Received May 30, 2019, accepted June 21, 2019, date of publication June 26, 2019, date of current version July 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2925114

A New Data Processing Architecture for Multi-Scenario Applications in Aviation Manufacturing

WEI WANG, LEI FAN, PU HUANG, AND HAI LI
School of Mechanical and Electrical Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

Corresponding author: Wei Wang (wangwhit@163.com)

This work was supported by the National Science and Technology Major Project under Grant 2017ZX04002001.

ABSTRACT The development of industry 4.0 has spurred the transformation of traditional manufacturing into modern industrial Internet-of-Things. The most notable feature during this transition is the improvement of digitization and intelligence based on the massive data drives. In such a data-driven environment, the processing, storage, and utilization of the industry data get more and more important. Usually, the traditional data processing architecture runs as a one-way streamline, which cannot adapt to the different requirements of the multi-scenario application. This paper proposed a new industrial big data processing architecture called Phi architecture, which can realize many functions such as batch data processing and stream data processing, distributed storage and access, and real-time control. Compared with other data processing architecture, the Phi architecture combined with edge computing and feedback control has the ability to deal with the different demands in aviation manufacturing. Next, the new architecture is designed for microservices pattern, which improves the flexibility and stability of the architecture, and makes it independent operated in multi-scenarios, such as state monitoring of workshop, adaptive data acquisition, feedback control, and user-oriented information classification. As a proof of concept, the architecture has been tested in a simulation digital manufacturing workshop. The results verify the improved effectiveness of the Phi architecture on the data feedback control and real-time processing. And, the development of microservices architecture greatly improves the efficiency, adaptability, and extensibility of the manufacturing process.

INDEX TERMS Data processing architecture, real-time feedback, edge computing, microservices, multi-scenario application.

I. INTRODUCTION

In recent years, the rapid development of Industry 4.0 has had a global impact on the industry and the economy [1]. In the context of Industry 4.0, data, which is the main carrier of the digital environment, encompasses large amounts of useful information, including processing statuses, working conditions, resource utilization rates and so on. This information can be applied to production monitoring, quality control, and used to optimize resource allocation. However, devices are usually equipped with multiple sensors; thus, gigabytes—or even terabytes—of data can be produced each day. Therefore, information technology (IT) infrastructure and data utilization architectures have vital influences on manufacturing system performance [2].

Given the continuous maturation of big data analysis (BDA) methodology and the rise of cloud and edge computing, considerable research has been conducted on how to combine cutting-edge IT techniques with manufacturing. These efforts have also spawned many new Industrial 4.0 theories, such as data-driven and cloud-based manufacturing. Simultaneously, new concepts and standards for manufacturing scenarios are continually being proposed, among which the concept of a cyber-physical system (CPS) plays an important role. A CPS manages the relationships between physical devices and the computing capabilities of interconnected systems [3], and it can be further developed to leverage big data, allowing a CPS to achieve the goal of intelligent, resilient and self-adaptable machines. Correspondingly, a 5-level

The associate editor coordinating the review of this manuscript and approving it for publication was Nagarajan Raghavan.

CPS structure (5C) has been proposed to provide guidelines for developing and deploying CPSs [4].

In terms of standards, ISA-95 and RAMI 4.0 are two compelling achievements. On one hand, the ISA-95 standard focuses on formalizing the interactions between manufacturing system and other business processes by specifying data flows and interfaces using enterprise modeling techniques [5]. On the other hand, the RAMI 4.0 standard is dedicated to describing the Industry 4.0 architecture from different dimensions. It uses three coordinate description models and represents the interactions between the dimensions [6].

To construct the complex systems enabled by the above concepts and standards, some advanced software development technologies such as the microservices framework have also been introduced into the Industry 4.0 research system. The microservices framework divides and encapsulates independent operations through functional blocking, and its modular nature can meet the requirements of many industrial scenarios [7].

A large number of experiments and applications have demonstrated that the technologies and architectures mentioned above achieve good results in common scenarios. However, aviation manufacturing, which is a technology-intensive industry typically equipped with high-end devices, focuses mainly on multi-variety specifications and has a small-batch production mode. These characteristics lead to a flexible manufacturing process but also introduce issues and challenges in the context of Industry 4.0 that should not be ignored. The following questions capture these challenges. 1) How should a big data processing architecture be designed to meet flexibility requirements? 2) How should the information generated by BDA be fed back to the device layer to dynamically adjust strategies such as data acquisition, fault warnings, and so on?

Lambda architecture (LA) [8] and edge computing are especially suitable for solving problems under these circumstances. Lambda architecture is widely used because it is capable of performing streaming and batch processing simultaneously for massive amounts of data. In addition, edge computing can be used in the device layer to achieve low latency and agile data processing. However, the problem is that there is no clear way to make them interactive.

This study explores how to achieve agile aviation manufacturing and makes three main contributions from the perspective of the implementation of Industry 4.0.

1. It combines edge computing and Lambda architecture to simultaneously address device layer data with low latency and process massive global data in a scalable fashion.

2. A feedback control loop is added in the proposed architecture between edge computing and LA; thus, the edge computing strategy can be time-varying, continuously and automatically optimized, and workpiece specific.

3. The proposed method is implemented and evaluated in an aviation manufacturing environment using a microservices approach to validate its viability and effectiveness.

The remainder of this paper is structured as follows. Section II presents related works concerning data processing and computing solutions. Section III presents the details of the Phi architecture, including its structural components, implementation scheme and microservices design. The application of Phi architecture in aviation manufacturing is discussed in Section IV. Related experiments and a results analysis are presented in Section V. Finally, conclusions are summarized in Section VI.

II. RELATED WORK

A. DATA PROCESSING SOLUTIONS

In the context of Industry 4.0, industrial data is characterized by diversity, high concurrency and low value, all of which challenge existing data processing solutions. According to prior research, the construction of CPSs may be one of the main influencing factors. ISA-95 and 5C [4] are architectures for implementing CPSs in manufacturing organizations; their goal is to improve and deeply integrate physical factories with cyber computational space and then to realize a transformation from data to information to value. To extend the existing 5C architecture, Jiang [9] proposed an 8C architecture (adding coalition, customer, and content) to assist in building CPSs for smart factories.

One goal of CPS construction is to break down the barriers of device independence and information islands to construct a unified data environment. However, this goal also directly increased the quantity and complexity of the underlying data in the factory. Nevertheless, these industrial data doubtlessly contain valid information, whether directly or indirectly, a concept that was well illustrated in [10]–[12] from the perspectives of manufacturing equipment, workpiece, and internal and external enterprise communications. Therefore, the key lies in the choice of data processing architecture or methods. Some data processing method problems are discussed in [13], [14], such as processing delays, incomplete data, and valuable knowledge extraction.

Currently, the most common data processing tools include Hadoop, Spark, Storm, and Kafka [15], but those tools have some limitations. Hadoop is usually applied to batch processing, but it does not meet the requirements for stream processing. Spark is a hybrid data processing tool that can address both batch and stream processing; however, it also has memory occupancy and processing delay problems. Storm is suitable only for stream data processing, and Kafka is mainly used for rapid message distribution, to provide fast processing support. Due to these limitations, some data processing architectures that combine the above tools have been proposed.

The Lambda and Kappa architectures (an alternative scheme to Lambda) have been widely applied in recent years due to their better real-time and batch data processing performances [16], [17]. Hasani *et al.* [18] used the Lambda architecture to build a processing platform that achieved processing and analysis of real-time big data and solved the problem of performing the real-time calculation of arbitrary functions on arbitrary data. By combining the Lambda architectural design pattern with database management, cost models, query management and cloud computing, Kiran *et al.* [19] presented a cost-effective general architecture that can be applied to big data and online data processing. A generic, scalable and fault-tolerant data processing architecture called Ahab was proposed by Vögler *et al.* [20]. Ahab combined the philosophy of the Lambda architecture with the cloud platform to achieve online and offline analysis capabilities. Ahab can also independently optimize and deploy the computing load. Batyuk and Voityshyn [21] applied Lambda architecture to address the streaming data problem of a real-time monitoring platform, which achieved near real-time processing. To apply Lambda architecture in a monitoring task, Suthakar *et al.* [22] used the Apache Spark ecosystem to optimize the Lambda architecture. This approach increased the scalability of stream data processing and achieved better for cumulative computation performance.

The works mentioned above serve as demonstrations for applying LA in end-to-end data processing situations; however, it is the effective use of data processing results that provides the real value of data processing—especially the reverse use of results, which can enhance resource interactions and improve CPS construction; however, this aspect is not addressed by LA.

B. CLOUD COMPUTING AND EDGE COMPUTING

Cloud computing has become a primary data processing solution in recent years. Cloud computing integrates the discrete computing resources of enterprises to achieve centralized management and allocation, and it localizes data processing tasks to cloud computing platforms; allowing enterprises to meet their requirements for data processing efficiency and storage.

Numerous works have focused on cloud computing applications in manufacturing. For example, Vögler *et al.* [20] proposed a cloud-based, distributed, big data analytics framework by integrating the Lambda architecture that worked well in the IoT environment. Wan *et al.* [23] made some contributions to real-time and offline data processing by proposing a method that combined cloud computing and data processing algorithms. And Wang and Ranjan [24] also discussed the processing requirements of large distributed data sets in an IoT scenario, and settled on a scheme based on cloud computing. However, given the real-time data processing needs and feedback from different manufacturing scenarios, centralized data processing approaches work less well and introduce problems similar to those of the existing data processing architecture (Lambda, Kappa).

