered as the input disturbance, the compensator is designed as
\[ u_{\text{com}}(k) = -\hat{d}(k). \]  
(53)

Thus, when \( k = N + 1, N + 2, \ldots \), we obtain the co-design of cloud-edge composite control law and DOB-based compensator as follows
\[
\begin{align*}
\dot{\hat{d}}(k) &= P(k) + Ly(k) \\
u_{\text{com}}(k) &= -\hat{d}(k) \\
u(k) &= u_{\text{cloud}}(k) + u_{\text{com}}(k) \\
P(k + 1) &= -L(\sum_{i=1}^{N} \hat{a}_i y(k - N + i) \\
&\quad + \sum_{i=0}^{N-1} \hat{b}_i (k - N + i)u(k - N + i) + \hat{b}(k)u(k) + \hat{b}(k)\hat{d}(k)))
\end{align*}
\]  
(54)

According to (54), when \( k = N + 1, N + 2, \ldots \), an alternative expression for calculating \( \hat{d}(k) \) is given by
\[
\hat{d}(k) = L(y(k) - \sum_{i=1}^{N} \hat{a}_i y(k - N + i - 1) \\
- \sum_{i=0}^{N-1} \hat{b}_i (k - N + i)u(k - N + i - 1) \\
- \hat{b}(k - 1)u_{\text{cloud}}(k - 1)).
\]  
(55)

Besides, when \( k = 1, 2, \ldots, N \), the data amount in the edge data buffer is not enough to execute the DOB-based compensator. Thus, the DOB-based compensator would not work and the cloud control variable would be used as the practical control input, i.e., \( u(k) = u_{\text{cloud}}(k) \).

C. Stability Analysis of the DOB

Now, we will prove the stability of the designed DOB. Rewrite the error dynamics (51) of the proposed DOB as
\[ \dot{\hat{d}}(k + 1) = H \hat{d}(k) + d(k + 1) \]  
(56)
where \( H = -L \hat{b}(k) \).

**Proof:** Since \( \hat{b}(k) \) is the first block-element of the bounded \( \hat{L}_u \), the estimated error \( \Delta b(k) \) is also bounded by \( \epsilon_3 \), i.e., the upper bound of \( \|\Delta L_u\| \). Therefore, though \( \hat{b}(k) \) is a time-varying variable, it is still bounded and the eigenvalues of matrix \( H \) could be arbitrary placed in a unit circle by selecting a proper \( L \). One can always find a positive-definite matrix satisfying \( PH^TPH - P = -Q \). It can be verified that \( \delta = \lambda_{\text{min}}(Q) < 1 \), where \( \lambda_{\text{min}}(Q) \) is the smallest eigenvalue of \( Q \) and \( \lambda_{\text{max}}(P) \) the largest eigenvalue of \( P \). Define a Lyapunov function as
\[ V(\hat{d}(k)) = \hat{d}(k)^T P \hat{d}(k) \]  
(57)
and calculate the increment of \( V(k) \) for \( k \in [N + 1, \infty) \) as
\[
\Delta V(k) = V(\hat{d}(k + 1)) - V(\hat{d}(k)) \\
= \hat{d}(k + 1)^T P \hat{d}(k + 1) - \hat{d}(k)^T P \hat{d}(k) \\
= (H \hat{d}(k) + d(k + 1))^T P(H \hat{d}(k) + d(k + 1)) \\
- \hat{d}(k)^T P \hat{d}(k) \\
= \hat{d}(k)^T (H^TPH - P) \hat{d}(k) + 2\hat{d}(k)^T HPd(k + 1) \\
+ d(k + 1)^T Pd(k + 1) \\
\leq -\lambda_{\text{min}}(Q) \|\hat{d}(k)\|^2 + 2\epsilon_4 \|H^TP\| \|\hat{d}(k)\| + \epsilon_3^2 \|P\|. 
\]  
(58)
Let \( \mu = 2\epsilon_4 \|H^TP\| \|\hat{d}(k)\| + \epsilon_3^2 \|P\| \). For \( k \in [N + 1, \infty) \), \( \Delta V(k) \) is obtained as
\[
\Delta V(k) \leq -\lambda_{\text{min}}(Q) \|\hat{d}(k)\|^2 + \mu \\
\leq -\lambda_{\text{min}}(Q) \|\hat{d}(k)\|^2 + \lambda_{\text{max}}(P) \|\hat{d}(k)\|^2 + \mu \\
= -\delta V(k) + \mu. 
\]  
(59)
Then, we have
\[ V(\hat{d}(k)) - V(\hat{d}(k - 1)) \leq -\delta V(\hat{d}(k - 1)) + \mu \]  
(60)
which gives
\[ V(\hat{d}(k)) \leq (1 - \delta) V(\hat{d}(k - 1)) + \mu. \]  
(61)
For \( k \in [N + 1, \infty) \), this together with (59) implies that
\[ V(\hat{d}(k)) \leq (1 - \delta)^{k - N - 1} V(\hat{d}(N + 1)) + \frac{\mu}{\delta} \]  
(62)
and
\[ \|\hat{d}(k)\|^2 \leq (1 - \delta)^{k - N - 1} \frac{\lambda_{\text{max}}(P)}{\lambda_{\text{min}}(P)} \|\hat{d}(N + 1)\|^2 + \frac{\mu}{\delta \lambda_{\text{min}}(P)} \]  
(63)
where \( \lambda_{\text{min}}(\cdot) \) is the smallest eigenvalue of matrix.

Considering that \( \delta \leq 1 \), we have \( \lim_{\delta \to 0} (1 - \delta)^{k - N - 1} = 0 \), which brings that the states of system (56) are UUB. 

Here, we obtain that the error dynamics of the proposed DOB is UUB. In addition, because of the separability of controller and observer designs and that the stability of the workflow-based DPC method is another question, the stability of cloud controller would be studied in the future. The stability research of the normal DPC method could be found in [28].

---

**References:**

[28] R. E. Kalman, “Analytical basis for the control of dynamical systems,” Transactions of the ASME Journal of Basic Engineering, vol. 80, no. 2, pp. 371–391, 1958.
D. Overall Algorithm of Cloud-Edge Collaborative Control System

The overall algorithm of the cloud-edge collaborative control system is summarized in Algorithm 3. This algorithm procedure is designed as four parts: (1) preparing the initial information; (2) cloud control platform: data processing and cloud control variable computation; (3) edge controlled node: DOB-based compensation, cloud-edge composite control variable computation and executing the control input; (4) data update with feedback. The four steps are performed in this system as a circulation. Notice that all the procedures of cloud control platform are finished in the workflow-based controller. Besides, in this system, we implicitly identify the system model in real time, yet the model is not established throughout the course of control. At each sampling time, this model is updated by real-time data. Then, the predictive control method is deployed in this implicit model. In fact, this model is cancelled as an intermediate process and the control process only relies on data.

VI. DESIGN AND IMPLEMENTATION OF CONTAINERISED WORKFLOW-BASED CLOUD CONTROL EXPERIMENTAL SYSTEM

To conduct the workflow-based DPC controller in real cloud environment and evaluate the performance of the cloud-edge collaborative control system, the containerised workflow-based cloud control experimental system is designed and implemented. First, the structure of this experimental system is provided. Second, the initialization procedure and execution form of this system are designed. Finally, the communication scheme of this system is established, including the data transmission in the container network, IP and port mappings between the cloud environment and edge node, and the communication topology switching from the data preparation stage to the DPC stage.