Moreover, the increasing data volume and number of computing missions also increase processing pressure on cloud platforms, which will gradually expose problems caused by large data processing loads and service response delays. These issues have also been raised by Satyanarayanan [25]. Corcoran and Datta [26] also realized that cloud platforms will be unable meet real-time operation and low latency demand requirements and suggested a mobile-edge computing solution that aims to extend cloud services to the edge of IoT. To address the above problems, Fu *et al.* [27] provided a solution that integrates edge computing and cloud computing and can better adapt to the challenges such as data processing and efficient data retrieval in industrial IoT. In addition, Georgakopoulos *et al.* [28] provided a roadmap of cloud computing with IoT for manufacturing that incorporated edge computing concepts.

Obviously, edge computing applications have worked well in industrial IoT. Compared with cloud computing, edge computing effectively alleviates the pressure on cloud data processing centers, reduces the complexity of data I/O processes in cloud computing centers, and improves the real-time responsiveness of services. The authors of [29]–[31] discussed the future development trends and opportunities of edge computing involving real-time data processing and services, low-latency control, flow control and so on. Additionally, Martín Fernández *et al.* [32] proposed an edge computing architecture focused on edge smart gateways that provide better control and isolation of running processes in IoT. Wan *et al.* [33] proposed an ELBS method based on fog computing with the goal of achieving a dynamic, flexible scheduling for smart factories. With fog computing support, instruction transmission delay was greatly reduced. This approach satisfied the real-time requirements of dynamic order analysis and equipment scheduling. Chen *et al.* [34] presented an edge computing architecture for IoT-based manufacturing and illustrated its roles in the fields of information interaction, data fusion and advanced analytics. They even provided a reference concerning the cooperation mechanism between cloud computing and edge computing. Sun and Ansari [35] and Yu *et al.* [36] also proposed application schemes for edge computing in the IoT, to further demonstrate the ability to construct a flexible supply of computing services and data stream processing.

Combined with the contributions and limitations of the above works, we were motivated to integrate edge computing into LA. This approach can both satisfy the data processing requirements and reduce computing center load to guarantee its performance. Moreover, capitalizing on processing results during the manufacturing process is the main concern. We provide the details of our approach in the next section.

III. PHI ARCHITECTURE: A NEW LAMBDA ARCHITECTURE IMPLEMENTATION

To expand the existing industrial data processing solutions, improve the processing effect and achieve effective utilization of data processing results, on the LA, we combine edge computing with a feedback loop, forming a new implementation of LA for aviation manufacturing called Phi architecture. The composition, characteristics and advantages of the Phi architecture are described in the following sections.

A. THE CPS OF AVIATION MANUFACTURING BASED ON 5C ARCHITECTURE

The complex manufacturing process and high manufacturing standards of aviation manufacturing also pose challenges

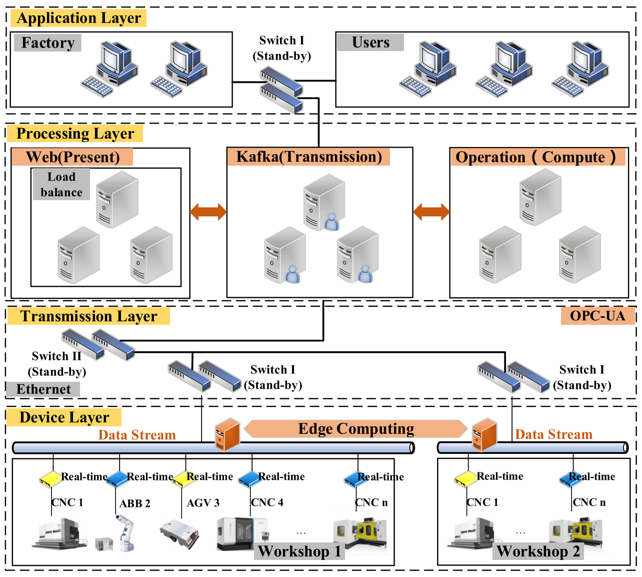


FIGURE 1. CPS framework for aviation manufacturing.

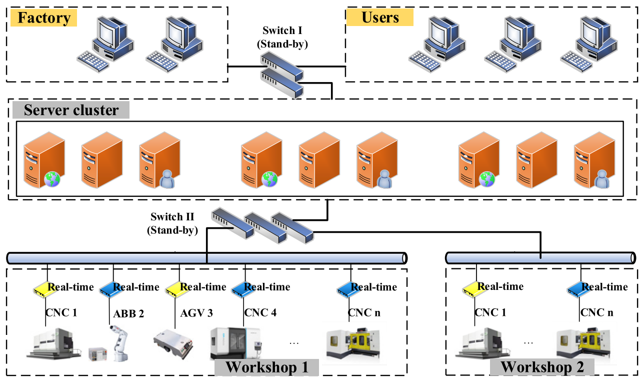


FIGURE 2. CPS framework of traditional manufacturing.

and requirements when construction its CPS. Rational CPS construction is important to an intelligent transformation and upgrading of industrial enterprises. In current CPS construction, 5C architecture is the most popular schema and consists of five levels: connection, conversion, cyber, cognition, and configuration. Combined with 5C architecture, we provide a corresponding CPS framework for an aviation manufacturing workshop.

The CPS framework for aviation manufacturing is composed of four main layers: a device layer, a transmission layer, a data processing layer, and an application layer. The details of this framework are shown in Fig. 1.

The device layer, which contains most of the resources and equipment of the workshop, is the main source of the workshop data (other sources include IT systems, parts, and tools). Under a corresponding data acquisition scheme, the data and information are obtained that form the main support for digitalization and visualization. The transport layer is a collection of various fieldbus protocols that are coordinated by the gateway to meet data transmission and interaction requirements. The processing layer is a server cluster that implements functions such as data processing and analysis, storage, and data transmission. The application layer, which includes the workshop management system and the user service terminal, is used to visualize the data collection process and the data processing results.

In contrast to the traditional industrial CPS framework (Fig. 2), the proposed CPS framework for aviation manufacturing makes the following improvements. The transport layer is further clarified: with the intelligent digital upgrade to aviation manufacturing, the continuous convergence of various heterogeneous devices, software and application systems in CPS continually increases the complexity of the communication environment. To achieve unified communications for different protocols, the corresponding communication standards and specifications (e.g. OPC-UA) must meet the interaction needs of underlying devices and enhance the interactivity between data. A unified data format is also conducive to improving transmission and processing efficiency. Moreover, the server cluster is divided into three functional components: web server, transmission server and operation server. The web server presents data processing results and workshop state information in a timely manner. The transmission server is equipped with message-oriented middleware with sufficient performance to support real-time data stream transmission. Edge computing is applied to data-collection end (device layer), which reduces unnecessary data transmissions, thus improving both computational efficiency and transmission speeds to achieve the real-time service requirement of aviation manufacturing.

In brief, this CPS framework for aviation manufacturing is better able to adapt to future development trends in aviation manufacturing, and—especially in the era of digitalization, intellectualization and big data—meet the needs of complex communication environments and the requirements for mass data acquisition, transmission and processing.

B. PHI ARCHITECTURE

Based on the CPS framework described above for aviation manufacturing, advances in aviation manufacturing, such as high-performance devices, auxiliary software and other functional modules, enable continuous integration and improvement. These will cause the CPS to become increasingly complex. Considering the data processing requirement we focus on in this paper, different devices and application system integrated into the CPS will cause a dramatic increase in data volume and cause the data set to become more diverse and complex. Obviously, the variety of data types, complexity of data streams and high concurrency of multiple data sources are big obstacles for data processing systems in an aviation manufacturing workshop.

To address the massive data generated during the manufacturing process, fulfill some real-time demands for computing services (e.g. motion control, flow control), and improve the efficiency of using data processing results, we propose an improved big data processing architecture for the aviation

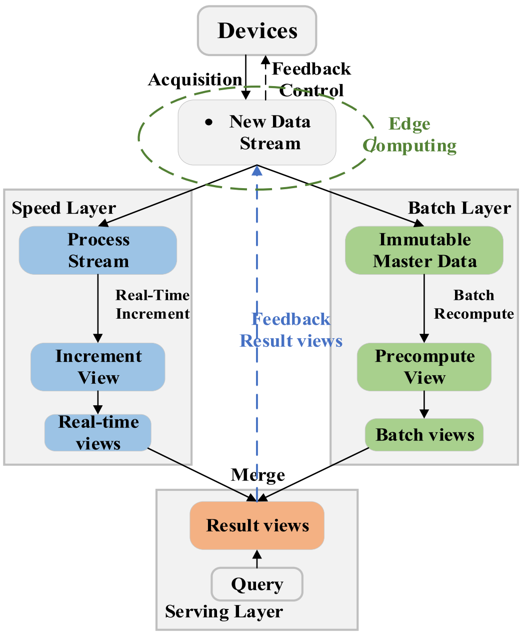


FIGURE 3. The Phi (Φ) architecture.

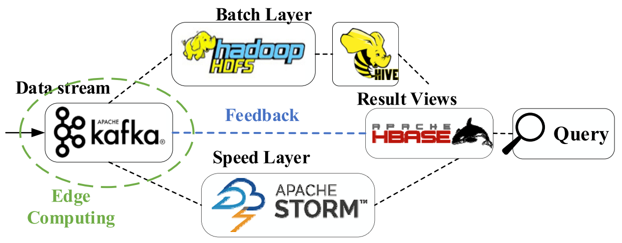


FIGURE 4. The software composition of the Phi architecture.

manufacturing, known as the Phi architecture, which is illustrated in Fig. 3.

The basic structure of the Phi architecture is quite similar to that of LA. To meet the batch processing and real-time processing requirements of massive amounts of data generated during the manufacturing process, Phi provides a batch processing module (Batch layer) and a real-time processing module (Speed layer). To provide query support for data processing views, it also provides a service layer. The detailed functions in these three layers are listed below.

1) Batch layer. This layer provides management and storage of the master data set. It periodically conducts simple pre-computing on the master data set (pre-computing arbitrary query functions). Processing immutable data sets off-line and storing the processing results in batch views can reduce the overall data volume and improve the real-time query performance for the data results.
2) Speed layer. During an actual manufacturing process, data is generated continuously, and these data must be processed in time to service results queries; however, the batch layer is usually applied to periodically calculate (recalculate) the main data set, which can easily affect system performance and cannot meet real-time demand requirements. However, instead of recalculating the entire master dataset, the speed layer can be used to process real-time data incrementally. Then, the processing results can be updated to the real-time view, meeting the real-time query requirement for viewing results.
3) Service layer. This layer is mainly used to respond to user requests by fusing the results of the real-time view and batch view into the final data set (usually the batch view), thus providing support for the final query service.

In addition, the Phi architecture integrates some new components for aviation manufacturing applications that are different from the Lambda architecture in certain aspects. One is the integration of edge computing, which offers real-time calculation at the network edge, and the other is a feedback loop used to achieve reverse operations by using the data processing results. The details of both benefits are listed below.

1) The integration of edge computing can balance the computing pressure in the cloud and support real-time responses by services such as low-latency control of servo motors, traffic monitoring, and flow control of the manufacturing data. This system is usually deployed on a host computer or industrial computer to process data such as abnormal device control or flow information from the manufacturing system.
2) The addition of feedback to the Phi architecture supports reverse operation and transmission to meet the needs of aviation manufacturing, for example, feedback control from the application layer to the device layer and information customization from the processing layer to the device and application layers.

In summary, the characteristics of the Phi architecture determine its application in areas related to customized production, especially for applications that require closed-loop and real-time control, such as intelligent manufacturing, processing and assembly, logistics and transportation. Combined with the applications in an aviation manufacturing workshop, we note that the Phi architecture is better able to adapt to the data processing and information transmission needs of the digital workshop. This aspect is important for big data processing in complex aviation manufacturing environments.

C. INTEGRATING PHI ARCHITECTURE AND BIG DATA PROCESSING TECHNOLOGIES

The Phi architecture can be integrated with Hadoop, Kafka, Storm (or Spark), HBase and other components to quickly build a big data processing system (Fig. 4). For example, we used Hadoop for the batch layer and Storm (or Spark) for the real-time processing layer in the same system. The incoming and outgoing data streams are handled by the Kafka platform which can support rapid transmissions with high throughput and low latency. We used Hive to build the data

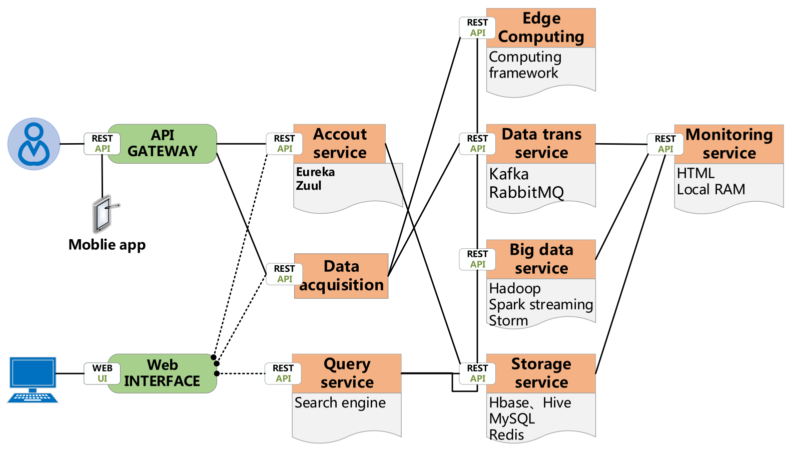


FIGURE 5. The design of Phi architecture based on microservices.

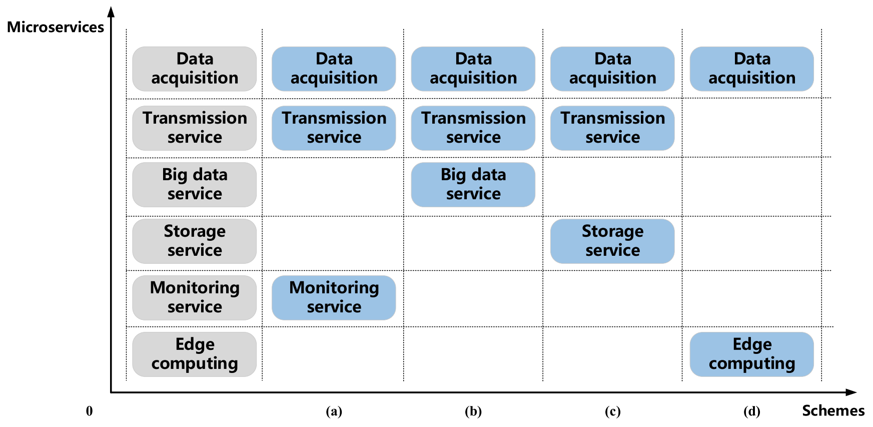


FIGURE 6. Microservices multi-scenario combination scheme.

warehouse, which offers functions such as data refining, query and analysis. Another database (HBase) is used to store the main data and the resulting views (merged batch and real-time views) to support the final query service. Edge computing surrounds the entire data process and is used to reduce the computing pressure on the data center. Services are initiated directly from the data end based on requirements, which reduces data transmission delay and meets real-time service requirements.

However, the integration of various equipment and auxiliary manufacturing systems in aviation manufacturing workshop CPSs, increases the difficulties of data processing system deployment, maintenance, sustainability and expansion. By using the microservice system development approach, the complex structure and functions of the system can be segmented and encapsulated, mutual interference between components can be reduced, and the composability of different microservices can be improved. The data processing system consists of several main modules: batch data processing, real-time data processing, result queries, edge data processing, data acquisition, storage and transmission.

As shown in Fig. 5, the data processing system based on Phi architecture is divided into multiple microservices, such as an API gateway, account service, data acquisition service, monitoring service, data transmission service, data processing service, edge computing service, data storage service and data query service that form the business functions. Such a microservice-based system is more flexible and supports modification and expansion according to user requirements.

Combining microservices to form applications is the primary benefit, because they can improve the system applicability in different scenarios, and abilities such as independent deployment and operation also help support that aspect. Based on the above service module and taking the aviation manufacturing workshop demo as an example, we list several different combinations below to suggest possible application scenarios, as shown in Fig. 6. The scheme (a) shows data collection and monitoring scenarios, scheme (b) presents data collection and processing scenarios, scheme (c) displays data collection and storage scenarios, and scheme (d) denotes real-time response scenarios for the data acquisition end. In addition, the data processing system can also be expanded to meet additional scenario applications for complex manufacturing processes.

IV. PHI ARCHITECTURE APPLICATION IN AVIATION MANUFACTURING

A. CHARACTERISTICS OF AVIATION MANUFACTURING

Aviation manufacturing has several characteristics that are different from traditional manufacturing, as discussed in the following four points.

1) Many parts are large and have complicated structures. As the performance of modern aircraft continues to increase, to achieve weight reduction and improve fatigue life, aircraft structural parts have becoming increasingly integrated, resulting in individual parts having increased size and complicated structures. Thus, higher requirements are continually proposed for parts processing.
2) Diversified materials. In order to achieve weight reduction and fuel economy, aircraft structural parts are developed from a variety of different materials, including aluminum alloys, titanium alloys and other composite materials. Moreover, the material ratios differ depending on the aircraft's performance requirements.
3) High value and high precision requirements. As a high value-added industry, aviation manufacturing not only involves expensive raw materials for parts and components, but also has relatively high processing costs, resulting in higher scrapping costs for parts.
4) Small batches of multiple classes. Manufacturing of modern aircraft structural components is a typical complex product that entails small batch production and even single-piece production modes. Achieving high efficiency, high quality and low-cost manufacturing in these modes is a huge challenge.

In summary, due to its unique characteristics, the aviation manufacturing must pay more attention to the flexibility, precision and yield of the production process than is required in other traditional manufacturing industries. Such restrictions require monitoring and preventive intervention in the production process.