A. Structure of the Experimental System

The structure of this containerised cloud control system is provided as Fig. 11. This system is made up of the cloud control platform and edge controlled node. In the edge controlled node, an edge data transfer is set to exchange data with the cloud controller. The data transfer also exchanges data with the controlled plant, which could be real dynamical system or mathematical model of simulation.

In the cloud control platform, there are task manager, Docker register, DAG file, remote dictionary storage (redis) server and containerised cloud workflow of which the tasks run in the overlay container network [29]. First, the redis server is created by the task manager. The redis server could share data online among different hosts via network. Thus, the redis server is used as a data transponder in the cloud environment. Then, the task manager defines the DAG files as ‘Map’ structure and writes the file into the redis server. When the workflow containers are created, the redis server would provide guides for the new containers based on the written DAG file, of which the details are provided in Section VI-B. The guide information contains:

- The task index and the level to this task belongs.
- The pre- and post- dependence relationships of this task.
- The image which would be pulled from Docker register. Besides, the task manager creates the images of workflow task and uploads the images to Docker register, which services as an image hub. Finally, the task images are pulled from Docker register and the workflow containers are created in the across-hosts overlay network. After the initialization guided by the redis server, the workflow-based DPC controller starts running.

B. Initialization and Execution of System

1) Procedure of the System Initialization: In this containerised workflow-based cloud control experimental system, the initialization of containers is an important procedure. When the containers in workflow are created, the containers have no knowledge about the position, level, IP and pre-/post-relationships in the whole workflow of each task. As a result, the workflow-based DPC controller could not be executed. Therefore, the initialization of containers is necessary for starting the workflow-based cloud controller. Fig. 8 shows the initialization procedure of the experimental system. The procedure is divided into the user operations and system operations. The user operations includes creating the redis server, defining DAG file, etc which are finished off-line. As for the online part, the system operations are divided into 12 steps, as listed in the below.

1) Write the defined DAG file including DAG topology map into the redis server.
2) Pull the images of tasks to the overlay network environment from Docker register.
Algorithm 3 Overall Algorithm of the Cloud-edge Collaborative Control Scheme

Input: The scale parameters \( N \) and \( j \) of the Hankel matrices, truncation precision \( \epsilon_1 \) and reference value \( r_f \).

Output: Cloud-edge composite control signal \( u(k) \) provided by the cloud controller and edge DOB-based compensator.

1: Preparing the Initial Information
   1) Set parameters:
      - First, set the scale parameters \( N \) and \( j \) to determine the size of the Hankel matrices. The truncation precision \( \epsilon_1 \) and reference \( r_f \) also needs to be provided.
   2) Collect data:
      - Generate a series of initial control input data \( \{ u(0), u(1), \ldots, u(2N + j - 1) \} \) by a certain algorithm such as PID or LQR. Then measure the output data of the plant \( \{ y(1), y(2), \ldots, y(2N + j) \} \) by sensors.
   3) Create Hankel matrices:
      - Create the initial Hankel matrices \( U_f, U_p, Y, Y_p \) of the collected input and output data.

2: Cloud Control Platform
   1) Cloud parameters:
      - At the sampling time \( k \), build the Hankel matrix \( V_p = \begin{bmatrix} W_p & U_f \end{bmatrix} \) where \( W_p \) is provided as the formula (8).
      - Conduct SVD operation on \( V_p \) as Algorithm 2 and obtain \( M_i, S_i, N_i, i = 1, 2, \ldots, r \) where \( r \) is the rank of \( V_p \).
      - Obtain the reserved SVD results: \( M_i, S_i, N_i, i = 1, 2, \ldots, k \) where \( k \) is the number of the singular values satisfying \( S_i \geq S_{\text{max}} \ast \epsilon_1 \) and \( S_{\text{max}} \) is the maximal singular value of \( V_p \).
      - Calculate the cloud parameters \( \hat{L}_w, \hat{L}_u \) as \( \hat{L}_w \hat{L}_u = \sum_{i=1}^{k} YN_iS_i^{-1}M_i^T \).

   2) Control variable:
      - Obtain the control sequence \( \hat{u}_f(k) = [\hat{u}_f(k)(1), \hat{u}_f(k+1)(1), \ldots, \hat{u}_f(k+N)(1)] \) as the formula (30) by using \( \hat{L}_w, \hat{L}_u \).
      - Take the first term of \( \hat{u}_f(k) \) as the cloud control variable \( u_{\text{cloud}}(k) \).

   3) Package and send:
      - Pack the cloud control variable \( u_{\text{cloud}}(k) \) and cloud parameters \( \hat{L}_w, \hat{L}_u \). Then send the information packet to the edge client together.

3: Edge Controlled Node
   1) Receive data:
      - The edge client receives the cloud control variable \( u_{\text{cloud}}(k) \) and cloud parameters \( \hat{L}_w, \hat{L}_u \).
   2) Composite control and connect actuator:
      - When \( k = 1, 2, \ldots, N \), let the DOB-based compensator being in idle state and set \( u(k) = u_{\text{cloud}}(k) \).
      - When \( k = N + 1, N + 2, \ldots \):
        a) Build the DOB-based compensator based on the cloud parameters \( \hat{L}_w, \hat{L}_u \) and obtain the edge compensative control variable \( u_{\text{com}}(k) \) as the formula (54).
        b) Obtain the composite control variable \( u(k) = u_{\text{cloud}}(k) + u_{\text{com}}(k) \).
      - Then, apply \( u(k) \) to the dynamic system and detect the corresponding output \( y(k) \).

   3) Package and send:
      - Pack the composite control input \( u(k) \) and the corresponding \( y(k) \) as a combined packet. Then store this packet in the edge buffer and send it to cloud server for the optimization at next step.

4: Data Update with Feedback
   1) Update Hankel Matrices:
      - The cloud server receives the latest packet of \( u(k) \) and \( y(k) \), and updates the Hankel matrices by removing the earliest data as well as adding the newest data to keep the fixed size \( N \).
   2) Feedback:
      - Repeat the course of Cloud Control Platform and continue the circulation until receiving a stop command.

3) Create the workflow containers as the pulled images.
4) Read the adjustable environment variables as the controller parameters such as the scale parameters \( N \) and \( j \).
5) After the containers start, obtain the IP of each container in the overlay container network.
6) Obtain the name of each task from the environment variables.
7) Store the ‘Name-IP’ pair as the ‘key-value’ mode into the redis server.
8) The containers read the defined DAG topology map from the redis server.
9) Create the task lists of the parents and children of each container based on the loaded DAG topology map and task name.
10) Read the container IPs of parents and children of each task container.
11) Create container IP lists of the parents and children tasks.
12) Create the TCP sockets with the pre- and post-containers.
2) Pyramid Structure of System Initialization: When the containers of the DPC workflow are created, the times of initialization including obtaining necessary information such as DAG map, task index, position, IP, port, etc. are different.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
for different containers. Some containers might finish the course earlier, while the others are initialized lately. The inconsistency of initialization would bring new uncertainty for the execution of the DPC workflow. Therefore, in this system, an initialization scheme of which the communication structure seems like pyramid is designed.

The pyramid communication structure is indeed an abstract of the relationship of the redis server and the task containers of a workflow. As shown in Figure 9, the redis server plays the role of guide and controller for the task containers of workflow. By several rounds of ‘key-value’ operations, the containers obtain the necessary information to execute the workflow. This stage is the initialization of the designed system. When a container is ready, it would send a ‘ready’ signal to the executor (VM or container) in which the redis server runs. When all the ‘ready’ signals are collected and acknowledged, this executor would send the ‘start’ signal to each container at the same time. The time differences of the containers receiving the ‘start’ signal is very small and could be ignored. Thus, the task containers could be considered as starting at the same time.