B. MODEL TRAINING ALGORITHM

In the Industry 4.0 context, many ''data-driven'' algorithm applications have similar offline-model training processes. For example, in temperature compensation and tool wear prediction, it is necessary to first train the collected data to construct a model through a specific algorithm; then, the trained model is used to achieve the corresponding function. Therefore, the data-to-model step can be stylized and implemented using the same algorithmic logic. In subsequent implementations, the only adjustment needed is to apply different training algorithms to adapt to different application requirements. The offline model training step procedure is shown in Algorithm 1. In the algorithm, each real-time stream data instance from Kafka is read, verified, and then stored in *batch_data[N]*, where N represents the maximum temporary storage capacity. When the amount of data exceeds N, *batch_data[N]* is saved to the Hadoop file system HDFS. When the model-training condition is triggered, *model_training_trigger* becomes true, and any unsaved items in *batch_data[N]* are immediately saved to HDFS and the *training_algorithm* is launched. The model trained using the latest full set of data (*all_data*) denoted as model $model_{trained}$.

Algorithm 1: Offline Model Training Procedure

```
Input: Real-time stream data realtime-data
Output: Trained Model $model_{tratned}$
Initialization;
while runing do
    while sink_runing do
        i ← 0;
        // Clear the cached data clear batch_data[N];
        while $i < N$ do
            cur_data ← read from realtime_data;
            flag ← validate(cur_data);
            if flag then
                if model_training_trigger then
                    all_data ← read from HDFS;
                    sink_runing ← false;
                    break
                else
                    batch_data[i] ← cur_data;
                    $i \leftarrow i + 1$;
                end
            else
                runing ← false;
                sink_runing ← false;
                break
            end
        end
        batch-data[N] → HDFS;
    end
    // Custom training algorithm
    $model_{trained}$ ← training_algorithm (all_data);
    sink_runing ← true;
end
```

In the algorithm, $model_{trained}$ is used to represent the model trained by the algorithm *training_algorithm* based on the collected data. Different *training_algorithm* types, such as a support vector machine (SVM), neural network (NN) or decision tree (DT) and others. can be applied to complete various diagnostic and predictive functions.

C. TEMPERATURE COMPENSATION

As mentioned earlier, high precision machining is one of the manufacturing characteristics in the aviation manufacturing, but many factors exist that affect the processing accuracy, including wear, processes, spatial accuracy, thermal deformations, etc. Therefore, reducing those errors and improving the processing accuracy is one problem we face in this industry. Taking thermal compensation for thermal deformation of a

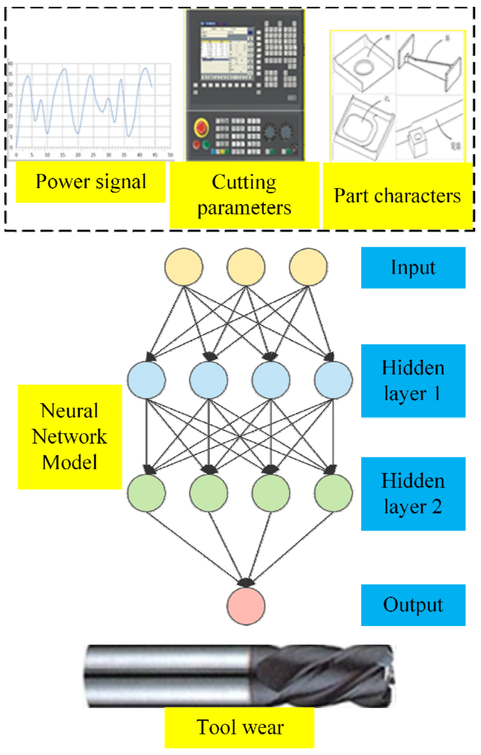


FIGURE 7. **The specific process for intelligent monitoring of tool wear.**

spindle as an example, a multivariate linear regression model is used in this case; the compensation process is shown in Algorithm 2. In the algorithm, the thermal error data of spindle is stored in the Y(dataset), the temperature data from Kafka is read periodically and verified, and then stored in the X(dataset). When the data acquisition of each cycle is complete, the data are saved to HDFS and function $Y = Xb + \varepsilon$ is triggered at that time. Based on this, we obtain b(the coefficient of each temperature data item). Finally, by inputting the periodic dataset$X$ to function $Y = Xb + \varepsilon$, the fitted spindle thermal error value can be calculated. Finally, an error compensation operation is performed based on that value.

Through Algorithm 2, the error value resulting from the spindle temperature changes during the machining process can be fitted; then, the machine coordinate system of the NC system can be adjusted online by the zero-point drift method based on the fitted value, to reduce the error.

Algorithm 2: Temperature Compensation Algorithm

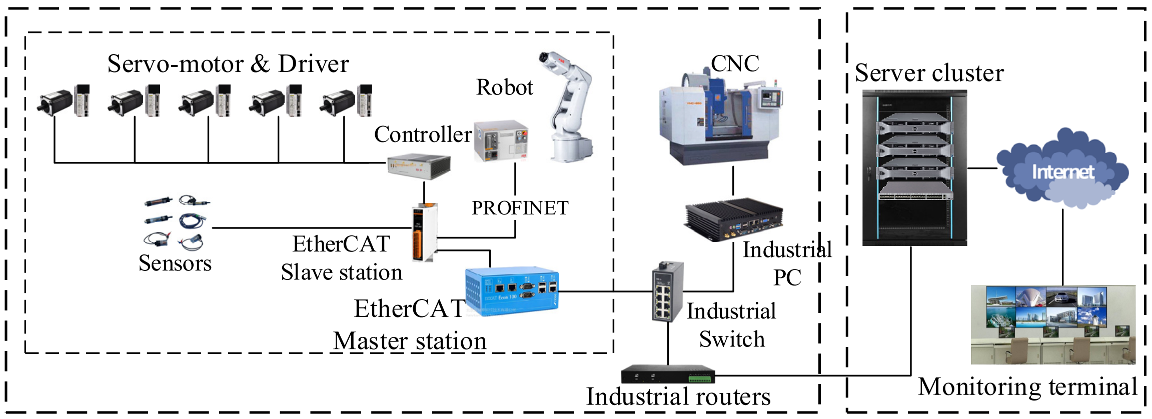
```
Input: X = $\begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ & & \dots & & \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}$ // Spindle
temperature data
Input: $\boldsymbol{\varepsilon}$ // Random error
Input: $\mathbf{Y} = [y_1, y_2, y_3, \dots\dots, y_{m-1}, y_m]^{\mathrm{T}}$ // Spindle
thermal error deformation data
Output: $\hat{\mathbf{b}} = \left[\hat{b}_0, \hat{b}_1, \hat{b}_2, \dots, \hat{b}_m\right]$ // Coefficient of each
temperature $\mathrm{X_{ij}}$
Output: $\hat{\mathbf{Y}}\left[\hat{y}_1, \hat{y}_2, \dots, \hat{y}_m\right]$ // Predicted spindle thermal
error deformation data
Begin
    Initialization
    $\mathbf{Y} = \mathbf{X}\hat{\mathbf{b}} + \boldsymbol{\varepsilon}$
    $\sum_{i=1}^{m} (y_i - \hat{y}_i)^2 =$
    $\min \sum_{j=1}^{n} \left(y_i - \hat{b}_0 - \hat{b}_1 \cdot x_{i1} - \dots - \hat{b}_{m*} \cdot x_{nj}\right)^2$
    //Solve the minimum of $\sum_{i=1}^{m} (y_i - \hat{y}_i)^2$
    for i ← to m
        for j ← to n
            $\varphi =$
            $\sum_{j=1}^{n} \left(y_i - \hat{b}_0 - \hat{b}_1 \cdot x_{i1} - \dots - \hat{b}_m \cdot x_{mj}\right)^2 =$
            0 // Establish function $\varphi$
            $\frac{\partial \varphi}{\partial \hat{b}_j} = 0$ // for each $\hat{b}_j$ Solving partial
            derivatives, find $\hat{b}$
        end for
    end for
    $0 = \mathbf{X^T}(\mathbf{Y} - \mathbf{X}\hat{\mathbf{b}})$
    if $(\mathrm{X^T X^{-1}}$ == true) then
        $\hat{\mathbf{b}} = \mathbf{X^T X^{-1} X^T Y}$ //existing unique value
        $\hat{\mathbf{b}} = \left[\hat{b}_0, \hat{b}_1, \hat{b}_2, \dots, \hat{b}_m\right]$
    end if
    $\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{b}} + \boldsymbol{\varepsilon}$
    $\hat{\mathbf{Y}} = \left[\hat{y}_1, \hat{y}_2, \dots, \hat{y}_m\right]$
end
```

D. INTELLIGENT TOOL WEAR MONITORING

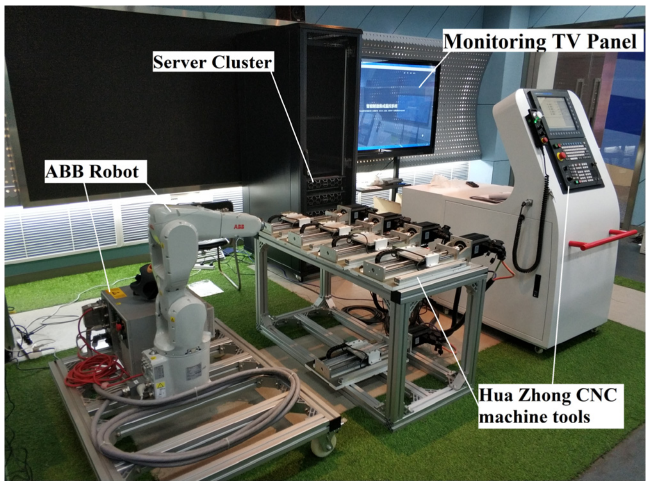
Tool wear can lead to an interruption in the cutting process, causing the workpiece to be scrapped or damage to the machine tool, both of which result in economic losses. Intelligent tool damage detection can reduce or avoid such situations.