The practical containerised workflow and the communications between containers are established. Then, the redis server would not take part in the workflow execution unless being invoked. Thus, the lines between the redis server and containers are dotted, and this structure is named as pyramid. Up to here, the problem of initialization is solved. The workflow-based DPC controller is ready to execute the control missions.

3) Execution of System: After the initialization procedure is finished, the practical DPC workflow is created and executed. In the cloud environment, the practical containerised DPC workflow is created referring to the abstract workflow template by the the experimental system, as presented in Fig. 10. Notice that Fig. 10 shows the mapping relationship of the abstract and practical workflows, but does not represent any specific workflow structure of the DPC method. Each task in the abstract workflow would be translated to an executing container scheduled into a processing node such as VM.
The inter-dependencies between the tasks would be translated into the across-nodes communications to transmit the intermediate data. Then, the practical DPC workflow is executed in the overlay container network to conduct the control missions. When the practical DPC workflow is repeated one time, a new cloud control variable and a pair of cloud parameters would be obtained.

C. Communication Scheme of Containerised Workflow-Based Cloud Control Experimental System

In this system, the cloud controller and controlled plant are deployed in different network environments where the problems such as containers communication and the transfers of IP and port, etc arise. Thus, a real-time, stable and high-quality communication scheme is required. The structure of the communication scheme is provided in Fig. 11. There are the overlay container network for the workflow-based cloud controller, IP mapping from cloud network to edge network and port mapping from edge network to cloud network, topology switching scheme in data-driven control in this scheme.

In the cloud-edge communication, the protocol is chosen as user datagram protocol (UDP) because of its high speed and the low enough packet loss rate when crossing different networks. In the cloud environment, since the parallel multiple communications among the task containers, transmission control protocol (TCP) is more suitable as its ability of connection-oriented point-to-point transmission to avoid the communication conflicts. In the edge controlled node, both of TCP and UDP are proper in the communication and TCP is applied in this work.

1) Data Transmission in the Workflow-Based Controller via Overlay Container Network: In this work, all the processing nodes are set as VMs and connected by network. However, the containers distributed in different processing nodes could not communicate with other containers by the network for VMs. Therefore, a high-level network is required for the temporary containers with different hosts, which is the container overlay network [30]. The container overlay network in the clusters is set as calico protocol, a scalable network to build the connections among containers. A high-level overlay network segment which is prepared for the containers by the calico protocol. When a new container is created, this protocol dispatches a unique overlay IP in the built the overlay network segment to communicate across the existing different hosts.

2) IP Mapping From Cloud Network to Edge Network: Since the amount of the public IP addresses is limited and each device running in the edge controlled node also needs an IP address, network and port translation (NAPT) technology is adopted in local area network (LAN). In the edge node, all the devices share the same public IP and this public IP is mapped into different edge IPs for the devices in LAN. However, the cloud controller has no knowledge about the hosts of the edge devices at the beginning. Thus, an IP transfer method is designed. The edge controlled plant sends an information group consisting of the input and output signals and internal tuple to the cloud controller. Then, the cloud controller obtains the network host of this edge plant. Furthermore, when the message is received by the cloud controller or edge plant, the receiver would check the attached port of the sender. If the ports of the receiver and sender are matched, the acknowledge process is completed.

3) Port Mapping From Edge Network to Cloud Network: When the containers run in the cloud control platform, the displayed IP for the edge controlled node is the IP of the hosted VM. Besides, multiple containers share the same IP of virtual machine. Therefore, when the output data is sent from the edge controlled node, the entry container could not be found. To solve this problem, the port mapping is used here. For example, the outside port ‘8080’ of host is mapped to the inner port ‘90’ of the entry container. Thus, the output data could be transmitted to the workflow-based controller directly.

4) Topology Switching Scheme in Data-Driven Control: As described in Algorithm 3, the initial data preparation occupies an important role in the DPC method. In the stage of preparing information, other basic control algorithm could be applied. But when the initial data is enough, the communication topology needs change because of the different requirements for the communication structures of the basic algorithms and the DPC algorithm. In the stage of initial data preparation, the edge plant connects to the initial cloud controller directly. This initial cloud controller runs in a single container. And this
initial cloud controller sends the latest data (the packet of input and output data) from the edge controlled node in each control period. In the same time, the time-costing establishments of the Hankel matrices in the workflow containers are executed. Otherwise, there would exist a considerable time delay in the switching period, since the establishments of the Hankel matrices are required to be finished in one period.

When the communication topology turns to the DPC stage, the initial cloud controller becomes a data router in the cloud environment but not an independent controller. In each control period, the data router still plays the role of the entry container and receives the latest data from the edge controlled node. Then, the data router dispatches the required data to the children tasks in the workflow-based cloud controller. The update method of the Hankel matrices costs relatively low computation burden. Next, the containers of the workflow-based cloud controller execute the tasks owned by themselves. The export container collects the data from its parents and calculates the cloud control variable as well as cloud parameters. Finally, the cloud control variable and cloud parameters are sent to the edge controlled node.

VII. EVALUATION AND DISCUSSION

In this section, three control examples, including the ball-beam system, vehicle tracking system and high-dimension numerical system, were conducted to evaluate the effectiveness of the proposed workflow-based cloud control methods and system. First, the experiment setup is described in detail. Second, four DPC methods to be tested are listed. Then, a brief of three control examples is provided. Finally, the detailed results and discussions of the three control examples are presented, respectively.

A. Evaluation Setup

The calculation of the DPC controller, including the existing native DPC and workflow-based DPC are practically carried out in Alibaba Cloud. Furthermore, the workflow-based DPC methods are conducted based on the containerised workflow-based cloud control experimental system established in Section VI. The acceleration of control algorithm is achieved in reality by the workflow-based method provided in Section IV, rather than relying on the numerical simulator. In the edge controlled node, the cloud-edge composite control variable is generated, as presented in Section V. Then, the composite control variable is delivered to the controlled plant, which is simulated by scripting codes. Thus, the evaluation is made up of the practical cloud experiment of the controller and the edge simulation of the controlled plant.

There are two structure designs of the DPC workflow with different MPT, which are used in the evaluation. The first structure is designed with the MPT being 4 and the task number being 6, as shown in Fig. 13. This DPC structure is used in the ball-beam system and vehicle tracking system, of which have strong real-time requirement and relatively less data. The second structure applies the design of Fig. 6 with the MPT being 10 and the task number being 19, which is carried out in the high-dimension control example. Besides, there are three kinds of cloud resource pool in the evaluation. The resources and node cluster information are listed in Table V. The first group is used to carry out the baseline, which the DPC controller is deployed in a single cloud server and calculated in the centralized mode. The second and third groups apply two ecs.hfc6.4xlarge nodes and each node has 16 CPU and 32 GB memory. The overlay network in the cluster is set as calico protocol, a scalable network to build the connections between containers. The version of redis is 3.5.3. The containerised cloud control system and the tasks of the DPC workflows are implemented by Python 3.7. The basic scientific libraries used in the implementation of the DPC workflows are Numpy (1.19.5) and Scipy (1.5.4), respectively. The operation system is Ubuntu 20.04.

B. Tested Cloud-Based DPC Methods

There are four cloud-based and two personal-computer-based (PC-based) DPC methods tested in the evaluation. The former two methods are the native DPC, of which the computation mode of the cloud controller is the traditional centralized structure [12], rather than the workflow-based structure. The middle two methods are the proposed workflow-based DPC in this paper. The last two methods are used for extra comparisons. The details are provided in the below.