There is a strong correlation between tool wear and a machine's power signal [37]; consequently, it is possible to carry out cutting experiments on tools with different degrees of wear and collect corresponding machine power signals to establish a relationship model between tool wear and machine power signals. In addition, machining characteristics and the part-cutting parameters also affect the power signal; thus, these should be included in the input to the model. The specific process for intelligent monitoring of tool wear is shown in Fig 7. Algorithm 1 is used to train the relational model, where *training_algorithm* represents the training algorithm (a NN in this case), and *all_data* represents the corresponding degrees of wear, machining characteristics, cutting parameters and machine power signals.

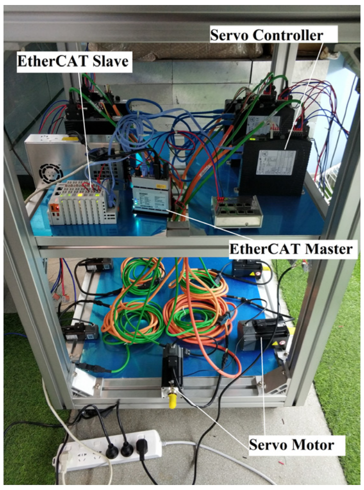
The cutting power signal of the machine generally has a high acquisition frequency and is mixed with environmental noise. If the collected data is uploaded directly to the cloud and processed, not only will the transmission waste network



(a)



(b)



(c)

FIGURE 8. Experimental environment overview (a) overall device networking; (b) physical workshop environment; and (c) EtherCAT module.

bandwidth but the real-time tool monitoring performance will be affected. Therefore, edge computing is introduced to solve this problem. The collected power signal is first down-sampled and denoised at the edge computing node, and then uploaded to the cloud. Another problem is that the parameters for the down-sampling and denoising algorithms used by the edge computing nodes not only differ for different workpieces but also for the same workpiece over time. Thus, a method must be provided to notify the edge computing nodes to update the algorithm parameters, which requires adding a feedback loop to the Lambda architecture.

V. EXPERIMENT AND VERIFICATION

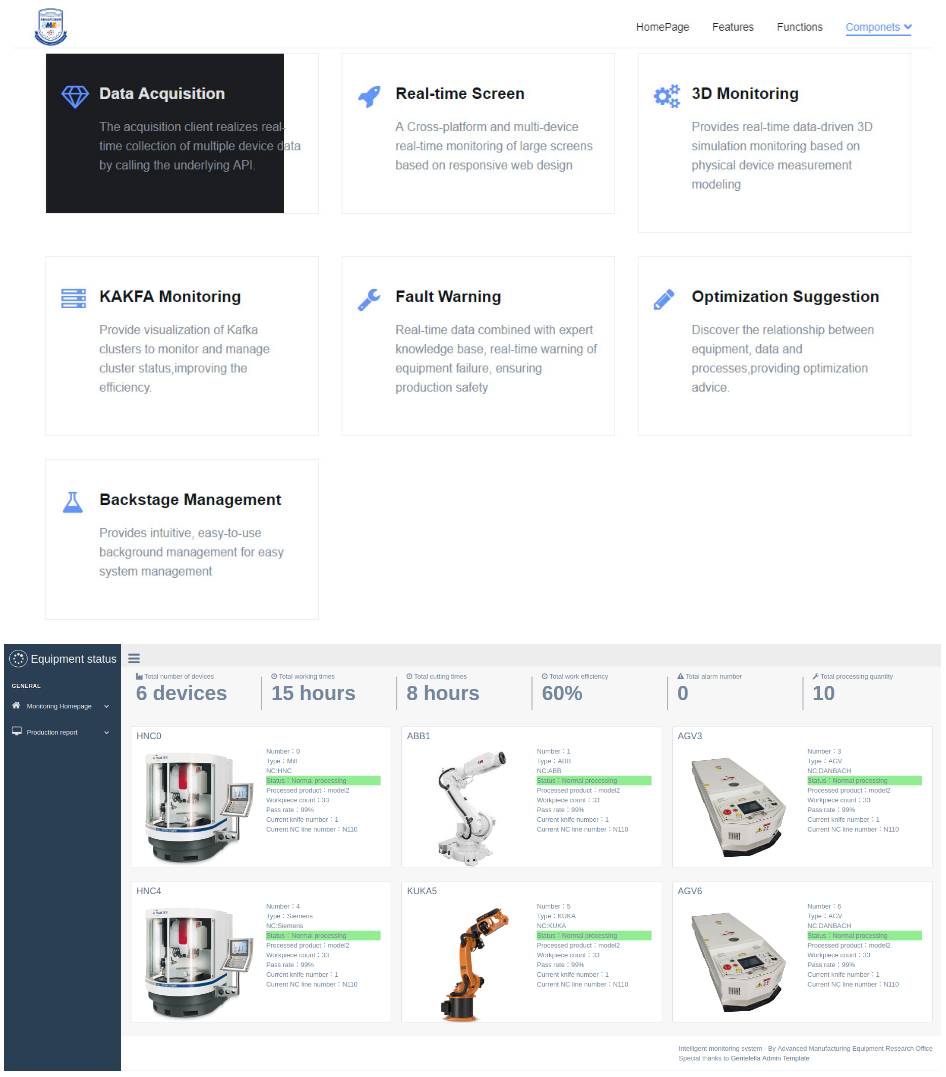
To demonstrate the superiority of the Phi architecture in system structure design, data processing performance and applications in different scenarios, we conducted an experiment in the digital manufacturing workshop simulation built by our team. The workshop configuration and the experimental details are explained in the following sections.

A. EXPERIMENTAL SETUP

To mimic the environment of the aviation manufacturing workshop, the hardware equipment in our lab includes an ABB robot, a Huazhong CNC machine tool, servo motors that represent moving components such as AGVs (automation guided vehicles), EtherCAT master and slave stations (motion control), a server cluster (Dell server, Intel Xeon Silver 4114 2.2G, 10C/20T, 9.6GT/s, 14 M Cache, 4*16 GB RDIMM, 2666 MT/s), and TV panel monitors. The lab building and equipment are shown in Fig. 8(a)-(c):

The server configuration for the simulation workshop includes the operating system, hardware configuration and software configuration. Both Linux and Windows 10 are used as operating systems. The Linux system is used to develop the microservice architecture, construct the API server, and so forth, and the Windows operating system is used to deploy the data acquisition program of the device (due to SDK limitations of the development system).

The entire software of the Phi architecture uses Docker as the container for each Microservice, and the



Data Acquisition
The acquisition client realizes real-time collection of multiple device data by calling the underlying API.

Real-time Screen
A Cross-platform and multi-device real-time monitoring of large screens based on responsive web design

3D Monitoring
Provides real-time data-driven 3D simulation monitoring based on physical device measurement modeling

KAKFA Monitoring
Provide visualization of Kafka clusters to monitor and manage cluster status,improving the efficiency.

Fault Warning
Real-time data combined with expert knowledge base, real-time warning of equipment failure, ensuring production safety

Optimization Suggestion
Discover the relationship between equipment, data and processes,providing optimization advice.

Backstage Management
Provides intuitive, easy-to-use background management for easy system management

Total number of devices	Total working times	Total cutting times	Total work efficiency	Total alarm number	Total processing quantity
6 devices	15 hours	8 hours	60%	0	10

HNC0
Number：0
Type：Mill
NC:HNC
Status：Normal processing
Processed product：model2
Workpiece count：33
Pass rate：99%
Current knife number：1
Current NC line number：N110

ABB1
Number：1
Type：ABB
NC:ABB
Status：Normal processing
Processed product：model2
Workpiece count：33
Pass rate：99%
Current knife number：1
Current NC line number：N110

AGV3
Number：3
Type：AGV
NC:DANBACH
Status：Normal processing
Processed product：model2
Workpiece count：33
Pass rate：99%
Current knife number：1
Current NC line number：N110

HNC4
Number：4
Type：Siemens
NC:Siemens
Status：Normal processing
Processed product：model2
Workpiece count：33
Pass rate：99%
Current knife number：1
Current NC line number：N110

KUKA5
Number：5
Type：KUKA
NC:KUKA
Status：Normal processing
Processed product：model2
Workpiece count：33
Pass rate：99%
Current knife number：1
Current NC line number：N110

AGV6
Number：6
Type：AGV
NC:DANBACH
Status：Normal processing
Processed product：model2
Workpiece count：33
Pass rate：99%
Current knife number：1
Current NC line number：N110

Intelligent monitoring system - By Advanced Manufacturing Equipment Research Office
Special thanks to Gentelella Admin Template

FIGURE 9. **The display of digital workshop.**

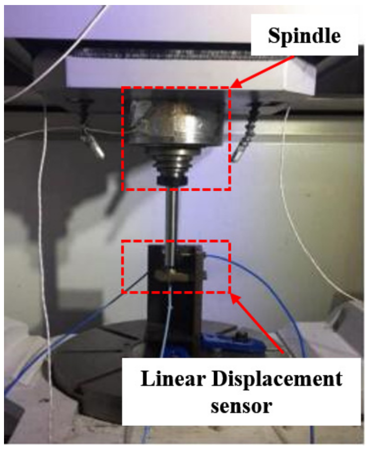
development framework is the Spring Cloud (Spring Boot), which can supply related components such as Zuul (API gateway), Eureka (discovery and registration), etc. Each microservice container operates in isolation based on the PID namespace supplied by the Linux kernel. By restricting the external access API, the routing gateway can proxy user requests to different services through a gateway address, which also supports automatic mapping to microservices registered on Eureka. A restful API is used as a stand-alone application for implementing microservices data management, and it both supplies the API of external calls and is the basis for Eureka's functional implementation. The Web UI development is based on JavaWeb and Thymeleaf. By using the Restful API interface of each microservice, different services can be combined to achieve various applications, such as data acquisition and workshop device status monitoring, as shown in Fig. 9.

B. INDUSTRIAL MULTI-SCENARIO APPLICATION VERIFICATION

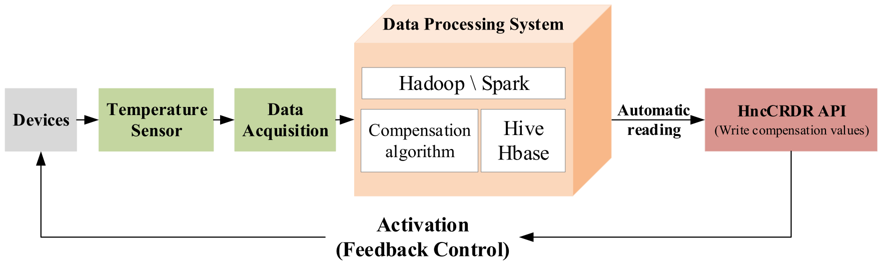
We combined the Phi architecture with a common industrial scenario of aviation manufacturing to verify the feasibility and advantages of the Phi architecture in data processing, the feedback loop and system structural development. The validation experiment includes three main parts: 1) on-line thermal error compensation based on spindle temperature; 2) industrial data monitoring and system throughput testing; and 3) low latency control of servo motors based-on edge computing. The specific implementation and results analysis of the above-mentioned experiment is as follows:

Experiment 1): On-line thermal error compensation based on spindle temperature

This experiment tests the real-time computing and feedback loop of the Phi architecture. By computing the metadata of the spindle temperature and generating the data processing



(a)



(b)

FIGURE 10. **Thermal error feedback compensation (a) experimental platform; (b) compensation process.**

results quickly, we can use the results stored in HBase to realize reverse compensation.

Experimental method: Initially, the external temperature sensor was used to collect the spindle temperature data. Then, the data processing system computed the thermal error compensation algorithm in a timely manner using those data and the calculation results were stored (the error compensation value). Then, through the feedback loop of the Phi architecture, a script was used to read results automatically and write the error compensation values through the HncCRDS API interface of the CNC system (Huazhong CNC) to issue the NC instructions and achieve an on-line offset of zero (CNC zero drift) and correct errors caused by spindle thermal deformation. As a verification experiment, the multivariable linear regression model (Algorithm 2) was used as the compensation algorithm. The specific experimental process and platform are shown in Fig. 10(a)-(b).

Experimental results: Combined with the real-time processing and feedback loop of Phi architecture, operations such as reading results and issuing NC compensation instructions can be implemented much faster, which helps improve the on-line thermal error compensation of the spindle. Through the testing, for each temperature reading, from data acquisition to real-time processing, to updating the calculation results (temperature compensation values) and reading the results, to issuing the NC system thermal error compensation instruction, the entire process time cycle is between 100 ms–500 ms. That result further indicates that the real-time processing and feedback loop of the Phi architecture has practical value for the on-line thermal error compensation of the machine tool spindle under a soft real-time standard. Finally, following the usual process of thermal error compensation experiments, the experiment was executed with a sampling period of 0.04 h (data set). The results of the spindle thermal error compensation are shown in Fig. 11; and the error after compensation was maintained at ($\pm 5\ \mu$m).

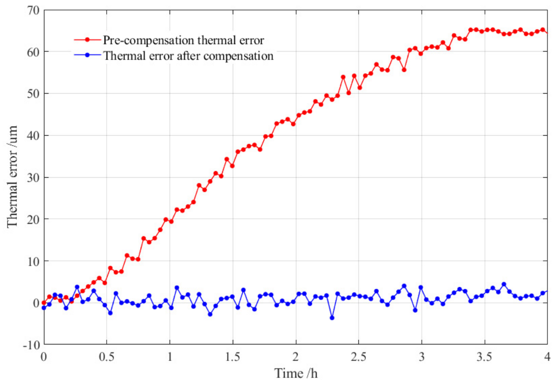


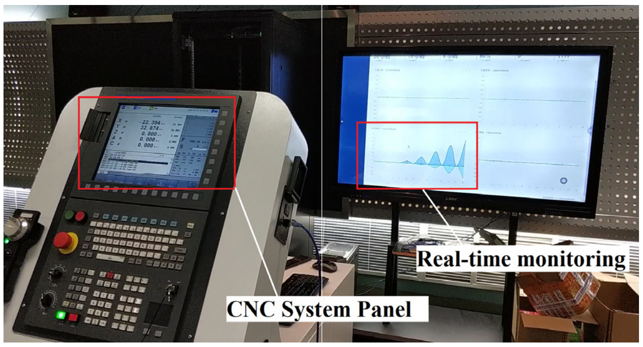
FIGURE 11. **Spindle thermal error compensation results.**

Experiment 2): Industrial data monitoring and system throughput testing

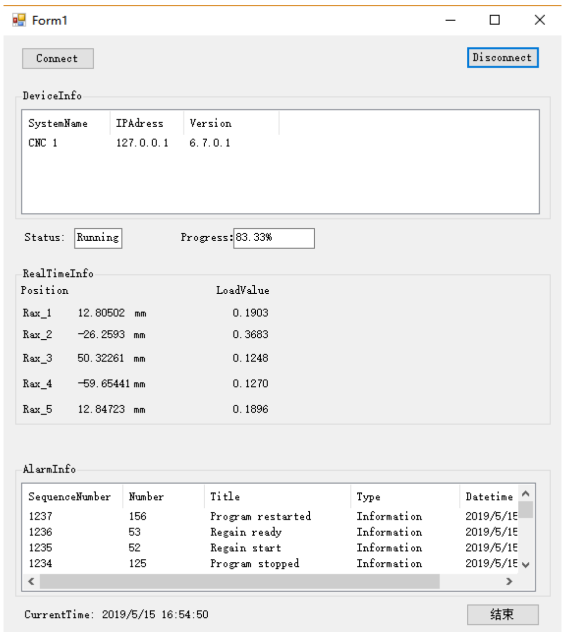
This goal of this experiment is to achieve a combination application consisting of the data acquisition service, transmission service and monitoring service. Meanwhile, it tests the system performance to confirm whether the entire system can withstand the requirements of different industrial scenarios.

Experimental method: Initially, through calling the API interface of each microservice, we achieved the combination of data acquisition service, transmission service and monitoring service. Then, we deployed this combination service into the workshop to collect status data from the CNC machine tools and display that status information on the workshop monitoring panel. Finally, we added the test dataset and tools provided by Kafka (kafka-producer-perf-test.sh) to test the performance of the entire system.

Experimental results: First, the above combination service was verified in the aviation manufacturing workshop simulation. The results showed that the development structure of the Phi architecture was highly adaptable, as shown



(a)



DeviceInfo

SystemName	IPAdress	Version
CNC 1	127.0.0.1	6.7.0.1

Status: Running   Progress: 83.33%

RealTimeInfo

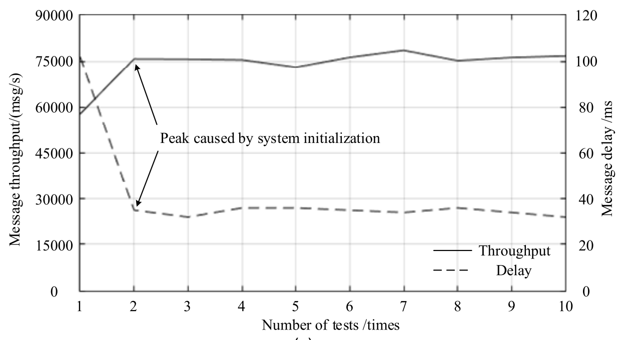
Position			LoadValue
Rax_1	12.80502	mm	0.1903
Rax_2	-26.2593	mm	0.3683
Rax_3	50.32261	mm	0.1248
Rax_4	-59.65441	mm	0.1270
Rax_5	12.84723	mm	0.1896