1) Native DPC: a native DPC controller is deployed in the cloud server, and no extra component is deployed in the edge controlled node. The first control variable in the received predicted control sequence is applied to the controlled plant.

2) Native DPC with time delay compensator: a native DPC controller is deployed in the cloud server, and a time delay compensator is deployed in the edge controlled node. The cloud controller sends a predicted control sequence, and the time delay compensator selects a proper element to reduce the influence of time delay.

| Experiment Group | Nodes | Total Resource Pool | Node Type |
|------------------|-------|---------------------|-----------|
| Group 1          | 1     | 4 CPU, 8GB          | ecs.hfc6.4xlarge |
| Group 2          | 2     | 32 CPU, 64GB        | ecs.hfc6.4xlarge |
| Group 3          | 4     | 64 CPU, 128GB       | ecs.hfc6.4xlarge |

Fig. 13. Workflow structure tested in control examples 1, 2 (MPT = 4).
3) **Workflow-based DPC:** a workflow-based DPC controller is deployed in the cloud resource pool, and no extra component is deployed in the edge controlled node.

4) **Workflow-based DPC with DOB:** a workflow-based DPC controller is deployed in the cloud resource pool, and a data buffer and a DOB-based compensator are deployed in the edge controlled node to reduce the uncertainty caused in the cloud workflow processing.

5) **PC-based native DPC:** a native DPC controller is deployed in a local PC with i7-10710U CPU, 16 GB memory.

6) **Optimization-based DPC:** an optimization-based DPC controller without workflow design is deployed in the same PC. Different with the group of native DPC, this DPC controller is implemented by solving optimization problem [31].

Remark 4: By adding suitable magnitude limits, the analytical-solution-based DPC used in the first four methods could deal with constraints problems, and has been applied in many practical constrained scenarios [24], [32], [33], [34]. In the optimization-based DPC method, the constraints are handled by solving optimization problem. The analytical-solution-based DPC method is more computation-efficient with limited performance losses. The comparison simulations and discussions are provided in Section VII-F.

C. Brief Results of the Three Control Examples

There are three control examples in this section, which are the ball-beam system, vehicle tracking system and high-dimension numerical system. The former two control examples were conducted to detect the computation efficiency of the real-time control performance by the proposed workflow-based DPC methods and system. The third control example is a 78-dimension numerical system, of which the structure of the state-space model is built according to a power network system. This control example is applied to detected the computation efficiency for the high-dimension system.

1) **Control example 1:** the regulating problem of ball-beam system. The DPC workflow with MPT = 4 was applied in the second group of cloud resource pool. By using the workflow-based method, the computation time of cloud controller was reduced from 25.1917 ms to 13.8064 ms, of which the percentage of reduction was 45.19%.

2) **Control example 2:** the tracking problem of vehicle system. The DPC workflow with MPT = 4 was applied in the second group of cloud resource pool. When the speed was set as 30 km/h, the computation time was reduced from 35.2602 ms to 9.0432 ms. The total time delay, consisting the cloud-edge round-trip communication delay, was reduced from 47 ms to within 20 ms, which is the preset control period. As the result, the native DPC methods could not control the vehicle, while the workflow-based DPC with DOB achieved the stable control performance.

3) **Control example 3:** the control problem of high-dimension numerical system. The DPC workflow with MPT = 10 was applied in the third group of cloud resource pool. The evaluation was repeated with different numbers of the retained singular values, which is another form of determining the truncation precision $\epsilon_1$. The computation time was reduced from 367.4199 ms to 54.7559 ms, of which the percentage of reduction was 85.10%. The relationship between the computation efficiency and the number of the retained singular values, and that between the control performance and the number of the retained singular values were also discussed.

D. Control Example 1: Ball-Beam System

The first control example is the ball-beam system. This example is designed to detect the computation efficiency and control performance for the real-time regulating problem, which means controlling the system output to a constant reference value.

1) **System Model:** The structure of the ball-beam system is shown in Fig. 14. The main variables and parameters of this ball-beam system are listed in Table VI. By choosing the ball position $\gamma$ and beam angle $\theta$ as the generalized position coordinates for this system [35], the Lagrangian equation of motion is given by:

$$0 = \left(\frac{J_b}{R^2} + m\right)\ddot{\gamma} + mg \sin \alpha - m\dot{\gamma} \dot{\alpha}^2$$  \hspace{1cm} (64)

where $\dot{\gamma}$ is the acceleration of the ball, $\alpha$ is the beam angle and $\dot{\alpha}$ is the angular velocity of the beam angle. When this system approaches the stable point $\alpha = 0$, the local linearization of (64) can be obtained. Since $\alpha$ is small at this point, $\dot{\alpha} \approx 0$ and $\sin \alpha = \alpha$. Therefore, the linear approximation of this system is given as

$$\ddot{\gamma} = -\frac{mg}{\frac{J_b}{R^2} + m}\alpha.$$ \hspace{1cm} (65)

The beam angle $\alpha$ could be expressed related to the gear angle $\theta$ by approximating linear equation $\alpha L = \theta d$. Thus, the relation between $\ddot{\gamma}$ and $\dot{\theta}$ is

$$\ddot{\gamma} = -\frac{mdg}{L(\frac{J_b}{R^2} + m)} \dot{\theta}.$$ \hspace{1cm} (66)

In this system, the motor gear angle $\theta$ is set as the control input $u$. Letting the state be $x = [x_1, x_2, x_3, x_4]^T = [\gamma \dot{\gamma} \theta \dot{\theta}]^T$.
TABLE VI
SYMBOL DESCRIPTION AND PARAMETER

| Symbol | Description | Parameter |
|--------|-------------|-----------|
| \( L \) | Length of the bean | 0.4 m |
| \( d \) | Radius of the gear | 0.04 m |
| \( R \) | Radius of the ball | 0.015 m |
| \( J_b \) | Moment of inertia of the ball | \( 9.9 \times 10^{-6} \) kg \cdot m^2 |
| \( m \) | Mass of the ball | 0.11 kg |
| \( g \) | Acceleration of gravity | 9.81 m/s^2 |
| \( P_N \) | Support force on the ball | — |

and output be \( y = \gamma \), i.e., the position of the ball, the state-space model is obtained as

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3 \\
\dot{x}_4 \\
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & -mg/L(2J+m) & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
1 \\
1 \\
\end{bmatrix} u \quad (67)
\]

\[
y = \begin{bmatrix}
1 & 0 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
\end{bmatrix} . \quad (68)
\]

The initial data was provided by PID algorithm with \( K_P = 9.0, K_I = 3.0 \) and \( K_D = 7.5 \). The input is the motor gear angle \( \theta \), and is constrained by \(-0.1 < \theta < 0.1\). The parameters of the proposed method were set as \( N = 30, j = 1500, \lambda = 0.031, \epsilon = 10^{-15} \) and \( L = 50 \). Both of the discretization period and control period were set as 20 ms. The reference output in PID stage was set as 0.2 m. When the control is switched to the DPC stage, the reference output changed to 0.1 m.