AlarmInfo

SequenceNumber	Number	Title	Type	Datetime
1237	156	Program restarted	Information	2019/5/15
1236	53	Regain ready	Information	2019/5/15
1235	52	Regain start	Information	2019/5/15
1234	125	Program stopped	Information	2019/5/15

CurrentTime: 2019/5/15 16:54:50

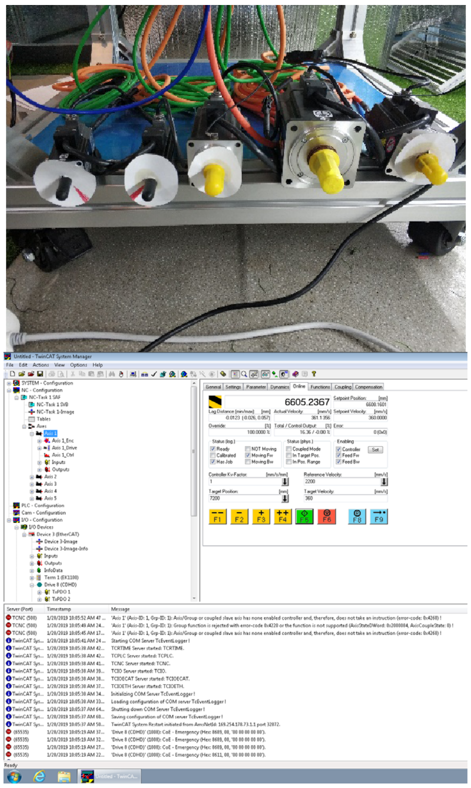
(b)



(c)

FIGURE 12. The operational status (a) status monitoring; (b) adaptive acquisition; and (c) system performance testing.

in Fig. 12(a)-(b). Moreover, by combining the data acquisition service and the Kafka testing tools, we tested the entire system throughput performance. The results are shown in Fig. 12(c). System throughput was maintained at 75,000 msg/s under stable conditions, and system delay ranged between 35 ms and 40 ms. These results show that the data processing system can fully meet the data processing requirements of the aviation manufacturing workshop.



```
6605.2367
Setpoint Position: [mm] 6600.1601
Lag Distance (min/max): [mm] -0.0123 (-0.026, 0.057)
Actual Velocity: [mm/s] 361.1356
Setpoint Velocity: [mm/s] 360.0000
Override: [%] 100.0000 %
Total / Control Output: [%] 16.36 / -0.00 %
Error: 0 (0x0)
Status (log.): [✓] Ready  [ ] Calibrated  [✓] Has Job  [ ] NOT Moving  [✓] Moving Fw  [ ] Moving Bw
Status (phys.): [ ] Coupled Mode  [ ] In Target Pos.  [ ] In Pos. Range
Enabling: [✓] Controller  [✓] Feed Fw  [✓] Feed Bw
Controller Kv-Factor: [mm/s/mm] 1
Reference Velocity: [mm/s] 2200
Target Position: [mm] 7200
Target Velocity: [mm/s] 360
```

```
Server (Port)    Timestamp                    Message
TCNC (500)       1/28/2019 10:05:52 AM 47 ...  'Axis 1' (Axis-ID: 1, Grp-ID: 1): Axis/Group or coupled slave axis has none enabled controller and, therefore, does not take an instruction (error-code: 0x4260) !
TCNC (500)       1/28/2019 10:05:49 AM 24...   'Axis 1' (Axis-ID: 1, Grp-ID: 1): Group function is rejected with error-code 0x4220 or the function is not supported (AxisStateDWord: 0x2000084, AxisCoupleState: 0) !
TCNC (500)       1/28/2019 10:05:45 AM 17...   'Axis 1' (Axis-ID: 1, Grp-ID: 1): Axis/Group or coupled slave axis has none enabled controller and, therefore, does not take an instruction (error-code: 0x4260) !
TwinCAT Sys...   1/28/2019 10:05:41 AM 24 ...  Starting COM Server TcEventLogger !
TwinCAT Sys...   1/28/2019 10:05:38 AM 42...   TCRTIME Server started: TCRTIME.
TwinCAT Sys...   1/28/2019 10:05:38 AM 42...   TCPLC Server started: TCPLC.
TwinCAT Sys...   1/28/2019 10:05:38 AM 41...   TCNC Server started: TCNC.
TwinCAT Sys...   1/28/2019 10:05:38 AM 39...   TCIO Server started: TCIO.
TwinCAT Sys...   1/28/2019 10:05:38 AM 38...   TCIOECAT Server started: TCIOECAT.
TwinCAT Sys...   1/28/2019 10:05:38 AM 37...   TCIOETH Server started: TCIOETH.
TwinCAT Sys...   1/28/2019 10:05:38 AM 34...   Initializing COM Server TcEventLogger !
TwinCAT Sys...   1/28/2019 10:05:38 AM 33...   Loading configuration of COM server TcEventLogger !
TwinCAT Sys...   1/28/2019 10:05:37 AM 64...   Shutting down COM Server TcEventLogger !
TwinCAT Sys...   1/28/2019 10:05:37 AM 60...   Saving configuration of COM server TcEventLogger !
TwinCAT Sys...   1/28/2019 10:05:37 AM 58...   TwinCAT System Restart initiated from AmsNetId: 169.254.178.73.1.1 port 32872.
(65535)          1/28/2019 10:05:19 AM 37...   'Drive 8 (CDHD)' (1008): CoE - Emergency (Hex: 8689, 00, '00 00 00 00 00').
(65535)          1/28/2019 10:05:19 AM 32...   'Drive 8 (CDHD)' (1008): CoE - Emergency (Hex: 8689, 00, '00 00 00 00 00').
(65535)          1/28/2019 10:05:19 AM 27...   'Drive 8 (CDHD)' (1008): CoE - Emergency (Hex: 8689, 00, '00 00 00 00 00').
(65535)          1/28/2019 10:05:19 AM 22...   'Drive 8 (CDHD)' (1008): CoE - Emergency (Hex: 8611, 00, '00 00 00 00 00').
```

FIGURE 13. Low-latency control of servo motor.

Experiment 3): Low latency control of servo motors based-on edge computing.

This project verifies the edge computing performance by initiating the computing service at the data acquisition end, allowing the collected data to be processed and analyzed in a timely manner and achieve real-time control under abnormal device conditions.

Experimental method: By deploying the acquisition program for the servo motor on the host computer of the EtherCAT module, servo operating information is read in real time. Due to the computing performance available in the edge computing module, the motor data can be processed immediately. The limit value of the motor position is be preset in the module. If abnormal data are read that exceed the pre-set value, a motor stop command will be triggered directly through the edge computing module. This experiment achieves low-latency control and verifies the validity of the edge calculation.

Experimental results: When reading the position data in the data stream, if the value exceeds the limit, the motor is quickly shut down by a triggered pre-coded script. In normal conditions, the control process would need to undergo data acquisition, transmission, processing (workshop data processing center), issuing control instructions and other links. Finally, to achieve control of the servo motor, that process must be implemented within 500 ms−2 s (specific times are related to data complexity and data volume). In this experiment, we reduced the frequency of the data transmission link by using the edge calculation module to process and analyze the collected data directly at the device-end; consequently, the data processing efficiency was improved to a certain extent (the entire control implementation requires between 100 ms and 1s). This result also served to verify the effectiveness of the Phi architecture edge calculation the experiment is depicted in Fig. 13.

The above experiment verified that the overall logic of the feedback loop in the Phi architecture is correct. Compared with the Lambda architecture, the Phi architecture can achieve real-time reverse control of devices by automatically reading the data results. Moreover, the edge computing layer of the Phi architecture can improve the response speed of the feedback control.

Additionally, the combined application test further validates the flexibility of the system developed using microservices compared with a monolithic architecture. However, the processing speed and feedback control time fall into the range of "soft real-time"; it fails to reach the "hard real-time" standard. This result is related to the amount of collected data and device performance; thus, under certain conditions, system performance could be improved. In addition, edge computing devices could be configured to provide better computational efficiency.

VI. CONCLUSIONS

In this paper, we discuss the defects of traditional data processing architecture regarding their ability to feedback data processing results and perform cloud-based data processing. Then, we propose a new data processing architecture called Phi architecture that combines feedback and edge computing to compensate for the shortcomings of traditional data processing architecture. To improve the stability, flexibility and expansibility of the system, we adopt a microservices development approach to implement the Phi architecture. Finally, we verified the feasibility of the Phi architecture and the superiority of the system development method by implementing several experimental test cases in a simulated digital manufacturing workshop. The novelty and main contributions of this paper are briefly listed below.