2) Computation Efficiency: The computation delays and total time delays in the cloud controller are recorded in Table VII. The results of the computation delay could be divided into two classes. The first class consists of the DPC and the native DPC with time delay compensator. Since the calculation mode of both the two methods are centralized, the computation times of the two methods were similar, which were 25.1917 ms and 25.3186 ms, respectively. The second class is made up of the workflow-based DPC and the workflow-based DPC with DOB. The computation delays of the two methods were 13.8064 ms and 13.5918 ms, respectively. Since the computation efficiencies of the same computation mode were similar, we calculated the reduced proportion by the native DPC method without time delay compensator and the workflow-based DPC without DOB-based compensator, of which the result was 45.19%. The computation delays were also shown in Fig. 15. From this figure, the peak burrs occurred both in the first two methods. But in the last two methods, the recorded computation delays were more stable. As for the total time delays, which consists of the computation delay, the results of the above five methods are 30.8195 ms, 35.8021 ms, 35.7958 ms, 23.2867 ms and 23.3095 ms, respectively. Since there is no computation delay in PC-based control, the total time delay is equal to the computation delay. Furthermore, one can find that because of the existence of computation delay, the total time delay of native DPC method is larger than that of PC-based native DPC method. The workflow-based DPC method receives the smallest total time delay.

3) Control Performances: The control performances of the ball-beam system is shown in Fig. 16. The result shows that the two workflow-based methods obtained better performances than the native methods. In detail, the two native DPC had relatively high overshoot because of the affect of high time delays. Compared with the native method, the method with time delay compensator could bring some

Fig. 15. Computation delays in cloud controller of the ball-beam system.

Fig. 16. Control results of the ball-beam system.
improvement of the control performance. But since the time delays had not been reduced, the improvement of control performance was limited. The two workflow-based methods actively reduced the computation delay, and so obtained more obvious improvement. In the workflow-based DPC method, some low-value information were lost in the distributed SVD process. Thus, the DOB was deployed to estimate the uncertainty brought by the truncation operation. As a result, the workflow-based DPC with DOB achieved the best control performance.

E. Control Example 2: Vehicle Tracking System

The second control example is the trajectory tracking problem of vehicle system, which is to control the vehicle position as the varying reference signals. In this example, the problem of trajectory tracking control is studied based on the kinematic error model [36].

1) System Model: As shown in Fig. 17, define the X-axis of the inertial coordinate system points to the east, the Y-axis points to the north, and the Z-axis is positive upward. In the meanwhile, we define the x-axis of the vehicle coordinate system points to the front of vehicle, and the y-axis points to the left of vehicle. Then, the vehicle speed at the rear axle center is defined as $v_r$. The yaw angle of vehicle $\phi$ is defined as the angle between the X-axis of the inertial coordinate system and the x-axis of the vehicle coordinate system, where the counterclockwise direction is positive. ($X_r, Y_r$) and ($X_f, Y_f$) represent the coordinates of the rear axle center and the front axle center in the inertial coordinate system, respectively. $l$ is defined as the wheel base and $R$ is the instantaneous steering radius of the rear axle center. $\delta_f$ is defined as the drift angle of the front wheel.

The vehicle speed at the rear axis $v_r$ is calculated by

$$v_r = \dot{X}_r \cos \phi + \dot{Y}_r \sin \phi. \quad (69)$$

The kinematic constraints of the front and rear axes are defined as

$$\begin{align*}
\dot{X}_f &\sin(\phi + \theta_f) - \dot{Y}_f \cos(\phi + \theta_f) = 0 \\
\dot{X}_r &\sin \phi - \dot{Y}_r \cos \phi = 0
\end{align*} \quad (70)$$

Combining the equations (69) and (70), we have

$$\begin{align*}
\dot{X}_r &= v_r \cos \phi \\
\dot{Y}_r &= v_r \sin \phi
\end{align*} \quad (71)$$

Based on the geometric relationships of the front and rear wheels, we could obtain

$$\begin{align*}
X_f &= X_r + l \cos \phi \\
Y_f &= Y_r + l \sin \phi
\end{align*} \quad (72)$$

Substituting (71) and (72) to (70), we could obtain the yaw velocity $\omega$ as

$$\omega = \frac{v_r}{l} \tan \delta_f. \quad (73)$$

In the meanwhile, based on $\omega$ and $v_r$, the steering radius $R$ and the drift angle of the front wheel $\delta_f$ could be obtained as

$$\begin{align*}
R &= \frac{v_r}{\omega}, \\
\delta_f &= \arctan(\frac{l}{R}).
\end{align*} \quad (74)$$

Then, based on the above kinematic relationships of vehicle system, the kinematic error model is derived, of which the schematic diagram is provided in Fig. 18. The distance error $e_d$ is defined as the difference between the rear axle center and its projection point on the center line of the road, where the left side of the road is the positive direction. The angle error $e_\phi$ is defined as $e_\phi = \phi - \phi_{road}$, where $\phi_{road}$ is the angle between the tangential direction of the center line of the road and the X-axis of the inertial coordinate system and the counterclockwise direction is positive.

Assuming $P_1$ is the projection point of the rear axle center on the center line of the road, $s$ is the arc length between $P_1$ and a reference point $P_0$ on the road, and $k_{ref} = \frac{1}{l}$ is the curvature at the $P_1$ location. Then, the moving speed of the $P_1$ along the center line of the road $\dot{s}$ could be provided as

$$\dot{s} = \frac{1}{1 - k_{ref} e_d} [v_x \cos(e_\phi) + v_y \sin(e_\phi)] \quad (75)$$

where $v_x$ and $v_y$ are the components of the vehicle speed at the rear axle center $v_r$ along the x-axis and y-axis of the vehicle system, respectively.

The equations of trajectory tracking error could be expressed as

$$\begin{align*}
\dot{e}_\phi &= \dot{\phi} - k_{ref} \dot{s}, \\
\dot{e}_d &= v_x \sin(e_\phi) + v_y \cos(e_\phi).
\end{align*} \quad (76)$$

Then, based on the hypothesis of small angle ($\sin(e_\phi) \approx e_\phi, \cos(e_\phi) \approx 1$) and letting $k_{ref} e_d \approx 0$, the equations of trajectory tracking error could be simplified as

$$\begin{align*}
\dot{e}_\phi &= \dot{\phi} - k_{ref}(v_x + v_y e_\phi) \\
\dot{e}_d &= v_x e_\phi + v_y.
\end{align*} \quad (77)$$
Ignoring the lateral speed of the vehicle $v_y$, we could obtain

$$\begin{bmatrix} \dot{e}_\phi \\ \dot{e}_d \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ \frac{v_x}{v} & 0 \end{bmatrix} \begin{bmatrix} e_x \\ e_d \end{bmatrix} + \begin{bmatrix} \frac{v}{v} \\ 0 \end{bmatrix} \tan(\delta_f) + \begin{bmatrix} 0 \\ -v_x \end{bmatrix} k_{ref} \tag{78}$$

Based on (73) and (74), we could obtain $k_{ref} = \tan(\frac{\delta_{ref}}{\pi})$, where $\delta_{ref}$ represents the forward control variable provided by the reference trajectory. Letting $u_1 = \tan(\delta_{ref}) \approx \delta_{ref}$, $u_2 = \tan(\delta_{ref}) \approx \delta_{ref}$, $\xi = [e_x, e_d]^T$, the system (78) could be written as

$$\dot{\xi} = A\xi + B_1 u_1 + B_2 u_2 \tag{79}$$

where $A = \begin{bmatrix} 0 & 0 \\ \frac{v}{v} & 0 \end{bmatrix}$, $B_1 = \begin{bmatrix} \frac{v}{v} \\ 0 \end{bmatrix}^T$, $B_2 = \begin{bmatrix} 0 \\ -v_x \end{bmatrix}^T$.

In this control example, two groups of evaluations were conducted with different vehicle speeds $v_x$, which were 20 km/h and 30 km/h. The vehicle is controlled to track the circle trajectory, of which the steering radius $R$ was 42 m. Thus, the curvature of the trajectory $k_{ref}$ was set as 0.024 m$^{-1}$. The wheel base $l$ was set as 0.5 m. The initial data was provided by LQR algorithm with $Q = 20 \cdot I_2$, $R = I$. The input is the drift angle $\delta_f$ of the front wheel, and is constrained by $-\frac{\pi}{4} < \delta_f < \frac{\pi}{4}$. Both of the discretization period and control period were set as 20 ms. The parameters of the proposed method were set as $N = 20$, $j = 1000$, $\lambda = 0.0041$ and $L = -0.2$.

2) Computation Efficiency: Like the control example of the ball-beam system, we recorded the computation times

| Method | Computation delay | Total time delay |
|--------|-------------------|-----------------|
| PC-based native DPC | 93.3217 | 93.3217 |
| Native DPC | 35.7293 | 49.2032 |
| Native DPC with time delay compensator | 36.2084 | 49.4660 |
| Workflow-based DPC | 26.4251 | 35.3887 |
| Workflow-based DPC with DOB | 26.0765 | 36.1280 |

TABLE VIII
RECORDED COMPUTATION DELAYS AND TOTAL TIME DELAYS
OF CONTROL EXAMPLE 2 (20 KM/H, $\epsilon_1 = 10^{-4}$) (MS)
of the four methods on the two groups of evaluations. In the evaluation of $v_r = 20$ km/s, the truncation precision $\epsilon_1$ was set as $10^{-4}$ and the corresponding results are provided in Table VIII. The computation delays of the two native DPC methods were similar, which were 35.7293 ms and 36.2084 ms, respectively. The computation times of the two workflow-based DPC methods were 26.4251 ms and 26.0765 ms, respectively. Compared with the native DPC method, the workflow-based DPC method reduced the computation time by 25.43%.

Then, the vehicle speed $v_r$ was adjusted to 30 km/s. The truncation precision $\epsilon_1$ was reduced to $10^{-2}$ for further improving the computation efficiency. The recorded results of the computation delays and total time delays were provided in Table IX. The computation times of the two native DPC methods were similar with these of 20 km/s, which were 35.2602 ms and 25.2561 ms, respectively. The computation times of the two workflow-based DPC methods were 9.0432 ms and 9.1726 ms, respectively. Compared with the native DPC method, the workflow-based DPC method reduced the computation time by 74.35%.

In addition, the computation times were also shown in Fig. 19(i) and Fig. 19(k). The blue line represents the computation time of the native DPC method, and the red line means that of the workflow-based DPC method. The computation times of the workflow-based DPC method were more stable than these of the native DPC method. Then, since the evaluations were conducted in the cloud-edge collaborative control scheme, the round-trip communication delays also existed. In Fig. 19(j), the total delays consisting of the computation and the round-trip communication delays were provided, with the vehicle speed $v_r = 20$ km/s. The total delay of the two native DPC methods were 49.2032 ms and 49.4660 ms, and that of the two workflow-based DPC methods were 35.3887 ms and 36.1280 ms. In Fig. 19(l), the total delays were shown, of which the vehicle speed $v_r$ was set as 30 km/s. The total delay of the native DPC method was similar with that of 20 km/s. The total time delays of the two workflow-based DPC methods were reduced significantly, which were 19.8524 ms and 18.3967 ms. In the most sampling points, the total delays were lower than 20 ms, the control period. That means the cloud control signals could be received in the required time. The improvement of the control performance would be also brought, of which the detailed results and analyses were provided in the below.

3) Control Performances: The evaluations of vehicle trajectory tracking were divided into the groups of 20 km/s and 30 km/s. The control performance of $v_r = 20$ km/s was provided from Fig. 19(a) to Fig. 19(d). The results show that in 20 km/s, all the methods could obtain desirable control performance. It states that in the relatively low-speed real-time scenario, a certain amount of time delays could be tolerated, and the non-workflow-based DPC controller was enough to carry out the control missions.

However, when the vehicle speed $v_r$ increased to 30 km/s, the native DPC controller could not guarantee a stable and desirable control performance. As shown in Fig. 19(e), after the data preparation stage being finished, the vehicle became out of control quickly and veered off the circle trajectory. In Fig. 19(f), the native DPC with time delay compensator had not bring improvement on the control performance. As for the workflow-based DPC methods, Fig. 19(g) shows the cloud controller gradually obtained the described control performance. The reason is that the cloud control signals could arrived at the controlled plant in the required time. But in the beginning stage, some shakes occurred in the moving curve, since the truncation precision $\epsilon_1$ was set relatively low and the control performance was affected.

Therefore, the DOB was designed here to reduce the negative influence brought by the accuracy loss. As shown in Fig. 19(h), the workflow-based DPC with DOB received the best performance in the whole process. When the cloud controller was switched from LQR algorithm to the workflow-based DPC method with DOB, the shakes were indeed small and even did not appear. That means the switching process was smooth and stable. The cloud-edge composite DPC signals were calculated fast and accurately. As the result, the vehicle was also controlled to track the circle trajectory fast and well.

**F. Control Example 3: High-Dimension Numerical System**

The third control example is a 78-dimension numerical system, of which the structure of the state-space model is built based on IEEE 10-generator 39-bus power system [37], [38]. This control example is applied to detected the computation efficiency for the high-dimension system.

1) System Model: We consider the classical linearised synchronous machine model for each node of the power network. The behaviour of bus $i$ could be described by the so-called swing equation:

$$m_i \ddot{\theta}_i + d_i \dot{\theta}_i - P_{mi} = - \sum_{j \in N_i} P_{ij},$$

where $\theta_i$ is the phase angle of bus $i$, $m_i$ and $d_i$ are the inertia and damping coefficients, respectively, $P_{mi}$ is the mechanical input power and $P_{ij}$ is the active power flow from buses $i$ to $j$. Assuming that there are no power losses, neglecting ground admittances, and letting $V_i = |V_i|e^{j\phi}$ be the complex voltage of bus $i$, the active power flow between buses $i$ and
The state space model has been uploaded to Github.

Nodes were referred to [39] and [38]. Then, a 78-dimension connected to bus network and the undirected edge \( |L_i \) and \( |P_{ij} \) are the Laplacian matrix of graph \( G(V_{P}, E) \) with \( N = |V_{P}| \) nodes. Each node corresponds to a bus in the power network and the undirected edge \( (i, j) \in E_{P} \), if bus \( i \) is connected to bus \( j \) with edge weight \( k_{ij} \) for all \( (i, j) \in E \).

The parameters including the the relationships between the nodes were referred to [39] and [38]. Then, a 78-dimension numerical system with 10-input and 10-output is obtained, of which the state space model has been uploaded to Github.  

\[ P_{ij} = k_{ij} \sin(\theta_i - \theta_j), \]  
where \( k_{ij} = |V_i||V_j|b_{ij} \) and \( b_{ij} \) is the susceptance of the power line connecting buses \( i \) and \( j \). Since the phase angles are close, we could provide the linearised equation of the dynamics of bus \( i \):

\[ m_i\ddot{\theta}_i + d_i\dot{\theta}_i = -\sum_{j \in N_i} k_{ij}(\theta_i - \theta_j) + P_{mi}. \]  

Letting \( x = [\theta_1, \theta_2, \ldots, \theta_N, \dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_N]^T \) and \( u = [P_{m1}, P_{m2}, \ldots, P_{mN}]^T \), we obtain

\[ \dot{x} = Ax + Bu + H\alpha, \]  
where

\[
\begin{align*}
A &= \begin{bmatrix} 0_N & I_N \\ -ML & -DM \end{bmatrix}, \\
B &= \begin{bmatrix} 0_N \\ M \end{bmatrix}, \\
M &= \text{diag}(\frac{1}{m_1}, \frac{1}{m_2}, \ldots, \frac{1}{m_N}), \\
D &= \text{diag}(d_1, d_2, \ldots, d_N),
\end{align*}
\]

and \( L \) is the Laplacian matrix of graph \( G(V_{P}, E) \) with \( N = |V_{P}| \) nodes. Each node corresponds to a bus in the power network and the undirected edge \( (i, j) \in E_{P} \), if bus \( i \) is connected to bus \( j \) with edge weight \( k_{ij} \) for all \( (i, j) \in E \).