1) The most significant feature of the Phi architecture is that it integrates a feedback loop and edge computing. Via the feedback loop, the transport layer and the processing layer are directly associated with the device layer, which is suitable for realizing multiple application scenarios in aviation manufacturing. The addition of edge computing supplies low-latency data processing while also reducing unnecessary data transmission. This new architecture can enable many applications that require real-time data processing, such as real-time workshop monitoring and low-latency machine tool control.
2) The new Phi architecture extends the application scene for aviation manufacturing. The Phi architecture can satisfy various industrial application scenarios, such as real-time and batch flow monitoring of manufacturing data, low latency equipment control, on-line compensation during the manufacturing process, hierarchical industrial data management and flow control.
3) Drawing lessons from the development model of social big data, the Phi architecture integrated new data processing technologies such as Hadoop, Spark and Kafka. These technologies can effectively solve the batch processing and data storage issues, meet the real-time processing demands of streaming data, and guarantee the real-time transmission of massive data. The microservices development approach reduces internal interference and enhances stability in complex industrial environments. Characteristics such as independent development and deployment and API interactions also help in supporting multi-scenario applications in industrial environments.
4) The experiments with thermal error compensation and low-latency application control indicated that the feedback and edge computing features of the Phi architecture are justified and effective. The real-time monitoring implementation demonstrates the architecture's real-time processing capability and the feasibility of creating combined applications with the Phi architecture. Generally, Phi architecture provides a better data solution for intelligent digital construction of an aviation manufacturing workshop.

REFERENCES

[1] K. Thramboulidis, D. C. Vachtsevanou, and A. Solanos, "Cyber-physical microservices: An IoT-based framework for manufacturing systems," in *Proc. IEEE Ind. Cyber-Phys. Syst. (ICPS)*, May 2018, pp. 232–239.

[2] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, "Intelligent manufacturing in the context of industry 4.0: A review," *Engineering*, vol. 3, no. 5, pp. 616–630, 2017.

[3] R. Baheti and H. Gill, "Cyber-physical systems," *Impact Control Technol.*, vol. 12, pp. 161–166, Mar. 2011.

[4] J. Lee, B. Bagheri, and H. A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manuf. Lett.*, vol. 3, pp. 18–23, Jan. 2015.

[5] H. O. Unver, "An ISA-95-based manufacturing intelligence system in support of lean initiatives," *Int. J. Adv. Manuf. Technol.*, vol. 65, nos. 5–8, pp. 853–866, 2013.

[6] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl, "Industry 4.0—An Introduction in the phenomenon," *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 8–12, 2016.

[7] J. Thönes, "Microservices," *IEEE Softw.*, vol. 32, no. 1, p. 116, Jan./Feb. 2015.

[8] N. Marz and J. Warren, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. New York, NY, USA: Manning, 2015.

[9] J. Jiang, "An improved cyber-physical systems architecture for industry 4.0 smart factories," in *Proc. Int. Conf. Appl. Syst. Innov. (ICASI)*, Sapporo, Japan, May 2017, pp. 918–920.

[10] D. Mourtzis and E. Vlachou, "A cloud-based cyber-physical system for adaptive shop-floor scheduling and condition-based maintenance," *J. Manuf. Syst.*, vol. 47, pp. 179–198, Apr. 2018.

[11] Y. Ye, T. Hu, C. Zhang, and W. Luo, "Design and development of a CNC machining process knowledge base using cloud technology," *Int. J. Adv. Manuf. Technol.*, vol. 94, nos. 9–12, pp. 3413–3425, 2018.

[12] O. Kwon, N. Lee, and B. Shin, "Data quality management, data usage experience and acquisition intention of big data analytics," *Int. J. Inf. Manage.*, vol. 34, no. 3, pp. 387–394, 2014.

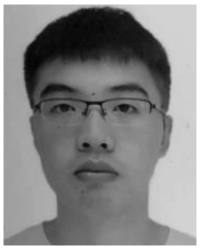
[13] Z. Hasani, M. Kon-Popovska, and G. Velinov, "Survey of technologies for real time big data streams analytic," in *Proc. 11th Int. Conf. Informat. Inf. Technol.*, Apr. 2014, pp. 11–13.

[14] Y. Zhang, S. Ren, Y. Liu, and S. Si, "A big data analytics architecture for cleaner manufacturing and maintenance processes of complex products," *J. Cleaner Prod.*, vol. 142, pp. 626–641, Jan. 2017.
[15] B. Yadranjiaghdam, N. Pool, and N. Tabrizi, "A survey on real-time big data analytics: Applications and tools," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Las Vegas, NV, USA, Dec. 2016, pp. 404–409.
[16] V. Persico, A. Pescapé, A. Picariello, and G. Sperlí, "Benchmarking big data architectures for social networks data processing using public cloud platforms," *Future Gener. Comput. Syst.*, vol. 89, pp. 98–109, Dec. 2018.
[17] J. Lin, "The lambda and the kappa," *IEEE Internet Comput.*, vol. 21, no. 5, pp. 60–66, Sep. 2017.
[18] Z. Hasani, M. Kon-Popovska, and G. Velinov, "Lambda architecture for real time big data analytic," in *Proc. ICT Innov.*, 2014, pp. 133–143.
[19] M. Kiran, P. Murphy, I. Monga, J. Dugan, and S. S. Baveja, "Lambda architecture for cost-effective batch and speed big data processing," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Oct./Nov. 2015, pp. 2785–2792.
[20] M. Vögler, J. M. Schleicher, C. Inzinger, and S. Dustdar, "Ahab: A cloud-based distributed big data analytics framework for the Internet of Things," *Softw., Pract. Exper.*, vol. 47, no. 3, pp. 443–454, 2017.
[21] A. Batyuk and V. Voityshyn, "Streaming process discovery for lambda architecture-based process monitoring platform," in *Proc. IEEE 13th Int. Sci. Tech. Conf. Comput. Sci. Inf. Technol. (CSIT)*, Lviv, Ukraine, Sep. 2018, pp. 298–301.
[22] U. Suthakar, L. Magnoni, D. Smith, and A. Khan, "Optimised Lambda Architecture for monitoring scientific infrastructure," *IEEE Trans. Parallel Distrib. Syst.*, to be published.
[23] J. Wan, S. Tang, D. Li, S. Wang, C. Liu, H. Abbas, and A. V. Vasilakos, "A manufacturing big data solution for active preventive maintenance," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2039–2047, Aug. 2017.
[24] L. Wang and R. Ranjan, "Processing distributed Internet of Things data in clouds," *IEEE Cloud Comput.*, vol. 2, no. 1, pp. 76–80, Feb. 2015.
[25] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
[26] P. Corcoran and S. K. Datta, "Mobile-edge computing and the Internet of Things for consumers: Extending cloud computing and services to the edge of the network," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 73–74, Oct. 2016.
[27] J.-S. Fu, Y. Liu, H.-C. Chao, B. K. Bhargava, and Z.-J. Zhang, "Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4519–4528, Oct. 2018. doi: 10.1109/TII.2018.2793350.
[28] D. Georgakopoulos, P. P. Jayaraman, M. Fazia, M. Villari, and R. Ranjan, "Internet of Things and edge cloud computing roadmap for manufacturing," *IEEE Cloud Comput.*, vol. 3, no. 4, pp. 66–73, Jul./Aug. 2016.
[29] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2016, pp. 20–26.
[30] P. G. Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, 2015.
[31] W. Shi and S. Dustdar, "The Promise of Edge Computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.
[32] C. M. FernÃąndez, M. D. RodrÃguez, and B. R. Muñoz, "An edge computing architecture in the Internet of Things," in *Proc. IEEE 21st Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2018, pp. 99–102.
[33] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4548–4556, Oct. 2018.
[34] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, and Q. Zhang, "Edge computing in IoT-based manufacturing," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 103–109, Sep. 2018.
[35] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.
[36] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.
[37] H. Shao, H. L. Wang, and X. M. Zhao, "A cutting power model for tool wear monitoring in milling," *Int. J. Mach. Tools Manuf.*, vol. 44, no. 14, pp. 1503–1509, Nov. 2004.

WEI WANG received the B.Sc. degree in mechanical engineering from Wuhan University, Wuhan, China, and the M.Sc. and Ph.D. degrees in mechanical engineering from the Harbin Institute of Technology, Harbin, China. He is currently a Professor with the Department of Mechanical and Electrical Engineering, University of Electronic Science and Technology of China. He has been involved in aircraft manufacturing for over 15 years. His research interests include smart manufacturing, industrial big data processing, data mining, and measurement in the manufacturing process. He has been serving as an ISO/TC39/SC2 machine tools test condition committee member.

LEI FAN was born in Mianyang, Sichuan, China, in 1994. He received the B.S. degree in industrial engineering from Guizhou University, Guiyang, China, in 2017. He is currently pursuing the M.S. degree in mechanical engineering with the University of Electronic Science and Technology of China, Chengdu, Sichuan, China. His research interests include data acquisition and transmission in the aviation manufacturing process, intelligent manufacturing, big data handling technology, the development of industrial cloud computing, microservices development, and applications.

PU HUANG received the B.S. degree in mechanical engineering from the University of Electronic Science and Technology of China, Chengdu, Sichuan, China, in 2016, where he is currently pursuing the M.S. degree in mechanical engineering. His research interests include the development of artificial intelligence, intelligent manufacturing, machine learning and deep learning algorithms, data acquisition and data mining in the manufacturing process, big data handling technology, and applications.



HAI LI received the B.S. degree in civil engineering from the Jincheng College of Sichuan University, Chengdu, Sichuan, China, in 2015, and the M.S. degree in mechanical engineering from the University of Electronic Science and Technology of China, Chengdu, in 2017, where he is currently pursuing the Ph.D. degree in mechanical engineering. He has authored five articles and holds one patent. His research interests include intelligent manufacturing, including data collection, remote monitoring of machine, fault diagnosis and prediction, and production scheduling.

• • •