As analysed in Remark 3, there exist large amount of the singular values in the same order of magnitudes. If we adjust the truncation accuracy by increasing or decreasing \( \epsilon_1 \), the change of the upper bounds of the uncertainty would be large. Therefore, in this control example, the number of the retained singular values is used to replace \( \epsilon_1 \) to adjust the truncation accuracy. Then, the relationship between the computation efficiency and the number of the retained singular values, and the relationship between the control performance and the number singular values would be discussed. The evaluation was conducted in the non-real-time scenario, which means that only after the calculation of the control variable being finished, the controlled plant would be updated.

2) Relationship Between the Computation Efficiency and the Number of Retained Singular Values: Six groups of evaluations were carried out to detect the computation efficiency and control performance, when different number of singular values were retained. The recorded results of this control example is provided in Table X. The baseline represents the native DPC method being conducted in a single standard cloud server with 4 CPU and 8 GB memory. Besides, five groups of experiments were also conducted, of which 100, 50, 20, 10 and 5 singular values were retained, respectively. From the recorded results, the computation delay of the baseline was 367.4199 ms. With the decreasing of the retained singular values, the computation efficiency increased. When only 5 singular values were retained, the computation delay became 54.7559 ms. Compared with the baseline, the computation delay reduced 85.10%. It states that the workflow-based DPC could improve the computation efficiency significantly for the high-dimension system.

3) Relationship Between the Control Performance and the Number of Retained Singular Values: On the other hand, the

| Number of retained singular values | Baseline | 100 | 50 | 20 | 10 | 5 | (5 with DOB) |
|-----------------------------------|----------|----|----|----|----|----|----------------|
| Computation delay (ms)            | 367.4199 | 261.1135 | 168.9818 | 151.2747 | 128.8647 | 54.7559 | 55.0891 |
| Reduction proportion of computation delay | \( \backslash \) | 28.93% | 54.01% | 58.83% | 64.93% | 85.10% | 85.01% |
| Average control error             | 1.0550 \times 10^{-2} | 1.4418 \times 10^{-2} | 1.4458 \times 10^{-2} | 1.4486 \times 10^{-2} | 1.4934 \times 10^{-2} | 1.5528 \times 10^{-2} | 1.4201 \times 10^{-2} |

![Fig. 20. Relationship between control performance and the number of retained singular values.](image-url)
The control performances of the native DPC and the workflow-base DPC with 100, 20, 5 retained singular values are presented in Fig. 20. When 100 and 20 singular values were retained, the numerical system still could be controlled to the reference signal relatively accurately and steadily. But the control performances of the two groups were worse than the native DPC because of the losses of computation accuracy. As for the group of retaining 5 singular values, though there existed certain differences from the reference signal, the output had not diverged. The remaining differences could be reduced by the DOB-based compensator, of which the UUB stability has been proved. As shown in Fig. 21, the control performance of the case retaining 5 singular values was improved significantly by the edge DOB-based compensator. It states that the computation efficiency, control performance, truncation accuracy and DOB-based compensator should be considered in the meanwhile, to guarantee the fast, stable and high-quality control services in practical applications.

To further analyze the trade-off, similarly with [40], we introduce a metric to describe the average control error, which is

\[ e_{ac} = \frac{1}{MT} \sum_{i=1}^{M} \sum_{j=1}^{T} |y_i(j) - r_{f,i}| \]

where \( M \) is the number of component of output \( y \), \( T \) is the number of time instants, and \( y_i(j) \) is the output value of \( i \)-th component of \( y \) at time instant \( j \). From the simulation results, the outputs become stable close to the reference \( r_{f,i} \) at the 500-th time instant of DPC stage, and consequently \( T \) is set as 500. From the structure of the control numerical example, the number of component of output \( y \) equals to 10, i.e., \( M = 10 \).

Then, the results are shown in Table X and Fig. 22. It is obvious that with the decreasing the number of retained singular values, the computation delay decreased 85.10%, and the average control error increased from \( 1.0550 \times 10^{-2} \) to \( 1.5528 \times 10^{-2} \). The proper decision of truncation operation is dependent on the requirement of specific control problem. For example, if higher control accuracy is needed, the truncation operation should be conducted mildly and more information should be kept. If certain degree of control accuracy loss could be tolerated and computation efficiency is critical, one can truncate more singular values to further improve computation efficiency.

Particularly, according to Table X and Figure 22, though the computation delay was decreasing significantly from the group of retaining 100 singular values to the group of retaining 20 singular values, the results of average control accuracy were close. If the average control error around \( 1.44 \times 10^{-2} \) is enough, we are better to choose a more efficient group. Besides, the DOB-based group received a computation delay close to the fast group. But the DOB-based compensator reduced the average control error from \( 1.5528 \times 10^{-2} \) to \( 1.4201 \times 10^{-2} \). The control accuracy was better than the groups retaining more singular values. It verifies that DOB could reduce the accuracy loss introduced by the truncation operation.

4) Comparisons With Optimization-Based DPC Method:
The comparisons on computation efficiency and control performance of the native analytical-solution-based and optimization-based DPC methods are conducted in this part. We add the input constraints, \([-0.1, 0.1]\), \([-0.01, 0.01]\), \([-0.001, 0.001]\) and \([-0.0001, 0.0001]\) on each input component, and compare the control performance \( J = (r_f - \hat{y}_f(k))^T (r_f - \hat{y}_f(k)) + u_f(k)^T \lambda u_f(k) \) of the two DPC methods.

The compared results are provided in Fig. 23. It shows that there were some losses of optimality on the metric of cost function, but the differences of the two DPC methods with constraints were very small. That means the analytical-solution-based DPC could deal with well the constraints problem with relatively small losses of optimality. On the other hand, the calculation of the optimization-based DPC method was more time-costing. We also recorded the execution times of the native analytical-solution-based DPC and optimization-based DPC methods on the same PC. The results are recorded in Table XI. It could be known that the analytical-solution-based DPC method has obvious advantages over the optimization-based DPC method on the computation efficiency. Based on this work, the workflow-based method of optimization-based DPC would be studied in the future.

Remark 5: However, the introduction of cloud computing also brings extra challenges for control systems. For examples, the network security problem needs be considered; the failure risk of cloud controller should be concerned; and the data privacy in cloud platform also can not be ignored. To solve the above problems, the researches on safety of cloud control...
system, cloud-edge switching control and encryption-based cloud control system have been noticed.

VIII. CONCLUSION

This paper proposes a cloud-edge collaborative containerised workflow-based fast DPC system with DOB, which could reduce the computation delay to within 10 ms for the real-time examples, and by at most 85.10% for the high-dimension example. This paper provides a novel framework of cloud control system, with the combination of cloud-edge collaborative control, workflow-based control and container technology. Although this is a preliminary work, based on its framework, more problems with values would be studied, such as the overall stability of the workflow-based DPC method, workflow-based model predictive control, and co-design of control algorithm and cloud resource elastic scheduling, etc.

REFERENCES

[1] Y. Xia, “From networked control systems to cloud control systems,” in Proc. 31st Chin. Control Conf., 2012, pp. 5878–5883.
[2] D. C. Marinescu, Cloud Computing: Theory and Practice. Orlando, FL, USA: Morgan Kaufmann, 2017.
[3] P. K. Senyo, E. Addae, and R. Boateng, “Cloud computing research: A review of research themes, frameworks, methods and future research directions,” Int. J. Inf. Manage., vol. 38, no. 1, pp. 128–139, Feb. 2018.
[4] A. B. Alexandru, K. Gatsis, Y. Shoukry, S. A. Seshia, P. Tabuada, and G. J. Pappas, “Cloud-based quadratic optimization with partially homomorphic encryption,” IEEE Trans. Autom. Control, vol. 66, no. 5, pp. 2357–2364, May 2021.
[5] T. Tanaka, M. Skoglund, H. Sandberg, and K. H. Johansson, “Directed information and privacy loss in cloud-based control,” in Proc. American Control Conf. (ACC), May 2017, pp. 1666–1672.
[6] G.-P. Liu, “Predictive control of networked multiagent systems via cloud computing,” IEEE Trans. Cybern., vol. 47, no. 8, pp. 1852–1859, Aug. 2017.
[7] Y.-Q. Xia, R. Gao, M. Lin, Y.-M. Ren, and C. Yan, “Green energy complementary based on intelligent power plant cloud control system,” Acta Autom. Sin., vol. 46, no. 9, pp. 1844–1868, Sep. 2020.
[8] Z. Wang, S. Yang, X. Xiang, A. Vasiljević, N. Mišković, and D. Nad, “Cloud-based mission control of USV fleet: Architecture, implementation and experiments,” Control Eng. Pract., vol. 106, Jan. 2021, Art. no. 104657.
[9] Y. Xia, “Cloud control systems,” IEEE/CIA J. Autom. Systica, vol. 2, no. 2, pp. 134–142, Apr. 2015.
[10] M. S. Mahmoud and Y. Xia, “The interaction between control and computing theories: New approaches,” Int. J. Automat. Comput., vol. 14, no. 3, pp. 254–274, Apr. 2017.
[11] Y. Xia, Y. Zhang, L. Dai, Y. Zhan, and Z. Guo, “A brief survey on recent advances in cloud control systems,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 69, no. 7, pp. 3108–3114, Jul. 2022.
[12] R. Gao, Y. Xia, L. Dai, Z. Sun, and Y. Zhan, “Design and implementation of data-driven predictive cloud control system,” J. Syst. Eng. Electron., vol. 33, no. 6, pp. 1258–1268, Dec. 2022.
[13] L. Ye, Y. Xia, L. Yang, and C. Yan, “SHWS: Stochastic hybrid workflows dynamic scheduling in cloud container services,” IEEE Trans. Autom. Sci. Eng., vol. 19, no. 3, pp. 2620–2636, Jul. 2022.
[14] J. Chen et al., “A parallel random forest algorithm for big data in a spark cloud computing environment,” IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 4, pp. 919–933, Apr. 2017.
[15] G. Xiao, J. Li, Y. Chen, and K. Li, “MaFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks,” J. Parallel Distrib. Comput., vol. 141, pp. 49–58, Jul. 2020.
[16] R. Gao, Y. Xia, G. Wang, L. Yang, and Y. Zhan, “Fast subspace identification method based on containerised cloud workflow processing system,” 2021, arXiv:2112.14349.
[17] W.-H. Chen, J. Yang, L. Guo, and S. Li, “Disturbance-observer-based control and related methods—An overview,” IEEE Trans. Ind. Electron., vol. 63, no. 2, pp. 1083–1095, Feb. 2016.
[18] J. Zhu, X. Li, R. Ruiz, and X. Xu, “Scheduling stochastic multi-stage jobs to elastic hybrid cloud resources,” IEEE Trans. Parallel Distrib. Syst., vol. 29, no. 6, pp. 1401–1415, Jun. 2018.
[19] H. Hao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, “Learning scheduling algorithms for data processing clusters,” in Proc. ACM SIGCOMM, Aug. 2019, pp. 270–288.
[20] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, “Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers,” IEEE Wireless Commun., vol. 24, no. 3, pp. 48–56, Jun. 2017.
[21] T. Goldschmidt, S. Hauck-Stattelmann, S. Malakuti, and S. Grünert, “Container-based architecture for flexible industrial control applications,” J. Syst. Archit., vol. 84, pp. 28–36, Mar. 2018.
[22] J. Mellado and F. Núñez, “A container-based IoT-oriented programmable logical controller,” in Proc. IEEE Conf. Ind. Cyberphys. Syst., vol. 1, Jun. 2020, pp. 55–61.
[23] R. Ranjan, I. S. Thakur, G. S. Aujla, N. Kumar, and A. Y. Zomaya, “Energy-efficient workflow scheduling using container-based virtualization in software-defined data centers,” IEEE Trans. Ind. Informat., vol. 16, no. 12, pp. 7646–7657, Dec. 2020.
[24] Y. Xia, W. Xie, B. Liu, and X. Wang, “Data-driven predictive control for networked control systems,” Inf. Sci., vol. 235, pp. 45–54, Jun. 2013.
[25] B. Huang and R. Kadali, Dynamic Modeling, Predictive Control and Performance Monitoring: A Data-Driven Subspace Approach. Edmonton, AB, Canada: Springer, 2008.
Runze Gao received the B.S. degree from the Beijing Institute of Technology, China, in 2017, where he is currently pursuing the Ph.D. degree in control science and engineering. His research interests cover cloud control systems, model predictive control, and data-driven predictive control.

Qiwen Li received the B.S. degree from the Beijing Institute of Technology, China, in 2020, where he is currently pursuing the M.S. degree. His research interests include cloud native technology, cloud control systems, and related privacy-preserving issues.

Li Dai received the B.S. degree in information and computing science and the Ph.D. degree in control science and engineering from the Beijing Institute of Technology, Beijing, China, in 2010 and 2016, respectively. She is currently an Associate Professor with the School of Automation, Beijing Institute of Technology. Her research interests include model predictive control, distributed control, data-driven control, stochastic systems, and networked control systems.

Yufeng Zhan received the Ph.D. degree from the Beijing Institute of Technology (BIT), Beijing, China, in 2018. He is currently an Assistant Professor with the School of Automation, BIT. Prior to joining BIT, he was a Post-Doctoral Fellow with the Department of Computing, The Hong Kong Polytechnic University. His research interests include networked systems, game theory, and machine learning.

Yuanqing Xia received the M.S. degree in fundamental mathematics from Anhui University, Hefei, China, in 1998, and the Ph.D. degree in control theory and control engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001. He was a Post-Doctoral Research Associate with the Institute of Systems Science, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, from 2002 to 2003. From 2004 to 2006, he was with the University of Glamorgan, Pontypridd, U.K., as a Research Fellow. From 2007 to 2008, he was a Guest Professor with Innsbruck Medical University, Innsbruck, Austria. He is currently a Professor with the School of Automation, Beijing Institute of Technology, Beijing. His research interests are cloud control systems, networked control systems, robust control and signal processing, active disturbance rejection control, unmanned system control, and flight control.